

University of Wollongong - Research Online

Thesis Collection

Title: Contributions to secure and privacy-preserving use of electronic credentials

Author: Siamak F Shahandashti

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

Contributions to secure and
privacy-preserving use of electronic
credentials

Siamak Fayyaz Shahandashti
University of Wollongong

Shahandashti, Siamak Fayyaz, Contributions to secure and privacy-preserving use of electronic credentials, Doctor of Philosophy thesis, School of Computer Science and Software Engineering - Faculty of Informatics, University of Wollongong, 2009. <http://ro.uow.edu.au/theses/3036>

This paper is posted at Research Online.

NOTE

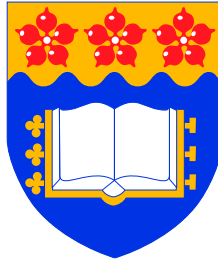
This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

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.



Contributions to Secure and Privacy-Preserving Use of Electronic Credentials

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Siamak Fayyaz Shahandashti

School of Computer Science and Software Engineering
Faculty of Informatics
October 2009

© Copyright 2009

by

Siamak Fayyaz Shahandashti

All Rights Reserved

Dedicated to

my wife: Sara

my mum: Behdokht

and my dad: Ali

Certification

I, Siamak Fayyaz Shahandashti, declare that this thesis, submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science and Software Engineering, Faculty of Informatics, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Siamak Fayyaz Shahandashti
October 5, 2009

Abstract

In this thesis, we make contributions to secure and privacy preserving use of electronic credentials in three different levels.

First, we address the case in credential systems where a credential owner wants to show her credential to a verifier without taking the risk that the ability to prove ownership of her credential is transferred to the verifier. We define *credential ownership proof* protocols for credentials signed by standard signature schemes. We also propose proper security definitions for the protocol, aiming to protect the security of both the credential issuer and the credential owner against concurrent attacks. We give two generic constructions of credential ownership proofs based on identity-based encryption and identity-based identification schemes. Furthermore, we show that signatures with credential ownership proofs are equivalent to identity-based identification schemes, in the sense that any secure construction of each implies a secure construction of the other. Moreover, we show that the GQ identification protocol yields an efficient credential ownership proof for credentials signed by the RSA signature scheme and prove the protocol concurrently-secure.

Then, we give a generic construction for universal (mutli) designated-verifier signature schemes from a large class of signature schemes, referred to as Class \mathbb{C} . The resulting schemes are efficient and have two important properties. Firstly, they are provably DV-unforgeable, non-transferable and also non-delegatable. Secondly, the signer and the designated verifier can independently choose their cryptographic settings. We also propose a generic construction for (hierarchical) identity-based signature schemes from any signature scheme in \mathbb{C} and prove that the construction is secure against adaptive chosen message and identity attacks. We discuss possible extensions of our constructions to identity-based ring signatures and identity-based designated-verifier signatures from any signature in \mathbb{C} . Furthermore, we show that it is possible to combine the above constructions to obtain signatures with combined functionalities.

Finally, inspired by the recent developments in attribute-based encryption, we propose *threshold attribute-based signatures* (t-ABS). In a t-ABS, signers are associated with a set of attributes and verification of a signed document against a verification attribute set succeeds if the signer has a threshold number of (at least t) attributes in common with the verification attribute set. A t-ABS scheme enables a signature holder to prove possession of signatures by revealing only the relevant (to the verification attribute set) attributes of the signer, hence providing *signer-attribute privacy* for the signature holder. We define t-ABS schemes, formalize their security and propose two t-ABS schemes: a basic scheme secure against selective forgery and a second one secure against existential forgery, both provable in the standard model, assuming hardness of the computational Diffie-Hellman problem. We

show that our basic t-ABS scheme can be augmented with two extra protocols that are used for efficiently issuing and verifying t-ABS signatures on committed values. We call the augmented scheme a threshold attribute based c-signature scheme (t-ABCS). We show how a t-ABCS scheme can be used to realize a secure *threshold attribute-based anonymous credential system* (t-ABACS) providing signer-attribute privacy. We propose a security model for t-ABACS and give a concrete scheme using t-ABCS scheme. Using the simulation paradigm, we prove that the credential system is secure if the t-ABCS scheme is secure.

Acknowledgments

I would like to start by thanking my supervisor *Rei* (Professor Reihaneh Safavi-Naini). Her vast knowledge of the area and her macroscopic intuition of the bits and pieces in the field assisted me to obtain a better understanding of cryptographic research. Although only few can come close to her twelve-hour-per-day average working time! She has also been of great support in my non-academic life.

From the beginning of my candidature in the now School of Computer Science and Software Engineering (SCSSE), then School of IT and CS (SITACS) at University of Wollongong, I have been based in 3.234 with lab-mates that I have the honour to call ‘friends’ now. I would like to credit these guys since without them it would have been a much harder job to live in a whole new country and do a PhD. I would like to specially thank *Angela* (Angela Piper), *Noi* (Rungrat Wiangsripanawan), and *Reza* (Mohammad Reza Reyhanitabar) for the uncountable little and big things they have done for me. I also extend my acknowledgment to my friends *Allen* (Man Ho Au), *Jeff* (Dr. Jeffrey Horton), *John* (Tsz Hon Yuen), *Martin* (Jan Martin Surminen), *Michael* (Wenming Lu), *Pairat* (Pairat Thorcharoensri), *Xinyi* (Xinyi Huang), and *Wei* (Wei Wu).

I spent almost three months in the *iCORE* Information Security Lab of the University of Calgary as a visitor and I would like to thank the lab for hosting me. During this period I made new friends, *Jason* (Dr. M. Jason Hinek) and *Michal* (Dr. Michal Sramka), and here I would like to thank them for being there for me. I also like to thank Dr. Shaoquan Jiang for the fruitful discussions we had.

I would like to thank *James* (James Atkinson), head of Campus East, a place that I called home for almost 3 years. James trusted me with a residential advisor position for 2 years which was both a unique life learning experience and a considerable financial assistance.

I thank Professor Willy Susilo for helping me out here and there. I appreciate Professor Philip Ogunbona, my co-supervisor, and Dr. Jonsang Baek’s guidance of my thesis. I also thank Professor Jennifer Seberry, Associate Professor Yi Mu, and Professor Farzad Safaei for their kindness towards me. I would like to mention Dr. Shuhong Wang for the usefull discussions we had.

I extend my gratitude to the anonymous reviewers of ASIACCS ’07, PKC ’08, AfricaCrypt ’09, and the IET Journal of Information Security for their comments on my papers, and similarly to Professor Colin Boyd of the Information Security Institute of the Queensland University of Technology and Associate Professor Michael Jacobson of the Institute for Security, Privacy and Information Assurance of the University of Calgary, my thesis examiners, for their extremely comprehensive and useful comments on my thesis.

Last, but certainly not least, I am most grateful of the love of my life, my lovely wife, *Sara*, for accompanying me in diving to a wholly unknown new life in Australia and supporting me through the period. I also thank my mum and dad, *Behdokht* and *Ali*, for their total support and love during the almost three decades of my life. I am grateful of my brothers, *Siavash* and *Kiavash*, for not saying no to any of the little annoying favours I asked them during these years. I thank my parents-in-law, *Maryam* and *Hassan*, and my brother-in-law, *Ata*, for their love and support.

Publications

The following publications have arisen from the research carried out during the PhD candidature and hence the material in this thesis is largely based on them.

- [SSB07] Siamak F Shahandashti, Reihaneh Safavi-Naini, and Joonsang Baek. Concurrently-Secure Credential Ownership Proofs. In Feng Bao and Steven Miller, editors, *ASIACCS '07*, pages 161–172. ACM, 2007.
- [SS08] Siamak F 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 (PKC '08)*, volume 4939 of *Lecture Notes in Computer Science*, pages 121–140. Springer, 2008.
- [SS09b] Siamak F Shahandashti and Reihaneh Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In Bart Preneel, editor, *AfricaCrypt '09*, volume 5580 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2009.
- [SS09a] Siamak F Shahandashti and Reihaneh Safavi-Naini. Generic Constructions for Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. *IET Information Security*, (to appear), 2009.

Free personal versions of the above publications are available via the World-Wide Web as follows:

- [SSB06] Siamak F Shahandashti, Reihaneh Safavi-Naini, and Joonsang Baek. Concurrently-Secure Credential Ownership Proofs. Available through corresponding author's home page: <http://sites.google.com/site/siamax/>. Full version of [SSB07].
- [SS07] Siamak F Shahandashti and Reihaneh Safavi-Naini. Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. Cryptology ePrint Archive, Report 2007/462, 2007. <http://eprint.iacr.org/2007/462>. Full version of [SS08].
- [SS09c] Siamak F Shahandashti and Reihaneh Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. Cryptology ePrint Archive, Report 2009/126, 2009. <http://eprint.iacr.org/2009/126>. Full version of [SS09b].

Notation

Xyz	algorithm Xyz
XYZ	security notion XYZ
\mathcal{X}_{yz}	oracle \mathcal{X}_{yz}
Xyz	string Xyz
ε	the empty string
$\mathbb{P}\text{oly}(k)$	the set of all algorithms polynomial time in k
St_{X}	internal state information of algorithm X
\parallel	concatenation
\setminus	set subtraction
$ x $	bit length of the quantity x
$ S $	cardinality of the set S
$\varphi(\cdot)$	Euler's totient function
$x \leftarrow a$	value a is assigned to variable x
$x \stackrel{N}{\leftarrow} a$	value $a \bmod N$ is assigned to variable x
$x \stackrel{\$}{\leftarrow} X$	a member is chosen randomly from set X and assigned to variable x
$x \leftarrow \mathsf{X}(a; r : \mathcal{O})$	algorithm X with access to oracle \mathcal{O} , input a , and random tape r is run and the output is assigned to variable x
$A \dashv(X) \rightarrow B \mid C$	A sends X to B if condition C holds
$(s, t) \leftarrow [\mathsf{X}(x) \leftrightarrow \mathsf{Y}(y)](a)$	interactive protocol between X with private input x and Y with private input y is run with public input a , X outputs s and Y outputs t
$\text{Tr}[\mathsf{X}(x) \leftrightarrow \mathsf{Y}(y)](a)$	transcript of a protocol run with public input a between X with private input x and Y with private input y
$\text{ZK-PoK}\{x : a = g^x\}$	zero knowledge proof of knowledge of x such that $a = g^x$, where a and g are public inputs to the protocol
$\mathsf{X}(a)$	algorithm X with input a and description [desc.] is run and x is
[desc.]	returned as output
Return x	

Contents

Abstract	V
Acknowledgments	VII
Publications	IX
Notation	X
1 Introduction	1
1.1 Credential Ownership Proofs	4
1.2 Universal Designated Verifier Signatures	6
1.2.1 Further Identity-Based Constructions	8
1.3 Attribute-Based Signatures	9
1.4 Attribute-Based Anonymous Credential Systems	11
2 Preliminaries	14
2.1 Public Key Cryptography and Provable Security	14
2.2 Computational Assumptions	16
2.3 Interactive Proof Systems	18
2.4 Identification and Signature Schemes	21
2.5 Identity-Based Cryptography	25
3 Concurrently-Secure Credential Ownership Proofs	29
3.1 Introduction	29
3.1.1 Related Work	30
3.1.2 Our Contributions	31
3.2 Defining Credential Ownership Proofs	32
3.2.1 Defining Credential Ownership Proof Security	33
3.2.2 Security Treatment of Baek et al.	36
3.3 Generic Construction from IBE	37
3.3.1 IBE and Its Security	38
3.3.2 Naor Transform	39
3.3.3 IBE-Based COP	40
3.4 Equivalence with IBI	42
3.5 Efficient COP from GQ	43

3.5.1	The GQ Identification Scheme	44
3.5.2	RSA-FDH Credential Ownership Proof	45
3.5.3	Efficiency of the Scheme	48
3.6	Concluding Remarks	49
4	Generic Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures	51
4.1	Introduction	51
4.1.1	Our Contributions	52
4.1.2	Related Work	55
4.2	Preliminaries	56
4.2.1	Proofs of Knowledge	56
4.2.2	The Fiat-Shamir Transform	57
4.2.3	On Public-Private Key Pairs	59
4.2.4	The Forking and Reset Lemmas	60
4.3	Defining the Class \mathbb{C} of Signatures	62
4.3.1	On Simulatability of Signature Schemes	64
4.3.2	Examples of Signatures in Class \mathbb{C}	65
4.4	Universal Designated Verifier Signatures	66
4.4.1	Definition and Security	66
4.4.2	Generic Construction of UDVS And Its Security	72
4.4.3	Comparison	81
4.5	Identity-based Signatures	83
4.5.1	Generic Construction of IBS and Its Security	83
4.5.2	Comparison	87
4.6	Combined Constructions	89
4.7	Concluding Remarks	90
5	Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems	92
5.1	Introduction	92
5.1.1	Our Contributions	94
5.1.2	Related Work	95
5.2	Notation and Preliminaries	96
5.3	Threshold Attribute-Based Signatures	97
5.3.1	Definition	97
5.3.2	Additional Protocols	99
5.3.3	Constructions	101
5.4	Threshold Attribute-Based C-Signatures	106
5.4.1	Definition	107
5.4.2	Construction	108
5.5	Threshold Attribute-Based Anonymous Credential Systems	114
5.5.1	Security Framework	114

5.5.2	Ideal Model for t-ABACS	116
5.5.3	A Concrete t-ABACS System	118
5.6	Concluding Remarks	123
6	Conclusions and Open Problems	125
	Bibliography	130
	Index	141

List of Tables

3.1	Comparison of security notions for IBE with the new notion highlighted	38
3.2	Equivalence between S+COPs and IBIs	43
3.3	Comparison of RSA-COP Costs with other ZK Solutions	49
4.1	Examples of Σ protocols for proof of knowledge	57
4.2	Examples of signatures in \mathbb{C}	66
4.3	Examples of signatures in \mathbb{C} (Cont'd)	67
4.4	Comparison of previous UDVS schemes with our GUDVS counterparts	82
4.5	Comparison of previous UMDVS schemes with their GUMDVS counterparts	84
4.6	Four new IBS schemes based on discrete logarithms, RSA, and pairings	89
4.7	Comparison of previous IBUDVS schemes with counterparts in our construction	90
6.1	Credential ownership proofs in comparison with other credential system solutions . . .	126
6.2	A summary of constructions in Chapter 4	127
6.3	Summary of how our UDVS constructions compare with previous schemes	128

List of Figures

2.1	The RSA experiment	16
2.2	One more RSA inversion experiment	17
2.3	The discrete logarithm and CDH experiments	18
2.4	A three-move public-coin protocol in the canonical form	19
2.5	Identification scheme IMP-ATK-security experiments ($\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$)	23
2.6	Signature scheme EUF-CMA-security experiment	24
2.7	The Fiat-Shamir transform	24
2.8	Identity-based identification scheme ID-IMP-ATK-security exp's ($\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$)	26
2.9	Identity-based signature scheme ID-EUF-CMA-security experiment	27
3.1	Credential ownership proof COP-IMP-ATK-security experiments	35
3.2	Universal designated-verifier signature proof IM-TYPE- i security experiments	36
3.3	Hypothetical IM-TYPE-3 security experiments	37
3.4	Identity-based encryption scheme OWE-ID-CCA-security experiments	39
3.5	The Naor transform of an identity-based encryption scheme	40
3.6	The identity-based encryption scheme based credential ownership proof	40
3.7	The GQ identification scheme	44
3.8	The RSA-FDH signature scheme	45
4.1	Schemes constructed generically in this chapter based on signature S in \mathbb{C}	55
4.2	A canonical Σ protocol for proof of disjunctive knowledge	58
4.3	The non-interactive proof from applying Fiat-Shamir to the protocol in Figure 2.4	58
4.4	The signature scheme from applying Fiat-Shamir to the protocol in Figure 2.4	59
4.5	RSA, GQ and DL family key generation algorithms	60
4.6	The bifurcation in the h_i inputs of the forked algorithm	61
4.7	Reset Lemma experiments	62
4.8	Mechanism of security proofs for non-FL-Based and FL-Based signatures	65
4.9	Universal designated-verifier signature DV-EUF-CMA-security experiment	69
4.10	Universal designated-verifier signature PR-security experiments	70
4.11	Our generic construction of universal designated-verifier signatures	73
4.12	Mechanism of GUDVS DV-EUF-CMA-security proof	76
4.13	Our generic construction of identity-based signatures	85
5.1	The real model vs. the ideal model	115

5.2	Simulation of the ideal model in Figure 5.1	120
-----	---	-----

Chapter 1

Introduction

Credentials are inseparable parts of our everyday lives. Student cards, driving licenses, banknotes, bank checks, shopping vouchers, movie tickets, payment receipts, prescriptions, and more recently public-key certificates are all examples of cases where a document issued by an authority is used to prove a statement to a verifier. Traditionally, the issuer would sign and/or stamp the document to give evidence of the provenance of the document and their intention as per the content of the document. New technologies then changed the way credentials were issued, providing new anti-counterfeiting measures for issuers. Security holograms, magnetic stripes, embedded integrated circuits, RFID chips, smart cards, watermarks, and fluorescent dyes are among such measures. Then came the computers and revolutionized every aspect of human life, giving birth to a whole-new dimension: the *digital world*. The digital revolution has changed and continues to change the way credentials are used in everyday life. Computerized credential issuance and verification can be seen almost everywhere now. Furthermore, these days, an *electronic* version of almost any form of credential is either being introduced or gradually replacing the physical version.

Cryptographic signature schemes have been proposed to provide an electronic equivalent of handwritten signatures and more generally, an electronic means of issuing credentials. To sign a document, called here a *message*, first the contents of the message is fed into a *signing* algorithm along with the *secret key* of the signer. The output of this algorithm is then called a *signature* and is appended to the message itself. A verifier can subsequently verify that a signature is indeed produced by a specific signer on a particular message by inputting the message–signature pair into a *verification* algorithm along with the *public key* of the signer and calculating the boolean output of the algorithm. A cryptographic signature scheme must be *unforgeable*, that is, it must be hard to forge a signature that verifies against a public key without the knowledge of the corresponding secret key. Successful verification of a signature assures the verifier of the *authenticity* and *integrity* of the message that has been signed. That is, the verifier is assured that the signature is indeed produced by an entity holding the secret key corresponding to the public key he has input to the verification algorithm, and the message is indeed what the signer has signed and has not been tampered with. The verification algorithm and the public key of the signer are of public knowledge. Hence, message–signature pairs are *publicly verifiable*. An immediate consequence of such public verifiability is that signatures become *non-repudiable*, since in case of a dispute, a mediator can verify the validity of a purported signature.

The electronic equivalent for our everyday credentials in the digital world is a pair consisting of

the entitlement statement of the credential as the message and a signature on it generated using a cryptographic signature scheme. The holder of such an electronic credential just needs to provide this pair to the occasional verifier to prove to them the entitlement stated in the credential. However, the convenience of using digital credentials comes in a package with the ease of keeping tabs on virtually unlimited number of identities and the low cost of searching and discovering patterns in such databases. Thus, minimal disclosure of private data becomes a matter of extreme importance in the digital world. We cannot afford anymore to reveal more than needed of our personal information embedded in each and every credential of ours. In this thesis, we make contributions to secure and privacy-aware use of electronic credentials. We provide new formalizations, more efficient realizations, and/or generic constructions for some existing cryptographic schemes and protocols. We also introduce some new cryptographic schemes and protocols and provide proper definitions, formal security treatments, and concrete schemes accordingly.

Credentials can be classified into two categories based on whether or not the credential statement specifies the identity of the credential holder. If not, the credential entitles the bearer of it, regardless of their identity, to the credential statement. Examples of such credentials include shopping vouchers and movie tickets. Since they contain no stated identity, such credentials inherently do not disclose any credential holder private information and, as one would assume, are meant to be *transferable* at the bearer's will. However, the mere fact that the credential is provided to the verifier in whole at the time of verification, exposes it to the danger of being cloned by, and hence transferred to, the verifier. If the credential in question is a *one-show* credential, such as a movie ticket, transfer of the credential only enables the verifier to enjoy the benefits of possessing a "ticket stub". However, this problem, i.e., credential holder's lack of control over transfer of the credential, becomes more evident a threat when the credential in question is a *multi-show* one, such as a monthly ride permit. In such a case, the credential holder basically loses her entitlement as a result of transfer of an exact clone of her credential to the verifier. Hence, other ways of proving possession of a credential without showing it needs to be explored. Surprisingly, we find cryptographic literature in this field lacking proper definitional treatment and security formalization. We discuss how to properly formalize what we call *credential ownership proofs* to solve this problem. As a consequence, we are also able to provide more efficient concrete solutions to the above problem. Our formalizations of credential ownership proofs provide methods for demonstrating ownership of credentials without revealing them and guarantee that no credential is transferred against the credential holder's will. We discuss in detail the gap in the literature and our contribution in this regard in Section 1.1 and our contribution itself forms Chapter 3 of this thesis.

On the other side are the credentials in which the statement of the credential specifies, implicitly or explicitly, the identity of the entitled entity. Such credentials are inherently immune against the aforementioned threat of being unwillingly transferred to another party since the entitled entity is defined in the credential statement. However, other concerns rise as the credential is provided in whole to the verifier. Among these concerns is breach of credential holder's *privacy* since the credential might be used as a proof of sensitive information about the credential holder. The credential statement, for instance, might contain information about the credential holder that are personal in nature. Providing a verifier with such a credential in whole practically gives the verifier a publicly convincing proof of the personal information about the credential holder. No patient wants the pharmacy clerk possess a copy

of their prescription, signed by a doctor, denoting they are diagnosed with such and such disease. The so called *verifier designation* paradigm has been introduced in the cryptographic literature to provide a means to convert one’s credential to one that merely convinces a specific designated verifier, thus, effectively preventing the designated verifier to be able to prove holder’s possession of the credential to a third party. Full security requirements for the so called *universal designated verifier signatures* have recently undergone revision and a new security measure has been identified in the literature. Hence, most of the concrete schemes proposed to date either have shown to be insecure or at least lack a security proof with respect to the new measure. We develop an almost-generic method of transforming signature schemes into universal designated verifier signatures and prove that our transformed schemes fully satisfy the revised security requirements. Our method works for a very large class of schemes that contains all but one of the signatures we have identified in the literature. We show that universal designated verifier signatures developed using our method possess extra desired properties and are comparable to previous schemes in terms of efficiency and signature length. We also show that our method can be used to transform signature schemes to other useful cryptographic schemes as well. A more detailed introduction to our work in this regard comes in Section 1.2, while the body of the work can be found in Chapter 4.

Even if the credential statement does not reveal sensitive information about the credential holder, verifying that the credential is issued by a particular issuer might do so. In practice, the same type of credentials are issued by multiple issuing authorities and often the fact that one’s credential is issued by a certain authority reveals extra unnecessary information about the credential holder. For instance, proving possession of a credential issued by a *specific* local branch of an organization reveals the probable locality of the credential holder as well, while all the verifier needs to know might be user’s possession of a credential issued by *any* branch of the organization. We identify a new cryptographic scheme that we call an *attribute-based signature* in which issuers are identified by their attributes and signatures can be verified to be issued by an issuer with attributes satisfying a verification policy, without the exact attributes of the issuer being revealed. We show that such schemes can be used to prevent disclosure of unnecessary information in time of credential verification. We formalize such schemes and provide proper definitions of their security. We also provide concrete schemes with security proofs. Section 1.3 introduces our contributions in this respect in a more detailed fashion, while the full technical details can be found in Chapter 5.

Anonymous credential systems have been introduced in cryptography as a means of protecting users’ privacy through maintaining their anonymity. Users in such systems are known to organizations that issue or verify credentials by their multiple *pseudonyms*, hence they remain anonymous to organizations and their transactions with (multiple or even the same) organizations remain unlinkable. Anonymous credential systems have been shown in the literature to be realizable using special signatures, that we call *C-signatures*, as building blocks. We introduce *attribute-based C-signatures* as new schemes in line with the above. We show that such signatures can be used to realize attribute-based anonymous credential systems, a privacy-enhanced version of the anonymous credential systems. We consider this a crucial privacy enhancement since revelation of unnecessary information about users through disclosure of the exact identity of their credential issuers undermines the anonymity and unlinkability of the anonymous credential systems and contradicts the purpose of such systems. We formalize attribute-based C-signatures and their security notions and give a concrete scheme satisfying our

definitions. We provide appropriate formalization of attribute-based anonymous credential systems and their security and show how a concrete system can be realized using a secure attribute-based C-signature. A more elaborate introduction to attribute-based C-signatures and their application in enhancing the privacy of anonymous credential systems is given in Section 1.4 and the formal treatment can be found in Chapter 5.

1.1 Credential Ownership Proofs

Consider the following scenario. An issuer, for instance a sports club, wishes to put in place an electronic ticketing system that supports issuance of both one-show and multi-show tickets. The tickets should not limit the entitled user to a particular identity and hence are meant to be transferable at ticket holder's will. That is, holders should have the flexibility to give their tickets to others. A basic credential system based on digital signatures can closely satisfy these requirements. The electronic ticket will have a statement m that declares the entitlement of the holder, and the signature σ of the issuer on m . When the ticket is presented to a ticket controller, the signature of the issuer is verified, and if valid, the statement m will be honored. Of course, since the electronic ticket, as any piece of digital information, is easily cloned, it is essential for the statement of the credential to include a unique identifier, such as a serial number. Checking this unique identifier against a log of those of already-shown tickets prevents 'double spending' of tickets and ensures that the 'one user per ticket' condition is enforced.

The system looks very attractive: it appears *secure* assuming the signature system provides unforgeability; it is *efficient* and requires one signature generation and verification for ticket generation and checking, respectively; and uses *standard cryptography*, thus, is easy to implement. However, the system does not guarantee holders' control over their credential transfer. The ticket can be illegally copied (cloned) by ticket controllers during ticket verification phase, since in this phase, the credential is wholly revealed by the user to be verified. A malicious controller makes a copy of a ticket during a showing by an honest user, and then successfully uses it to acquire future service from the system, simply by making sure that they are the first to present the ticket to the system. Even if the ticket is not valid for future service, the copy of the credential counts as a "ticket stub" and can be used by the malicious controller to claim that they have purchased the ticket. In both cases, the user will effectively lose the privilege that the ticket needs to guarantee.

An immediate solution to this problem is to employ interactive *zero knowledge proofs* [GMR89] at the time of verification. Interactive proofs enable ticket holders to prove possession of their tickets in an interactive fashion without revealing their credentials in whole to the controllers. The zero knowledge property guarantees that the verifier does not learn any information other than the validity of the signature. However, using zero knowledge verification is an overkill for the problem, since the zero knowledge security guarantee is much stronger than what one expects from the above system and zero knowledge proofs are typically computationally expensive and require communication of large amounts of data. Further inspection of the literature in this area yields a number of other proposals, such as *deniable message authentication schemes* [DDN00] and *non-transitive signatures* [Des88, OO90], that are all formalized in a way that require one or multiple zero knowledge protocols when implemented.

This shows that proper formalization permitting more efficient implementations is lacking in this area.

We propose and formalize the notion of *credential ownership proofs* and their security. A credential ownership proof for a signature scheme is an interactive protocol that enables a credential holder to prove possession of her credential to a verifier. During the protocol, the credential holder reveals the claimed entitlement of her credential, but does not reveal the issuer’s signature on the entitlement. A secure credential ownership proof guarantees that the ability to prove possession of the credential is not transferred to the verifier. That is, prover’s interaction with multiple verifiers should not allow misuse of the system by enabling a successful run of the protocol without having the required credentials. This guarantee ensures credential holder’s control over credential transfer.

To formalize credential ownership proof security, we consider three types of attacks. In all types, the goal of the adversary is to prove ownership of a credential it does not possess to an honest verifier. The adversary also can ask for credentials of its choice to be revealed to it during all three types of attack, but it is assumed to possess the revealed credentials afterwards. This models the credentials that are either intentionally or unintentionally revealed to untrusted entities by the credential holder. In a *passive* attack, the adversary gets to eavesdrop credential ownership proof communications of credential holders of its choice. Security against this type of attacks guarantees that outsiders to the system are not able to “steal” credentials by tapping the communication lines. This level of security is adequate for closed systems in which the credential verifiers are trusted. In an *active* attack, the adversary gets not only to eavesdrop communications, but also to communicate as a malicious verifier with credential holders of its choice one after another. Security against this type of attacks guarantees that both outsiders and malicious credential verifiers are not able to steal credentials if credentials are verified one at a time. This level of security provides a stronger guarantee than security against passive attacks and is proper for open systems in which the verifiers are not assumed to be trusted and no more than one credentials is verified at a time. In a *concurrent* attack, the adversary gets not only to eavesdrop communications, but also to communicate as a malicious verifier with many credential holders of its choice simultaneously. Security against this type of attacks guarantees that both outsiders and malicious credential verifiers are not able to steal credentials even if a malicious verifier or a coalition of malicious verifiers are able to interleave communications with different credential holders arbitrarily. This level of security is provides a stronger guarantee than security against passive and active attacks and is proper for open systems in which the verifiers are not trusted and credentials are verified in arbitrary time. Based on the properties of each system, a credential ownership proof satisfying a particular level of security might be used.

Next, we consider construction of credential ownership proofs. We provide two generic constructions for credential ownership proofs based on other cryptographic schemes and prove our constructions secure given that the underlying cryptographic scheme is secure. The first construction is based on *identity-based encryption schemes*. We show how to construct a secure credential ownership proof given a secure identity-based encryption scheme. The level of security we require from the underlying identity-based encryption scheme is actually weaker than the accepted level of security for identity-based encryption schemes. This construction provides realization of credential ownership proofs based on a range of different system parameters and security assumptions. Besides, it is of theoretical interest as it shows that secure credential ownership proofs are realizable in the standard cryptographic model without ideal assumptions such as existence of random oracles.

The second generic construction is based on *identity-based identification schemes*. We actually show an equivalence between the following two: on one hand, a pair consisting of an unforgeable signature scheme and an associated secure credential ownership proof, and on the other hand, a secure identity-based identification scheme. That is, we show that a secure construction for each can be realized given a secure construction for the other. We show a one-to-one relationship between entities, algorithms, and parameters in the two and give a bilateral translation of security notions between them.

Both generic constructions above use identity-based cryptography, which could be considered as less traditional and requiring more advanced knowledge of cryptography. It is desirable to have a credential ownership proof that is secure and uses “textbook” cryptography. We provide a concrete credential ownership proof based on the RSA system of Rivest, Shamir, and Adleman [RSA78] and prove the scheme secure under concurrent attacks. This scheme is very efficient and can be easily implemented using commonly used cryptographic libraries.

1.2 Universal Designated Verifier Signatures

Credential holder identity is embedded in the entitlement of a credential to limit the entitlement to a specific identity. Such credentials are intrinsically non-transferable. Thus, one can be sure that an untrusted verifier is not able to “steal” one’s entitlement as per the credential. However, concerns for user *privacy* arise as soon as credential holder specific information is integrated in credentials. A major concern in this regard is the lack of any guarantee for credential holders that *they* are the only entities who can “prove” the credential entitlement. That is, an untrusted verifier may be able to prove credential holder sensitive information to third parties after interacting with the credential holder. This phenomenon that might breach user privacy stems from the fact that digital signatures are publicly verifiable and hence non-repudiable.

To solve this problem, Steinfeld et al. have proposed *universal designated-verifier signatures* [SBWP03]. In a universal designated-verifier signature scheme, one can transform a signature to a *designated signature* that only convinces a particular designated verifier. This transform does not need the knowledge of any secret key and can be carried out using the public key of the designated verifier, hence the adjective “universal”. The transform is called (*verifier*) *designation*. The credential holder first designates the signature part of her credential using the verifier’s public key and gets a designated signature. Then she sends the designated credential, consisting of the entitlement of the credential and the designated signature on it, to the verifier. She is guaranteed that this designated credential only convinces the intended designated verifier and the verifier is not able to “prove” her credential entitlement details to a third party.

Since their introduction, there have been a number of proposals for universal designated-verifier signature schemes. All these proposals assume a particular combination of cryptographic system settings on the signer side on one hand and the verifier side on the other. That is, each proposal works only if a specific signature scheme is used by the credential issuer to issue credentials and a specific family of key pairs is used by all the possible designated verifiers. This fact is a limiting factor for widespread usage of universal designated-verifier signature schemes, since in practice it is virtually impossible (and generally, in contradiction with freedom of choice) to force all entities to use the same cryptographic

system settings. Even if an agreement on a uniform system usage is reached by all entities in the system, transition from the currently deployed systems to the new system is costly. On the other hand, security definitions of universal designated-verifier signature schemes have been revisited in the literature since their introduction and new security measures have been identified. Namely, Lipmaa et al. argued that the original security definitions of signatures with added verifier designation do not sufficiently capture the intuitively expected requirements and introduced a new security notion called *non-delegatability* [LWB05]. They showed that in some universal designated-verifier signature schemes, the credential issuer is able to delegate his signing ability with respect to a fixed designated verifier to a third party, without revealing his secret key or even enabling the third party to sign with respect to other designated verifiers. They argue that, in many scenarios, such delegation property is undesirable and must be prevented. None of the many universal designated-verifier signature schemes proposed to date, except one recent scheme, has treated non-delegatability as a security requirement. Furthermore, many of the proposed universal designated-verifier signature schemes are shown to be delegatable, including the original scheme by Steinfeld et al. The two facts mentioned above, namely the nonexistence of setting-flexible universal designated-verifier signature schemes and the lack of treatment of non-delegatability as a security requirement for universal designated-verifier signature schemes, point out a gap in the literature. In this thesis, we fill this gap by designing a suite of setting flexible and provably non-delegatable universal designated-verifier signature schemes.

To design provably non-delegatable universal designated-verifier signature schemes, we first need a proper definition of non-delegatability for such schemes. A formal treatment of such a definition does not exist in the literature. We propose a formal definition for non-delegatability of universal designated-verifier signature schemes by adopting some existing ideas in the literature and tailoring them to suit our specific security guarantee. A non-delegatable universal designated verifier signature scheme guarantees that only entities with “knowledge” of specific secret keys can generate designated signatures and hence the ability to generate designated signatures cannot be delegated without revealing at least one of the secret keys in question. Our definition makes use of an accepted notion of “knowledge” in cryptographic literature as the ability to efficiently compute a quantity. Hence, we define non-delegatability as the guarantee that if an entity is able to produce designated signatures it must “know” at least one of the secret keys in question, i.e., it must be able to efficiently compute at least one of them.

We define a large class \mathbb{C} of signature schemes and a large class \mathbb{K} of families of key pairs. Class \mathbb{C} is defined based on certain requirements on the signature scheme and we show that it contains all but one of the major signature schemes that we surface from the literature. Class \mathbb{K} is defined based on a requirement on the key pair used by a possible verifier in a universal designated verifier signature scheme and we show that all the major families of key pairs used in cryptographic systems today belong to this class, among them the two major families of key pairs: the RSA key pairs and the discrete logarithm based key pairs.

We give a generic construction of universal designated-verifier signature schemes from the combination of any signature scheme in class \mathbb{C} and any family of key pairs used by the verifier in class \mathbb{K} . We prove that for an arbitrary combination as above, our construction of the universal designated verifier signature scheme is both secure in the two original senses, i.e., unforgeability and non-transferability privacy, and non-delegatable in the sense we define. Since our construction works for any combination

of settings for signature schemes and verifier keys, it can be used in an “on the fly” basis to designate signatures. That is, the credential issuer and possible designated verifiers are free to choose any arbitrary signature scheme and key family, respectively, and yet the signature holder is able to construct a designated credential based on our construction and the issuer and verifier settings. This leaves the possible credential issuers and possible credential verifiers free to choose their own cryptographic settings independently. Our construction also provides the existing credential systems with ready-to-use verifier designation and designated signature verification methods that do not require the existing systems to be reset or updated. Our construction of universal designated verifier signature scheme is hence a non-delegatable setting-flexible construction that can be adopted easily in the majority of the established systems with different cryptographic settings.

In terms of efficiency, we compare instances of our construction with their counterparts proposed in the literature with the same signature scheme and same family of verifier key pair. Surprisingly, we find out that, although our construction is generic and our instances provide the extra provable non-delegatability property over their counterparts, our schemes have comparable and sometimes even better performance in terms of efficiency. Our comparison also reveals an extra property provided by instances of our construction. We find that considerable proportions of calculations required for designation of a credential in different instances of our construction can be carried out off-line. That is, a credential holder is able to pre-compute the computationally costly parts of designating a signature of hers even before she decides on the designated verifier. Furthermore, our schemes provide computationally inexpensive on-line designation calculations, as cheap as only one modular multiplication, in contrast with designation costs of at least one modular exponentiation for their counterparts.

In practice, it is often beneficial to be able to designate a credential to *multiple* verifiers at once, since credential holders usually need to show each of their credentials to more than one verifier. A *universal multi-designated-verifier signature scheme* is a generalization of universal designated-verifier signature scheme in which the credential holder can designate her credential to a set of designated verifiers at once. The resulting designated credential only convinces the designated verifiers. An interesting property of our generic construction is that it can be extended to universal multi-designated-verifier signature schemes. Our generic construction of universal multi-designated-verifier signature schemes permits independent choices of the signature scheme and verifier keys. That is, the signers and verifiers in the system are free to independently choose arbitrary signatures and key settings, respectively, and yet one is able to designate signatures to an arbitrarily chosen set of verifiers on the fly.

1.2.1 Further Identity-Based Constructions

Verification of a credential against a public key ensures the verifier that the credential is issued by an entity in possession of the corresponding secret key. Hence, a successful verification merely links a credential with a public key. To successfully link a credential to a specific issuer, one needs to somehow make sure that the public key used in verification does in fact belong to the issuer in question. *Public key certificates* are used to deliver such guarantees. A public key certificate is in fact a credential issued by an authority for which a trusted public key is known. The entitlement of a public key certificate includes the identity and the corresponding public key of an issuer. This entitlement is

signed by the authority. By successfully verifying such a credential, one can obtain a guarantee that a purported public key does indeed belong to an issuer. Hence, in practice, claiming a credential entitlement involves the user presenting the credential along with a public key certificate of the issuer to the intended verifier.

Identity-based cryptography is proposed to eliminate the need for public key certificates [Sha84]. The idea is to use the identity of the issuer as his public key. Each entity receives a secret key corresponding to their identity from a *key generation center* after their identity being proved to the center. This secret key can be used to issue credentials and verifiers are able to verify the issued credentials directly against issuer identities.

We show that our definition of the class \mathbb{C} enables us to generically construct *identity-based signatures* from any signature in the class. This generic construction provides a framework that explains a large portion of the identity-based signature schemes in the literature. Furthermore, it enables construction of new schemes based on different security assumptions. A desirable property of our techniques used in our generic constructions of universal designated-verifier signature schemes and identity-based signature schemes is that they can be combined together to give us a generic construction of *identity-based universal designated-verifier signatures*. Such signatures serve as important tools providing user privacy in the identity-based setting.

1.3 Attribute-Based Signatures

A credential is an authority's certification of one or more features of the intended credential holder. Consequently, a credential usually contains data that precisely identifies the issuing authority and the credential holder and determines the statement that is certified by the described authority about the described credential holder. Often credentials with similar or even the same certified statements are issued by different "branches" of an authority or even by different authorities. In the former case, the distribution of the authority to issue credentials among the "branches" is administered in order to possibly divide the computational load among the different branches and/or to accommodate for customer needs. For instance, local branches of an administration might all issue standard membership cards for customers. In the latter case, such distribution among independent authorities is possibly because each potential credential holder is eligible to acquire a credential from a specific authority. For instance, different companies might issue similar proofs of work experience, relevant for their employees. Although the credential statement, in this sense, needs to be wholly revealed at the time of verification, preserving privacy of the credential holder requires that bits of information about the identity of both the issuer and the credential holder are *selectively* disclosed. Now, selective disclosure of credential holder personal information is an issue that has drawn a fair bit of attention already, but the problem here that has not been adequately dealt with is that careless disclosure of bits of information about the issuer's identity can do as much harm to credential holder privacy. Proving possession of a credential from a specific "branch" in the above example might undesirably reveal the probable "locality" of the credential holder. In the same manner, disclosing the exact company one has worked for might be unnecessary in many situations where only a proof of experience in the field suffices.

Although a considerable amount of work exists on selective disclosure of credential holder personal information, we find the research on selective disclosure of issuer identity information inadequate. To formalize this concept, we view the identity of the issuers in our credential system as a combination of *attributes*, rather than a single string defining the identity. We formalize the notion of *attribute-based signatures* as a signature scheme in which signers are described by attributes and signatures can be verified to have been produced by a signer with attributes satisfying a *verification policy*. In an attribute-based signature scheme, each signer is assumed to possess a set of attributes that we call the *signer attribute set*. A central authority issues secret keys to the signers in the system based on their signer attribute set. At the time of signing, the signer chooses a subset of his signer attribute set, called a *signing attribute set*, and signs as an entity with such attributes by inputting his secret key, the chosen signing attribute set, and the message to be signed to the signing algorithm and generating a signature. The signing attribute set, the message, and the generated signature compose the credential which is given to the credential holder. At the time of verification, the verifier chooses a *verification policy* and feeds the message, the signature, and the verification policy to the verification algorithm that subsequently produces a boolean output. This output indicates whether or not the signing attribute set, chosen by the signer, satisfies the verification policy.

We focus on *threshold* verification policies. A threshold verification policy can be determined by a *verification attribute set* and a threshold, and a set of attributes is said to satisfy the policy if the number of attributes it has in common with the verification attribute set is greater than or equal to the threshold. We call attribute-based signatures that support threshold verification policies, *threshold attribute-based signatures*. We formalize the security of threshold attribute-based signature schemes and require that they provide unforgeability, as one would expect from any signature scheme. Unforgeability guarantees that signatures cannot be forged without the knowledge of proper secret keys. Furthermore, here we need to prevent another type of attack which involves a coalition of signers working with each other to “aggregate their attributes” and produce signatures that seem to be coming from a signer with a mixture of their attributes. We formalize the notion of *collusion resistance* as a countermeasure for this attack. collusion resistance guarantees that no group of colluding signers are able to come up with a signature that none of them could generate alone.

Our threshold attribute-based signature schemes enable a credential holder to selectively reveal their credential signer’s attributes. That is, the credential holder can choose which attributes of their credential issuer is visible to the verifier at the time of verification. This is done by designing our schemes to produce signatures that include components corresponding to each attribute in the signing attribute set. The credential holder then simply chooses the signature components corresponding to the attributes of the issuer they want to reveal and as long as there are at least a threshold number of these components, the verification succeeds.

Furthermore, we show how our threshold attribute-based signature can provide *signer-attribute privacy*. We then define threshold attribute-based signature schemes with an additional protocol to interactively prove possession of a credential valid against a threshold policy, using similar ideas as those of credential ownership proofs. This protocol enables a credential holder to prove possession of a credential satisfying the verification policy without revealing it. Furthermore, we require that as a result of such a proof, the verifier does not get to know anything further than the validity of the holder’s credential. That is, the only information the verifier gets is that the credential issuer

attributes satisfy the threshold verification policy. In particular, the verifier does not find out which (at least) threshold-sized subset of the verification attribute set correspond to the attributes of the credential issuer. We call this property signer-attribute privacy and believe that it provides a crucial privacy enhancement to credential systems.

Our threshold attribute-based signature schemes enable a functionality that can be seen either as *threshold attribute-based verification* or *fuzzy verification*, depending on the application at hand. Threshold attribute-based verification of signatures has applications in situations where credentials are verified against whether or not a threshold of attributes are possessed by the issuer. Fuzzy verification of signatures is useful in situations where an approximate identity of the signer is available to the verifier and the verification must go through if the approximation is close enough. This functionality is useful particularly in *biometric signatures*. Such signatures can be implemented in different ways, one of which is as identity-based signature using each signer’s biometric features as their identity. Using biometric as users’ identities provides an attractive way of providing user authentication and ensuring that the identity is not forged. To implement such schemes, biometric features are used as each user’s identity and secret keys are issued to users by the key generating authority after measurement of their biometrics at the time of registration. Each entity can either have their own measurement of the user’s biometrics as the user’s identity or trust another entity’s measurement to obtain a user identity. Thus, a verifier’s measurements of the user’s biometrics is not exactly the same as the one collected at the registration phase. Thus, “fuzzy” matching between the two biometric measurements is required. With suitable choices of threshold and mapping, “close” biometric measurements map to attribute sets for signing and verifying that have sufficient overlap and result in successful verification of the signature.

Our choice to work in the recently proposed area of attribute-based cryptography enables us to borrow some design ideas from the recent works on attribute based encryption. We propose a threshold attribute-based signature scheme and prove it secure against forgery and collusion attacks in the standard cryptographic model. Our scheme admits to efficient protocols for proof of possession of a credential. We prove that, using such proofs, our schemes provides signer-attribute privacy.

1.4 Attribute-Based Anonymous Credential Systems

Among the major steps toward realizing a privacy-enhanced digital world is the introduction and design of *anonymous credential systems*. A credential system, in this regard, is a system composed of two types of entities: those who issue and/or verify credentials, or simply the “organizations”, and those to whom credentials are issued and whose credentials are verified, or simply the “users”. The definition of the system determines concrete procedures for credential issuance and verification. In an anonymous credential system, users are known to the organizations merely by their *pseudonyms*. That is, each user can choose pseudonyms, independent of her real identity, to use as her identity in her correspondence with organizations. The user can choose to use the same or different pseudonyms to be known as in different transactions. An anonymous credential system enables the user to get issued a credential on one pseudonym and prove possession of the same credential on a different pseudonym as long as she owns both pseudonyms. Since the only information about a user identity

that an organization finds out from its transactions with her is the user pseudonym(s), the user remains *anonymous*. Furthermore, the system guarantees a high level of desirable properties including *unlinkability* and *non-transferability*. Unlinkability guarantees that different transactions (involving different pseudonyms) of the same user remain unlinkable to each other, even if the transactions are carried out with the same organization. Non-transferability ensures that a user can neither get issued credentials on behalf of another user nor transfer her credential to another user. This is achieved by requiring the user to prove possession of the involved pseudonyms at all transactions, whether it be issuance or verification of a credential.

Commitment schemes were proposed in cryptography to mimic the physical world's putting a note in an envelope for some time and opening it later. While the note inside is hidden, it cannot be changed. A cryptographic commitment scheme works as follows. One inputs a secret quantity and a random value to the *committing* algorithm and gets a commitment. A commitment is in a way that *hides* the secret value, that is, by seeing a commitment, one cannot infer any information about the committed secret value. Whenever the committer decides to reveal the secret, the committed value along with the random value used at the time of committing is provided to a verifier. The verifier can run the committing algorithm and check whether or not the output is the same as the commitment. The commitment scheme should *bind* the committer to the committed value, that is, it should be impossible for the committer to change his committed value after committing to it.

It has been shown that anonymous credential systems can be realized using a special kind of signature schemes that we call *c-signatures*. C-signatures consist of a commitment scheme, a signature scheme, and protocols for signing committed values and verification of signatures on committed values. The protocol for signing a committed value is an interactive protocol between a user with a secret value and a signer with a secret key. The user knows the public key of the signer and the signer knows a commitment to the user's secret value. The protocol enables the user to obtain a signature on her secret value. The protocol guarantees that the user is not able to cheat, that is, the user is not able to obtain a signature on any message different from her secret committed value. Furthermore, the protocol guarantees that the signer does not find out any information about the message he is signing, i.e. the user's secret value. The protocol for verification of signatures on committed values is an interactive protocol between a user with a secret value and a signature on it and a verifier that knows a commitment on the user's secret value and the public key of the signer. The protocol enables the verifier to verify the validity of user's held signature with respect to the public key of the signer without learning anything about the user's secret value or the signature she possesses. The protocol also guarantees that only users with valid signatures can convince verifiers. To realize an anonymous credential system using a c-signature, users use their real identity as their secret value and different commitments to it as their pseudonyms and signatures serve as credentials. It has been proved that if for a c-signature: the commitment scheme is hiding and binding, the signature scheme is unforgeable, and the additional protocols are secure in the above senses, then the realized anonymous credential system is anonymous, unlinkable, and non-transferable.

We propose *attribute-base c-signatures* as a building block for anonymous credential systems that provide signer-attribute privacy besides the discussed security properties. Analogously, an attribute-base c-signature consists of a commitment scheme, an attribute-based signature scheme, and protocols for

attribute-based signing of committed values and attribute-based verification of signatures on committed values. We show that our concrete threshold attribute-based signature scheme can be extended to a threshold attribute-based c-signature. We provide the relevant commitment scheme and design efficient additional protocols for our threshold attribute-based signature scheme and prove that they provide the security requirements of a threshold attribute-based c-signature.

Our threshold attribute-based c-signatures enable us to realize *threshold attribute-based anonymous credential systems*. We first define such systems and the security properties of them. Then we proceed to show that using a threshold attribute-based c-signature scheme, such systems can be realized and prove that the system provides all the security requirements if the underlying threshold attribute-based c-signature is secure. Our proposed threshold attribute-based anonymous credential systems provide signer-attribute privacy in addition to anonymity, unlinkability, and non-transferability. We view this property as a crucial privacy enhancement to anonymous credential systems, the lack of which might undermine other properties such as unlinkability and even anonymity. Without signer-attribute privacy unnecessary information about the users in the system is exposed. Such information might enable an organization or a group of colluding organizations to link transactions with each other and break the unlinkability property. If the amount of unnecessary information about users in the system exceeds a reasonable threshold, then even the anonymity of users in the system might be lost. By providing selective disclosure of signer attributes and furthermore signer-attribute privacy, our threshold attribute-based anonymous credential system is a step forward to eliminate or at least lower the probability of such risks.

Chapter 2

Preliminaries

2.1 Public Key Cryptography and Provable Security

Throughout this thesis, we assume a *public key cryptography* (also called *asymmetric cryptography*) setting. The concept of public key cryptography was introduced by Diffie and Hellman in their seminal work: *New Directions in Cryptography* [DH76]. The idea is that each entity, Alice, in the system owns a pair of keys: a *public key* that she publishes and a *secret key* (also called a *private key*) that she keeps secret to herself. The public key is used by others in the system to calculate Alice-specific public functions, such as Alice-intended encryption and Alice-alleged signature verification. Alice, on the other hand, uses her secret key to calculate her secret functions, such as Alice-intended ciphertext decryption and Alice signature generation. Assuming that Alice keeps her secret key private, others in the system are assured that if secure schemes are in place, nobody other than Alice can calculate these Alice-specific secret functions.

It is not possible for every user in the system to retrieve Alice’s public key directly from herself or a fully trusted authority through a fully secure communication channel. Hence, there must be some measure in place to guarantee that an alleged public key retrieved otherwise does indeed belong to Alice. Such measure is usually implemented through use of *public key certificates*. A public key certificate is a document containing a statement indicating that a specific public key belongs to a particular identity and a *certificate authority* (CA)’s signature on this statement. One needs to obtain the certificate authority’s public key to verify the validity of the signature in a public key certificate. In turn, the CA’s public key is either fully trusted or verified through a public key certificate of its own. The chain of certification continues till a CA with trusted public key is found. In practice, a tree-like certification infrastructure should be in place, in which a small number of *root CAs* exist and the CAs in each level certify public keys for a larger number of CAs and entities in a lower level. This way, the trust in the public keys of a few number of high-level CAs propagates to the public keys of lower level and finally leaf-level entities. Such an infrastructure is called a *public key infrastructure* or PKI for short.

Cryptographic schemes are designed to serve the “good guys”, also known as the *honest users*, and withstand the attacks by the “bad guys”, also known as the *adversary*, collectively. Hence, no cryptographic scheme definition is complete without a definition of the security of the scheme. There are generally two approaches to define the security of a cryptographic scheme: the *game-based approach*

and the *simulation paradigm*.

The game-based approach has been used as a natural way of defining security by cryptographers as early as Goldwasser and Micali's definition of semantic security for encryption schemes [GM82]. In this approach, security of a scheme is defined through a game between a *challenger* and an adversary. The challenger sets up the required parameters and keys for the scheme and occasionally provides the adversary with the quantities it is allowed to know during the attack. In other words, the challenger is there to create the environment in which the adversary is supposed to carry out its attack. At the end of the game, the adversary outputs its calculated quantities, based on which a predicate indicating the adversary's success or failure in the attack is calculated. A scheme is said to be computationally (resp. information-theoretically) secure based on such a definition if the probability of any computationally-bounded (resp. -unbounded) adversary winning the game is negligible. This probability is calculated over the random selections the challenger and the adversary make during the game.

The simulation paradigm originates from multi-party computation literature such as [GMW87] and DBLP:journals/jacm/GoldreichMW91. The intuition behind this paradigm is that a real system can be dubbed secure if it “emulates” a certain ideal system designed to trivially guarantee the expected security properties. To formally define security using this paradigm, one first defines a protocol-independent *framework* that contains the model of computation and communication and a notion of *emulation*. The former determines the types of entities involved in the real-life and the ideal protocol execution, what each of them can see and do, and how they communicate. The latter defines what it means to say a real protocol execution emulates (i.e., is as secure as) the ideal protocol execution. Then, an *ideal model* specific to the protocol is defined to capture the expected security properties. Security in the ideal model is guaranteed by including a trusted entity that collects the inputs of different entities in each round, performs the necessary calculations locally, and hands each entity's outputs for that round back to them. An actual cryptographic scheme is then called secure if one can prove that whatever a feasible adversary can obtain in its attack on the actual scheme is also feasibly attainable in the ideal model. For a detailed discussion on this paradigm please refer to [Gol04].

A cryptographic scheme is said to have “provable security”, as opposed to heuristic security, if there is formally shown a reduction from the security of the scheme to the hardness of a number of computational problems that are believed to be hard to solve. Security reductions must be carried out in a well-defined model of computation. A fairly popular model of computation is the so-called *random oracle model*. In the Random Oracle Model, all the algorithms in the system are assumed to have access to a truly random function, the random oracle. On the other hand, if no such access or the like is assumed, we are said to be working in the so-called *plain model*, also known as the *standard model*. The random oracle model was first used in cryptography by Fiat and Shamir [FS86] and later rigorously formalized by Bellare and Rogaway [BR93]. In practice, random oracles are replaced by cryptographic hash functions. Although design and analysis of cryptographic schemes are easier in the Random Oracle Model, since hash functions do *not* implement random (or even pseudo-random) functions, standard-model schemes are generally preferred to their random oracle model counterparts with comparable computational cost.

2.2 Computational Assumptions

Provable security of the cryptographic schemes rely on computational assumptions. We review a number of these assumptions below.

Let Gen_{RSA} denote a *prime-exponent RSA key generator* that on input a security parameter k generates a triple (N, e, d) where N is the product of two distinct primes, $2^{k-1} \leq N < 2^k$, $e < \varphi(N)$ is an odd prime, $\gcd(d, \varphi(N)) = 1$, and $ed = 1 \bmod \varphi(N)$. Associated with Gen_{RSA} , we define the following.

The *RSA (inversion) assumption* is the underlying assumption in the RSA cryptosystems [RSA78]. Intuitively, it says that it is hard for an adversary to invert the exponentiation function modulo an RSA modulus on random challenge. Formally, it is defined as follows.

Definition 2.2.1 (RSA Assumption) For Gen_{RSA} and an adversary A , we define the advantage of A in solving the RSA problem as the probability below, where the experiment $\text{Expt}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA}}(k)$ is defined as in Figure 2.1.

$$\text{Adv}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA}}(k) \triangleq \Pr [\text{Expt}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA}}(k) = 1]$$

We say that the RSA assumption holds if the maximum advantage a polynomial-time adversary can get in solving the RSA problem is negligible in k .

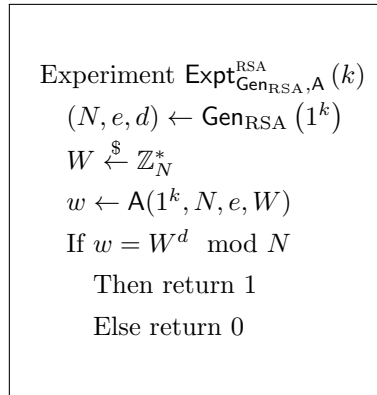


Figure 2.1: The experiment used to define the RSA assumption.

The *one more RSA inversion assumption* was first proposed in [BNPS01]. Intuitively, it says that it is hard for an adversary to invert the exponentiation function modulo an RSA modulus on a number of random challenges even if it can obtain the inverse for some of them. Formally, it is defined as follows.

Definition 2.2.2 (One More RSA Inversion Assumption) For Gen_{RSA} and an adversary A , we define the advantage of A in solving the one more RSA inversion problem as the probability below, where the experiment $\text{Expt}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA-OMI}}(k)$ and the relevant oracles are defined as in Figure 2.2.

$$\text{Adv}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA-OMI}}(k) \triangleq \Pr [\text{Expt}_{\text{Gen}_{\text{RSA}}, A}^{\text{RSA-OMI}}(k) = 1]$$

We say that the one more RSA inversion assumption holds if the maximum advantage a polynomial-time adversary can get in solving the one more RSA inversion problem is negligible in k .

<p>Experiment $\text{Expt}_{\text{Gen}_{\text{RSA}}, \mathbf{A}}^{\text{RSA-OMI}}(k)$</p> <p>$q, n \leftarrow 0$</p> <p>$(N, e, d) \leftarrow \text{Gen}_{\text{RSA}}(1^k)$</p> <p>$(w_1, w_2, \dots, w_n) \leftarrow \mathbf{A}(1^k, N, e : \text{Inv}(\cdot), \text{Chal})$</p> <p>If $w_i = W_i^d \bmod N$ for all $i = 1..n$ and $q < n$</p> <p> Then return 1</p> <p> Else return 0</p>	<p>Oracle $\text{Inv}(Y)$</p> <p>$q \leftarrow q + 1$</p> <p>Return $Y^d \bmod N$</p> <hr/> <p>Oracle Chal</p> <p>$n \leftarrow n + 1$</p> <p>$W_n \xleftarrow{\\$} \mathbb{Z}_N^*$</p> <p>Return W_n</p>
--	---

Figure 2.2: The experiment used to define one more RSA inversion assumption and the relevant oracle. q denotes the number of queries made by the adversary to the RSA inversion oracle Inv . q denotes the number of challenges returned to the adversary by the challenge oracle Chal .

The one more RSA inversion assumption is a stronger assumption than the RSA assumption, since if an adversary succeeds in solving the RSA problem, then it can solve the one more RSA inversion problem as well.

Let Gen_{DL} denote a *discrete logarithm parameter generator* that on input a security parameter k generates a pair (q, g) where q is a prime such that $q \mid p - 1$ for a prime p with $2^{k-1} \leq p < 2^k$ and g is a generator of G_q , a subgroup of \mathbb{Z}_p^* of order q . Associated with Gen_{DL} , we define the following.

The *Discrete Logarithm (DL) assumption* intuitively says that it is hard for an adversary to compute discrete logarithms in a cyclic group. Formally, it is defined as follows.

Definition 2.2.3 (Discrete Logarithm Assumption) For Gen_{DL} and an adversary \mathbf{A} , we define the advantage of \mathbf{A} in solving the discrete logarithm problem as the probability below, where the experiment $\text{Expt}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{DL}}(k)$ is defined as in Figure 2.3 (left).

$$\text{Adv}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{DL}}(k) \triangleq \Pr [\text{Expt}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{DL}}(k) = 1]$$

We say that the discrete logarithm assumption holds if the maximum advantage a polynomial-time adversary can get in solving the discrete logarithm problem is negligible in k .

The *Computational Diffie-Hellman (CDH) assumption* is the underlying assumption in the Diffie-Hellman key agreement scheme [DH76]. Intuitively, it says that it is hard for an adversary to compute g^{ab} given only g^a and g^b in a cyclic group. Formally, it is defined as follows.

Definition 2.2.4 (CDH Assumption) For Gen_{DL} and an adversary \mathbf{A} , we define the advantage of \mathbf{A} in solving the CDH problem as the probability below, where the experiment $\text{Expt}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{CDH}}(k)$ is defined as in Figure 2.3 (right).

$$\text{Adv}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{CDH}}(k) \triangleq \Pr [\text{Expt}_{\text{Gen}_{\text{DL}}, \mathbf{A}}^{\text{CDH}}(k) = 1]$$

We say that the CDH assumption holds if the maximum advantage a polynomial-time adversary can get in solving the CDH problem is negligible in k .

Experiment $\text{Expt}_{\text{Gen}_{\text{DL}}, \text{A}}^{\text{DL}}(k)$ $(q, g) \leftarrow \text{Gen}_{\text{DL}}(1^k)$ $W \xleftarrow{\$} G_q$ $w \leftarrow \text{A}(1^k, q, g, W)$ If $W = g^w$ Then return 1 Else return 0	Experiment $\text{Expt}_{\text{Gen}_{\text{DL}}, \text{A}}^{\text{CDH}}(k)$ $(q, g) \leftarrow \text{Gen}_{\text{DL}}(1^k)$ $a, b \xleftarrow{\$} \mathbb{Z}_q$ $w \leftarrow \text{A}(1^k, q, g, g^a, g^b)$ If $w = g^{ab}$ Then return 1 Else return 0
---	--

Figure 2.3: The experiments used to define the discrete logarithm (left) and the computational Diffie-Hellman (right) assumptions.

The computational Diffie-Hellman assumption is a stronger assumption than the discrete logarithm assumption, since if an adversary succeeds in computing discrete logarithms, then it can solve the computational Diffie-Hellman problem as well.

2.3 Interactive Proof Systems

A *language* is a set. A class of languages that are often referred to in cryptography is the class NP. Formally, the class NP is defined as follows.

Definition 2.3.1 (NP) *A language L is in NP if there exists a boolean relation $R_L \subseteq \{0, 1\}^* \times \{0, 1\}^*$ and a polynomial $p(\cdot)$ such that membership in R_L can be determined in polynomial time and $x \in L$ if and only if there exists a w such that $|w| \leq p(|x|)$ and $(x, w) \in R_L$. Such a w is called a witness for $x \in L$.*

Thus, NP consists of the set of all languages for members of which short and efficiently-verifiable proofs of membership exists. Another way of looking at NP is that it is the set of all instances of computational problems that have solutions that, once given, can be efficiently tested for validity. That is, a member x is seen as a problem instance, a witness w as a solution to it, and determining membership in R_L as verifying the validity of the solution.

Consider a pair of entities that interact with each other: a prover who follows the algorithm P and a verifier that follows the algorithm V . The interaction is carried out in *rounds* (a.k.a. *moves*). In the beginning of each round, one entity starts calculating the round output according to the inputs it has, the quantities it has received from the other entity in the previous rounds, and the algorithm it follows. The round output consists of two parts: a part that is supposed to be kept to the entity itself and another part that is supposed to be sent to the other entity. The active entity then sends the latter part to the other entity and this finishes the round.

In the beginning of their interaction both P and V receive a number of inputs. The inputs in common between both are called *public inputs* and the ones exclusive to each are called *private inputs*.

The prover wants to prove to the verifier that a public input x does indeed belong to a language L . An interactive proof system is a pair (P, V) such that V is polynomial-time, P can convince V if $x \in L$, and if $x \notin L$, no matter what the proving party does it cannot convince the verifier. The last two criteria are called completeness and soundness, respectively. The notion of *interactive proof systems* originating from the seminal work of Goldwasser et al. [GMR89] formalizes the above requirements. We review the definition of these proofs and some other cryptographic schemes from [Gol01] in the following.

Definition 2.3.2 (Interactive Proof System) *A pair (P, V) is called an interactive proof system for a language L if the following conditions hold:*

Completeness: for every $x \in L$ the probability that V is convinced after interaction with P on public input x is 1, and

Soundness: for some polynomial p , for every $x \notin L$, and every algorithm P^ the probability that V is not convinced after interaction with P^* on public input x is at least $1/p(|x|)$.*

Define *non-interactive proof system* in the Random Oracle Model

A *public-coin* protocol is a protocol in which the verifier chooses all its messages during the protocol run randomly from publicly known sets. A three-move public-coin protocol can be written in the so called *canonical* form as shown in Figure 2.4. The prover algorithm consists of a pair of algorithms, respectively for so-called *committing* and *responding*, denoted by $P = (Cmt, Rsp)$. The verifier algorithm, in turn, consists of a pair of algorithms, respectively for so-called *challenging* and *deciding*, denoted by $V = (Chl, Dcd)$, where the challenging algorithm is limited to only drawing a challenge randomly from a publicly-known set, called the *challenge space*. The work-flow of the algorithm is as follows. In the first move, the prover runs the algorithm Cmt to compute the *commitment* Cmt and sends it to the verifier. Then the verifier chooses a random *challenge* Chl from a challenge space $ChSp$ and sends it back in the second move. The prover then computes a *response* Rsp according to the algorithm Rsp based on the information from the first run and the challenge, and sends Rsp to the verifier. Algorithm Dcd is run by the verifier at the end to compute a *decision* d based on the commitment, the challenge and the response.

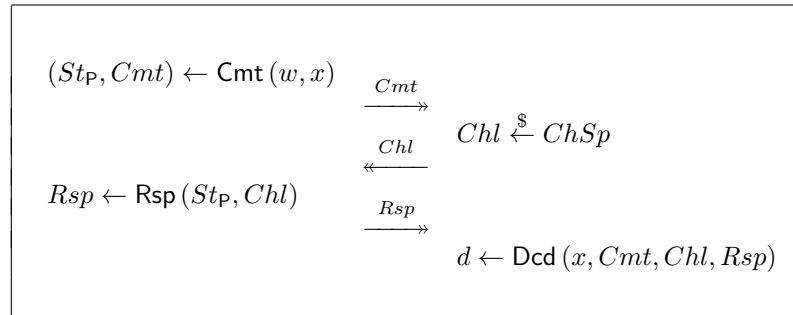


Figure 2.4: A three-move public-coin protocol in the canonical form, with prover $P = (Cmt, Rsp)$, verifier $V = (Chl, Dcd)$, prover's private input w , verifier's private input nil, and public input x

Zero knowledge proofs, originally defined by Goldwasser et al. [GMR89], are, intuitively, proofs that convey no “knowledge” to the verifier. The term knowledge refers to the ability of a party to feasibly calculate a particular quantity. In other words, zero knowledge proofs are proofs that do not increase

the verifier's computational ability. That is to say, definition of zero knowledge should guarantee that whatever the verifier can feasibly calculate after its interaction with the prover, it can feasibly calculate on its own. One way to guarantee this is to require that whatever the verifier “sees” during its interaction with the prover can be simulated by itself, no matter what strategy it follows. Formally, the *view* of a verifier interacting with a prover is defined as the random variable containing its random choices and the messages it receives from the prover during the interaction. Furthermore, (computational) zero knowledge is defined as follows.

Definition 2.3.3 (Zero Knowledge) *Let (P, V) be an interactive proof system. We say that (P, V) is zero knowledge if for every probabilistic polynomial-time algorithm V^* there exists a probabilistic polynomial-time algorithm M^* , called the simulator, such that the following two ensembles are computationally indistinguishable: (i) the view of V^* interacting with P on public input x , and (ii) the output of M^* on input x .*

Equivalently, one can require M^* to simulate a *transcript* indistinguishable from that of communications between P and V^* for every V^* , with the transcript of the communications consisting of all the messages the parties send to each other during the interaction. Note that, transcript can be calculated knowing the algorithm and view of the verifier.

A relaxed version of the above definition is the notion of *honest-verifier zero-knowledge* (HVZK). An honest verifier zero knowledge proof guarantees no knowledge conveyed to a verifier, only if the verifier behaves honestly. That is, the verifier follows the protocol V that it is supposed to follow. Formally, definition of honest verifier zero knowledge is derived from the above definition by restricting the verifier algorithm to V .

Definition 2.3.4 (Honest Verifier Zero Knowledge) *Let (P, V) be an interactive proof system. We say that (P, V) is honest verifier zero knowledge if there exists a probabilistic polynomial-time algorithm M , called the simulator, such that the following two ensembles are computationally indistinguishable: (i) the view of V interacting with P on public input x , and (ii) the output of M on input x .*

An interesting result in this area is the following proved by Goldreich et al. [GMW91].

Theorem 2.3.1 *If one-way functions exist, then any NP language has a zero knowledge interactive proof.*

In some cryptographic applications, it is more natural to consider an alternative definition to that of the interactive proof systems. This alternative is the concept of *proofs of knowledge*. A proof of knowledge guarantees that only the provers who explicitly *know* a witness to membership of x in L are able to convince a verifier. To formalize such a property, existence of an *extractor* is required that is able to extract an actual witness out of a convincing prover. This guarantees that any convincing prover must be able to feasibly calculate a witness, or technically “know” a witness. Bellare and Goldreich define proofs of knowledge as follows [BG92].

Definition 2.3.5 (Proof of Knowledge) *A pair (P, V) is a proof of knowledge if the following conditions hold:*

Non-Triviality: for every $x \in L$ the probability that V is convinced after interaction with P on public

input x is 1, and

Validity (with Error κ): there exists a constant $c > 0$ and a probabilistic polynomial-time algorithm K , called the extractor, such that for every algorithm P^* and every $x \in L$, K satisfies the following: if the probability that V is convinced after interaction with P^* on public input x , defined as $p(x)$, is greater than $\kappa(x)$, then on input x and oracle access to $P^*(x)$, K outputs a witness w for $x \in L$ in expected time bounded by

$$\frac{|x|^c}{p(x) - \kappa(x)}.$$

Witness indistinguishability is another notion of security for interactive proof systems that was first introduced by Feige and Shamir [FS90]. Witness indistinguishability guarantees that provers using different witnesses remain indistinguishable to the verifier. Hence it can be seen as a relaxation of the notion of zero knowledge.

Definition 2.3.6 (Witness Indistinguishability) Let (P, V) be an interactive proof system. We say that (P, V) is witness indistinguishable if for every probabilistic polynomial-time algorithm V^* and any two witnesses w_1 and w_2 , such that $(x, w_1) \in R_L$ and $(x, w_2) \in R_L$, the following two ensembles are computationally indistinguishable: (i) the view of V^* interacting with P on private input w_1 , and (ii) the view of V^* interacting with P on private input w_2 .

2.4 Identification and Signature Schemes

An *identification scheme* is a cryptographic scheme using which a prover with a secret key can identify herself to a verifier who knows the corresponding public key. Fiat and Shamir were the first to propose efficient identification schemes [FS86]. Formally, an identification scheme is defined as follows.

Definition 2.4.1 (Identification Scheme) An identification scheme $I = (\text{KeyGen}, P, V)$ is a triple of randomized algorithms as follows:

KeyGen is the key generation algorithm that on input a security parameter k returns a pair consisting of a secret key sk and a public key pk and

P and V are the prover and verifier algorithms, respectively, and (P, V) defines a protocol that is run on P 's private input sk , V 's null private input, and public input pk , and at the end of the protocol V outputs a decision d indicating its acceptance or rejection of P 's proof.

A natural requirement is that a prover who knows the secret key must be able to always convince a verifier. This defines the correctness requirement for identification schemes as follows.

Definition 2.4.2 (Correctness of an Identification Scheme) An identification scheme $I = (\text{KeyGen}, P, V)$ is said to be correct if for any pair (sk, pk) generated by KeyGen the probability that V outputs 1 after interaction with P on P 's private input sk , V 's null private input, and public input pk , is 1.

The accepted notion of security for identification schemes is that of security against *impersonation* against *passive*, *active*, or *concurrent* attacks. These attacks were first defined by Fiege et al. [FFS88]

and later extended by Bellare and Palacio [BP02]. We denote these notions by IMP-PA, IMP-AA, and IMP-CA, respectively. A passive impersonator is one that attempts to impersonate a prover after eavesdropping its identification conversations with others for some time. An active impersonator gets to actually play the role of a verifier for some time and then attempts to impersonate the prover. A concurrent impersonator gets to play the role of multiple verifiers *concurrently* for some time and then attempts to impersonate the prover. Concurrency refers to the impersonator's ability to run arbitrarily interleaved protocols with different *clones* of the prover, that is instances of the same prover algorithm with the same input (secret key) but different random choices. To formalize such attacks a transcript oracle \mathcal{Tr} and a prover oracle \mathcal{Prov} are introduced. Oracle \mathcal{Tr} is provided to a passive impersonator and upon query, provides a transcript of an identification protocol. Oracle \mathcal{Prov} is provided to active and concurrent impersonators, and upon query on a session identifier and a message, provides the response of the prover clone of the queried session to the queried message. The oracle only allows serial sessions with different clones of the prover for active impersonators, but allows arbitrarily interleaved sessions with different clones of the prover for concurrent impersonators. Impersonation attack takes place in two stages. In the first stage, the adversary gathers information about the prover by either eavesdropping the prover's conversations with others (passive attack), playing the role of a verifier (active attack), or playing the role of multiple verifiers concurrently (concurrent attack) until it declares it is ready to impersonate. In the second stage, the adversary attempts to impersonate the prover for a verifier. Hence, an adversary A is denoted as a pair of algorithms consisting of a *cheating verifier* CV that plays in the first stage and a *cheating prover* CP that plays in the second stage. The formal definition comes in the following.

Definition 2.4.3 (Security against Impersonation) For an identification scheme $I = (\text{KeyGen}, P, V)$ and an adversary $A = (CP, CV)$, we define the advantage of A in impersonation under passive, active, and concurrent attacks as the probabilities below, where the experiments $\text{Expt}_{I,A}^{\text{IMP-ATK}}(k)$ for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles are defined as in Figure 2.5.

$$\text{Adv}_{I,A}^{\text{IMP-ATK}}(k) \triangleq \Pr [\text{Expt}_{I,A}^{\text{IMP-ATK}}(k) = 1]$$

We say I is IMP-ATK-secure for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ if the maximum advantage a polynomial-time adversary can get in an IMP-ATK-attack is negligible in k .

Signature schemes are among the basic tools in public key cryptography envisioned by Diffie and Hellman [DH76]. They allow a signer holding a secret key to generate, on a message, a signature that can be verified later by a verifier holding the corresponding public key. Formally, a signature scheme is defined as follows.

Definition 2.4.4 (Signature Scheme) A signature scheme $S = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is a triple of algorithms as follows:

KeyGen is the randomized key generation algorithm that on input a security parameter k returns a pair consisting of a secret key sk and a public key pk ,

Sign is the randomized signing algorithm that on input a secret key sk and a message m returns a signature σ on the message, and

Verify is the (usually) deterministic verification algorithm that on input a public key pk , a message m , and a candidate signature σ returns a binary decision d indicating the validity of the candidate

<p>Experiment $\text{Expt}_{\mathcal{I},\mathcal{A}}^{\text{IMP-ATK}}(k)$</p> <p>$PS \leftarrow \emptyset$</p> <p>$(pk, sk) \leftarrow \text{KeyGen}(1^k)$</p> <p>If $\text{ATK} = \text{PA}$</p> <p style="padding-left: 20px;">Then let \mathcal{O} denote $\mathcal{T}r$</p> <p style="padding-left: 20px;">Else let \mathcal{O} denote $\mathcal{P}rov$</p> <p>$St_{\mathcal{A}} \leftarrow \text{CV}(1^k, pk : \mathcal{O})$</p> <p>$(\varepsilon, d) \leftarrow [\text{CP}(St_{\mathcal{A}}) \leftrightarrow \mathcal{V}](pk)$</p> <p>Return d</p>	<p>Oracle $\mathcal{T}r$</p> <p>$Tr \leftarrow \text{Tr}[P(sk) \leftrightarrow \mathcal{V}](pk)$</p> <p>Return Tr</p> <hr/> <p>Oracle $\mathcal{P}rov(s, M_{\text{in}})$</p> <p>If $s \notin PS$ then</p> <p style="padding-left: 20px;">If $\text{ATK} = \text{AA}$ then $PS \leftarrow \{s\}$</p> <p style="padding-left: 20px;">If $\text{ATK} = \text{CA}$ then $PS \leftarrow PS \cup \{s\}$</p> <p style="padding-left: 20px;">$St_{\mathcal{P}}[s] \leftarrow (sk; \rho)$ for random ρ</p> <p style="padding-left: 20px;">$(M_{\text{out}}, St_{\mathcal{P}}[s]) \leftarrow P(M_{\text{in}}, St_{\mathcal{P}}[s])$</p> <p>Return M_{out}</p>
---	--

Figure 2.5: The experiments used to define identification scheme IMP-ATK-security for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles. PS denotes the set of active prover sessions that, in the case of an active attack, is limited to only *one* active session at a time. ε denotes the empty string. s denotes the session identifier.

signature with respect to the public key and the message.

A natural expectation is that any signature generated by the signing algorithm passes the verification test against the corresponding public key. Formally, the notion of correctness is defined as follows for signature schemes.

Definition 2.4.5 (Correctness of a Signature Scheme) *A signature scheme $S = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is said to be correct if for any pair (sk, pk) generated by KeyGen and any message m , if $\sigma \leftarrow \text{Sign}(sk, m)$, the probability that $\text{Verify}(pk, m, \sigma)$ outputs 1 is 1.*

The standard notion for security for signature schemes is that of *existential unforgeability under chosen message attack*, denoted by EUF-CMA, first rigorously defined by Goldwasser et al. [GMR88]. The adversary is given access to a *signing oracle* that provides it with a signature on its chosen messages and it succeeds in the attack if it is able to forge a valid signature on a new message. The formal definition is as follows.

Definition 2.4.6 (Unforgeability under Chosen Message Attack) *For a signature scheme $S = (\text{KeyGen}, \text{Sign}, \text{Verify})$ and an adversary \mathcal{A} , we define the advantage of \mathcal{A} in existential forgery under a chosen message attack as the probability below, where the experiment $\text{Expt}_{S,\mathcal{A}}^{\text{EUF-CMA}}(k)$ and the signing oracle are defined as in Figure 2.6.*

$$\text{Adv}_{S,\mathcal{A}}^{\text{EUF-CMA}}(k) \triangleq \Pr [\text{Expt}_{S,\mathcal{A}}^{\text{EUF-CMA}}(k) = 1]$$

We say S is EUF-CMA-secure if the maximum advantage a polynomial-time adversary can get in an EUF-CMA-attack is negligible in k .

Experiment $\text{Expt}_{S,A}^{\text{EUF-CMA}}(k)$ $M \leftarrow \emptyset$ $(pk, sk) \leftarrow \text{KeyGen}(1^k)$ $(m, \sigma) \leftarrow A(1^k, pk : \text{Sign}(\cdot))$ If $m \notin M$ and $\text{Verify}(pk, m, \sigma) = 1$ Then return 1 Else return 0	Oracle $\text{Sign}(m)$ $M \leftarrow M \cup \{m\}$ Return $\text{Sign}(sk, m)$
--	---

Figure 2.6: The experiment used to define signature scheme EUF-CMA-security and the relevant oracle. M denotes the set of messages queried to the signing oracle.

Any three-move public-coin protocol can be transformed into a signature scheme using the *Fiat-Shamir transform* [FS86]. The idea is to replace the verifier with a random function. The rationale behind the idea is that what enables the prover to cheat in such a protocol is the fact that the verifier provides her with an *unpredictable* challenge. Fiat and Shamir argue that this unpredictable challenge can be mimicked by output of a truly random function. The Fiat-Shamir transform, denoted by FS, applied to identification schemes, is formally defined as follows.

Definition 2.4.7 (Fiat-Shamir Transform) Let $I = (\text{KeyGen}, P, V)$ be a three-move public-coin identification scheme written in canonical form as in Figure 2.4, that is $P = (\text{Cmt}, \text{Rsp})$ and $V = (\text{Chl}, \text{Dcd})$. The Fiat-Shamir transform of I is defined as the signature scheme $\text{FS}(I) = (\text{KeyGen}, \text{Sign}, \text{Verify})$, where the key generation algorithm is defined to be the same and the signing and verification algorithms are defined as in Figure 2.7. $H : \{0, 1\}^* \mapsto \text{ChSp}$ is a random oracle.

Algorithm $\text{Sign}(sk, m : H)$ $(St_P, \text{Cmt}) \leftarrow \text{Cmt}(sk, pk)$ $\text{Chl} \leftarrow H(\text{Cmt} \parallel m)$ $\text{Rsp} \leftarrow \text{Rsp}(St_P, \text{Chl})$ Return (Cmt, Rsp)	Algorithm $\text{Verify}(pk, m, \sigma : H)$ Parse σ as (Cmt, Rsp) $\text{Chl} \leftarrow H(\text{Cmt} \parallel m)$ Return $\text{Dcd}(pk, \text{Cmt}, \text{Chl}, \text{Rsp})$
--	--

Figure 2.7: The Fiat-Shamir transform of an identification scheme in canonical form as in Figure 2.4. $H : \{0, 1\}^* \mapsto \text{ChSp}$ is a random oracle.

Abdalla et al. proved that the Fiat-Shamir transform yields EUF-CMA-secure signatures if the underlying identification scheme has certain properties [AABN02]. The following theorem is a special case of what they proved.

Theorem 2.4.1 Let $I = (\text{KeyGen}, P, V)$ be a three-move public-coin identification scheme with a commitment Cmt distributed uniformly over a set CmtSp , such that $1/|\text{CmtSp}|$ is negligible in the security parameter k . If I is IMP-PA-secure, then $\text{FS}(I)$ is EUF-CMA-secure in the Random Oracle Model.

2.5 Identity-Based Cryptography

Identity-based cryptosystems were proposed by Shamir [Sha84] to overcome the problem of lack of a comprehensive public-key infrastructure, which implementation of the public-key cryptosystems faces. In identity-based systems, public-key certificates are no longer needed, and the identities of the users are used as their public keys. However, users lose their ability to construct their own secret keys by themselves and must depend on a *key-generation center* (KGC) to provide them with their respective private keys.

An *identity-based identification scheme* is a cryptographic scheme using which a prover with a secret key given to her by a key-generation center can identify herself to a verifier who knows her identity. Formally, an identity-based identification scheme is defined as follows.

Definition 2.5.1 (Identity-Based Identification Scheme) *An identity-based identification scheme $\text{IBI} = (\text{MKeyGen}, \text{UKeyGen}, \text{P}, \text{V})$ is a quadruple of randomized algorithms as follows:*

MKeyGen is the master key generation algorithm that on input a security parameter k returns a pair consisting of a master secret key msk and a master public key mpk ,

UKeyGen is the user key generation algorithm that on input a master secret key msk and a user identity I returns a user secret key usk , and

P and V are the prover and verifier algorithms, respectively, and (P, V) defines a protocol that is run on P 's private input usk , V 's null private input, and public input (mpk, I) , and at the end of the protocol V outputs a decision d indicating its acceptance or rejection of P 's proof.

The correctness requirement for an identity-based identification scheme is defined analogously.

Definition 2.5.2 (Correctness of an Identity-Based Identification Scheme) *An identity-based identification scheme $\text{IBI} = (\text{MKeyGen}, \text{UKeyGen}, \text{P}, \text{V})$ is said to be correct if for any pair (msk, mpk) generated by MKeyGen and any I , if usk is generated by UKeyGen on input msk and I , the probability that V outputs 1 after interaction with P on P 's private input usk , V 's null private input, and public input (mpk, I) , is 1.*

The accepted notion of security for identification schemes is that of identity-based security against impersonation against passive, active, or concurrent attacks, as first rigorously defined by Bellare et al. [BNN04]. We denote these notions by ID-IMP-PA, ID-IMP-AA, and ID-IMP-CA, respectively. The adversary attacking an identity-based identification scheme, gets to either eavesdrop conversations of the provers of its choice (passive attack), play the role of a verifier for provers of its choice (active attack), or play the role of multiple verifiers concurrently for provers of its choice (concurrent attack). Furthermore, the adversary is given the ability to *initialize* and *corrupt* the provers of its choice. Upon initialization of a prover, a user secret key is generated for the prover. Upon corruption of a prover, its user secret key is revealed to the adversary. Again, the attack is carried out in two stages. In the first stage, a cheating verifier CV gathers information about the system and chooses a target identity to attack. In the second stage, a cheating prover CP attempts to impersonate the target identity, given the information gathered by the cheating verifier in the first stage. The formal definition comes in the following.

Definition 2.5.3 (Identity-Based Security against Impersonation) For an identity-based identification scheme $\text{IBI} = (\text{MKeyGen}, \text{UKeyGen}, \text{P}, \text{V})$ and an adversary $\text{A} = (\text{CP}, \text{CV})$, we define the advantage of A in identity-based impersonation under passive, active, and concurrent attacks as the probabilities below, where the experiments $\text{Expt}_{\text{IBI}, \text{A}}^{\text{ID-IMP-ATK}}(k)$ for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles are defined as in Figure 2.8.

$$\text{Adv}_{\text{IBI}, \text{A}}^{\text{ID-IMP-ATK}}(k) \triangleq \Pr [\text{Expt}_{\text{IBI}, \text{A}}^{\text{ID-IMP-ATK}}(k) = 1]$$

We say IBI is ID-IMP-ATK-secure for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ if the maximum advantage a polynomial-time adversary can get in an ID-IMP-ATK-attack is negligible in k .

<p>Experiment $\text{Expt}_{\text{IBI}, \text{A}}^{\text{ID-IMP-ATK}}(k)$</p> <p>$PS, HU, CU, AU \leftarrow \emptyset$</p> <p>$(mpk, msk) \leftarrow \text{MKeyGen}(1^k)$</p> <p>If $\text{ATK} = \text{PA}$</p> <p> Then let \mathcal{O} denote \mathcal{Tr}</p> <p> Else let \mathcal{O} denote \mathcal{Prov}</p> <p>$(I, St_A) \leftarrow \text{CV}(1^k, mpk : \text{Init}, \text{Crpt}, \mathcal{O})$</p> <p>If $I \notin HU$ then return 0</p> <p>$AU \leftarrow \{I\}$</p> <p>$(\varepsilon, d) \leftarrow [\text{CP}(St_A : \text{Init}, \text{Crpt}, \mathcal{O}) \leftrightarrow \text{V}](mpk, I)$</p> <p>Return d</p>	<p>Oracle $\text{Crpt}(I)$</p> <p>If $I \notin HU \setminus AU$ then return \perp</p> <p>$CU \leftarrow CU \cup \{I\}$</p> <p>$HU \leftarrow HU \setminus \{I\}$</p> <p>Return $usk[I]$</p> <hr/> <p>Oracle $\mathcal{Tr}(I)$</p> <p>If $I \notin HU$ then return \perp</p> <p>$Tr \leftarrow \text{Tr}[\text{P}(usk[I]) \leftrightarrow \text{V}](mpk, I)$</p> <p>Return Tr</p> <hr/> <p>Oracle $\mathcal{Prov}(I, s, M_{\text{in}})$</p> <p>If $I \notin HU \setminus AU$ then return \perp</p> <p>If $(I, s) \notin PS$ then</p> <p> If $\text{ATK} = \text{AA}$ then $PS \leftarrow \{(I, s)\}$</p> <p> If $\text{ATK} = \text{CA}$</p> <p> Then $PS \leftarrow PS \cup \{(I, s)\}$</p> <p> $St_P[I, s] \leftarrow (usk[I]; \rho)$ for random ρ</p> <p>$(M_{\text{out}}, St_P[I, s]) \leftarrow \text{P}(M_{\text{in}}, St_P[I, s])$</p> <p>Return M_{out}</p>
<p>Oracle $\text{Init}(I)$</p> <p>If $I \in HU \cup CU \cup AU$ then return \perp</p> <p>$usk[I] \leftarrow \text{UKeyGen}(msk, I)$</p> <p>$HU \leftarrow HU \cup \{I\}$</p> <p>Return 1</p>	

Figure 2.8: The experiments used to define identity-based identification scheme ID-IMP-ATK-security for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles. PS denotes the set of active prover sessions that, in the case of an active attack, is limited to only *one* active session at a time. ε denotes the empty string. s denotes the session identifier. HU , CU , and AU denote the sets of honest, corrupted, and attacked users, respectively.

Identity-Based signature schemes are defined analogously. They allow a signer holding a secret key given to her by a key-generation center, to generate, on a message, a signature that can be verified later by a verifier knowing her identity. Formally, an identity-based signature scheme is defined as

follows.

Definition 2.5.4 (Identity-Based Signature Scheme) An identity-based signature scheme $\text{IBS} = (\text{MKeyGen}, \text{UKeyGen}, \text{Sign}, \text{Verify})$ is a quadruple of algorithms as follows:

MKeyGen is the randomized master key generation algorithm that on input a security parameter k returns a pair consisting of a master secret key msk and a master public key mpk ,

UKeyGen is the randomized user key generation algorithm that on input a master secret key msk and a user identity I returns a user secret key usk ,

Sign is the randomized signing algorithm that on input a user secret key usk and a message m returns a signature σ on the message, and

Verify is the deterministic verification algorithm that on input a master public key mpk , a user identity I , a message m , and a candidate signature σ returns a binary decision d indicating the validity of the candidate signature with respect to the master public key, the user identity, and the message.

Furthermore, the notion of correctness is defined as follows for identity-based signature schemes.

Definition 2.5.5 (Correctness of an Identity-Based Signature Scheme) An identity-based signature scheme $\text{IBS} = (\text{MKeyGen}, \text{UKeyGen}, \text{Sign}, \text{Verify})$ is said to be correct if for any pair (msk, mpk) generated by *MKeyGen*, any user identity I , and any message m , if usk is generated by *UKeyGen* on input msk and I , and $\sigma \leftarrow \text{Sign}(\text{usk}, m)$, the probability that $\text{Verify}(\text{mpk}, I, m, \sigma)$ outputs 1 is 1.

<p>Experiment $\text{Expt}_{\text{IBS}, A}^{\text{ID-EUF-CMA}}(k)$ $HU, CU \leftarrow \emptyset$ $(\text{mpk}, \text{msk}) \leftarrow \text{MKeyGen}(1^k)$ $(I, m, \sigma) \leftarrow A(1^k, \text{mpk} : \text{Init}, \text{Sign}, \text{Crpt})$ If $I \in HU$ and $m \notin M[I]$ and $\text{Verify}(\text{mpk}, I, m, \sigma) = 1$ Then return 1 Else return 0</p>	<p>Oracle $\text{Init}(I)$ If $I \in HU \cup CU$ then return \perp $\text{usk}[I] \leftarrow \text{UKeyGen}(\text{msk}, I)$ $M[I] \leftarrow \emptyset$ $HU \leftarrow HU \cup \{I\}$ Return 1</p>
<p>Oracle $\text{Sign}(I, m)$ If $I \notin HU$ then return \perp $M[I] \leftarrow M[I] \cup \{m\}$ Return $\text{Sign}(\text{usk}[I], m)$</p>	<p>Oracle $\text{Crpt}(I)$ If $I \notin HU$ then return \perp $CU \leftarrow CU \cup \{I\}$ $HU \leftarrow HU \setminus \{I\}$ Return $\text{usk}[I]$</p>

Figure 2.9: The experiment used to define identity-based signature scheme ID-EUF-CMA-security and the relevant oracles. HU and CU denote the sets of honest and corrupted users, respectively. $M[I]$ denotes the set of messages queried to the signing oracle for identity I .

The standard notion for security for signature schemes is that of *identity-based existential unforgeability*

under chosen message attack, denoted by ID-EUF-CMA, first rigorously defined by Cha and Cheon [CC03]. The adversary is given access to a *signing oracle* that provides it with a signature of its chosen identity on its chosen messages. Furthermore, the adversary is given the ability to initialize and corrupt the provers of its choice. Upon initialization of a prover, a user secret key is generated for the prover. Upon corruption of a prover, its user secret key is revealed to the adversary. The adversary is said to succeed in the attack if it is able to forge a valid signature of an uncorrupted identity on a new message. The formal definition is as follows.

Definition 2.5.6 (Identity-based Unforgeability under Chosen Message Attack) *For an identity-based signature scheme $\text{IBS} = (\text{MKeyGen}, \text{UKeyGen}, \text{Sign}, \text{Verify})$ and an adversary \mathbf{A} , we define the advantage of \mathbf{A} in identity-based existential forgery under a chosen message attack as the probability below, where the experiment $\text{Expt}_{\text{IBS}, \mathbf{A}}^{\text{ID-EUF-CMA}}(k)$ and the signing oracle are defined as in Figure 2.9.*

$$\text{Adv}_{\text{IBS}, \mathbf{A}}^{\text{ID-EUF-CMA}}(k) \triangleq \Pr [\text{Expt}_{\text{IBS}, \mathbf{A}}^{\text{ID-EUF-CMA}}(k) = 1]$$

We say IBS is ID-EUF-CMA-secure if the maximum advantage a polynomial-time adversary can get in an ID-EUF-CMA-attack is negligible in k .

Chapter 3

Concurrently-Secure Credential Ownership Proofs

3.1 Introduction

A credential system consists of users and organizations. Organizations issue credentials to users. Users can later show their credentials to (the same or) other organizations to enjoy the privileges they are entitled to. In this chapter, we formalize *credential ownership proofs* that allow a credential owner to prove ownership of her credential to a verifier without enabling the verifier to impersonate her at a later time. We give generic constructions for credential ownership proofs based on identity-based schemes and prove them secure. We also give an efficient construction based on RSA system and compare its efficiency with previous schemes.

A COP for a signature scheme is an interactive protocol attached to the scheme, that allows a credential holder to prove interactively the ownership of a claimed credential to a verifier. A secure COP must ensure security of both credential holders and credential issuer. That is, prover's interaction with multiple verifiers should not allow mis-use of the system by enabling a successful run of the protocol without having the required credentials. We show that COPs can be much more efficient than zero-knowledge proofs (see Table 3.3). In particular, the COP for RSA-based credentials that we construct, is based on the GQ protocol [GQ88], which is known to be only *honest verifier zero knowledge*. This makes COPs and their secure construction of immediate practical importance.

We note the following remarks with respect to users' privacy in a credential system based on COPs:

- In such a credential system, showings of a ticket are linkable through the value of the signed message, which, as mentioned before, must be unique to prevent double spending. However users cannot be traced as the tickets can be purchased by any user. Such linkability property enables conducting (anonymous) statistical analyses on the behavior of users.
- The system allows users to make 'clones' of their tickets and share them with their trusted friends. The 'double spending' protection of the ticket controlling system ensures that for each game only a single user will actually use the ticket. The users, however, have higher flexibility with the tickets and are able to share their tickets, which is a very attractive property.

3.1.1 Related Work

Credential systems and their vast range of security properties in different applications have been intensively studied in recent years. The closest work to ours is *Universal Designated Verifier Signature Proofs (UDVSP)* introduced by Baek et al. [BSS05]. A UDVSP is similar to a COP as it enables a signature holder to prove the ownership of a signature to a verifier, however, the security properties of the two are very different. In UDVSP the goal is to remove a restriction of *Universal Designated Verifier Signature (UDVS)* proposed by Steinfeld et al. [SBWP03] and the security model is mainly geared to ensure security of a single credential holder and with no concern about security of the credential issuer. In particular, in their security model, there is no security guarantee against an adversary who corrupts a number of credential holders and gets to know their credentials (e.g. ticket controller who collects many copies of tickets) or possibly can obtain tickets of his choice by directly asking the credential issuer. Baek et al’s definition of security is one-sided and resembles security definition of an identification protocol. It focuses on the security of a single credential holder (corresponding to the owner of the secret in identification protocols), but not the issuing authority, and does not address a system of different credential holders. In Section 3.2.1, we further elaborate on these definitions and show that our security requirements captures all the requirements of the above scenario.

In the following we outline other most relevant systems in relation to our work.

Anonymous Credentials. Anonymous credential systems (a.k.a. *pseudonym systems*) ensure privacy of users in securely accessing services of organizations. Organizations issue credentials on users’ pseudonyms, and hence the name ‘pseudonym systems’. Pseudonym systems were introduced by Chaum [Cha85, CE87] and more recently further formalized and studied in [LRSW99, Lys02]. The aim of these systems is to simultaneously guarantee anonymity of credential holders, unlinkability of credential showings, and non-transferability of credentials. These properties ensure ultimate users’ privacy and organizations’ security at a high computation and communication cost. Indeed, most of such systems use multiple zero knowledge protocols for issuing and verifying credentials (see e.g. [Bra00, Lys02]). In many real life applications, such as our motivating scenario above, a more moderate level of security is considered sufficient as long as much higher efficiency can be provided. It is however crucial to clearly state the required security properties and prove it is achievable for the proposed constructions. Our work is a step in this direction.

Identification Protocols. One of the security goals of a COP protocol, protection of the secret of the credential holder, can be seen as parallel to the security goal of an identification protocol, if authority’s signature is the prover’s *secret* in the identification protocol. Security requirement of COP with regard to credential holder is in line with that of identification protocols as defined in [FFS88], with attacker’s goal being impersonation of the prover, without having her secret. The strongest attack model for identification protocols allows the adversary to pose as a verifier and run arbitrary-interleaved (i.e. concurrent) sessions [BP02] with the prover before taking up the role of a malicious prover. We model a similar attacker for COP (having concurrent sessions with prover) and also allow the attacker to have access to signing oracle that models the credential issuer. COP security also requires security for credential issuer, which is not required in identification protocols.

DMA and NTS Schemes. Deniable Message Authentication (DMA) [DDN00] and Non-Transitive Signature (NTS) [Des88, OO90] schemes enable a sender to construct an authenticated message for a receiver such that the receiver cannot convince a third party about the origin of the message. In other words, they provide unforgeability guarantee of a digital signature but the message can be repudiated as the receiver can simulate the transcript of the protocol. The security goals in DMA and NTS schemes are complementary to a basic credential system as described above. That is, the credential holder may require that his credential ownership proof be repudiable. Although in a COP protocol the proof is interactive, but the proof transcript may not be simulatable and so in this sense, non-repudiable. DMA and NTS systems use zero knowledge proofs to provide repudiation property. Such proofs are costly and we wish to avoid them in simple credential systems, like the ones discussed above. In fact this (avoiding ZK proofs) has been one of the motivations of our work.

3.1.2 Our Contributions

We formalize the security model of a credential system, consisting of credential issuers, credential holders, and verifiers, with emphasis on security of credential showing. A credential issuer signs a credential using a secure (i.e. unforgeable against chosen message attack) signature scheme. A credential holder wants to prove to a verifier that he has a credential but would like to make sure that the credential cannot be copied. This protection against copying is crucial in scenarios as described above and so it is imperative that the credential holder only provide ‘proof’ for ownership of the credential and not the actual credential. An immediate question is what properties such a proof should have, and if efficient constructions exist.

In this chapter, we define *credential ownership proof (COP)* protocols for signature schemes. We give security notions for COPs that capture precisely the security requirements of the scenario given earlier, and guarantees security of credential holders and issuers both. COPs have also the desirable property that they provide some level of control for the credential holders over unwanted distribution of their credentials. This, however, is not in the strongest sense of repudiability of the credential, but as a side effect of employing proofs.

Next, we consider construction of COPs. We provide two generic constructions for signature schemes and their associated secure COPs. The first construction is based on identity-based encryption (IBE). It has been observed that a secure identity-based encryption scheme can be utilized to construct a secure signature scheme¹. We show that it is possible to define a secure COP for this scheme in a natural way. We reduce security of this COP to one-wayness of the underlying IBE under chosen-ciphertext attacks (denoted OWE-ID-CCA). This is a new security notion for IBE, that is, in terms of strength, in between the two (folklore standard) notions of one-wayness (under chosen-plaintext attacks, denoted OWE-ID) and indistinguishability under chosen-ciphertext attacks (denoted IND-ID-CCA), introduced in [BF01]. We show that this generic construction results in a scheme provably-secure based on standard computational assumptions in the *standard model* (i.e. not in the *random oracle model*).

The second generic construction is based on identity-based identification (IBI). We show an equivalence

¹This is attributed to Moni Naor [BF01, p. 226].

between a signature that is EUF-CMA (i.e. existentially unforgeable under chosen-message attacks) plus an associated COP that is secure in our model, and an IBI that is secure against impersonation under concurrent attacks (in the sense of [BNN04]). We show a one-to-one relationship between entities and algorithms, and give a bilateral translation of security notions of the two.

An interesting observation in this context with regard to COPs is their application in construction of secure IBIs. Kurosawa and Heng [KH04] gave a generic construction for an IBI from a signature scheme and an HVZK proof of knowledge (PoK) protocol. Security of their construction however was proved only against a *passive* adversary. We show that replacing PoK protocol with a COP protocol in their construction will result in security against *active* and *concurrent* attacks. This results in a generic construction of secure IBI schemes from COP schemes.

Both generic constructions above use identity-based cryptography, which could be considered as less traditional and requiring more advanced knowledge of cryptography. It is desirable to have a signature scheme with an associated COP that are provably secure and use standard (textbook) cryptography. We show that, for a credential system based on RSA signature, a secure COP can be obtained based on the GQ identification protocol [GQ88]. GQ, as an identification protocol, is proved to be secure against impersonation under concurrent attacks [BP02]. It is also widely known that GQ can be used to prove knowledge of RSA-FDH signatures, although adequate formalization of this does not seem to exist in the literature. We prove that the COP protocol based on GQ is secure in our model under concurrent attacks.

Combining RSA-FDH signature scheme of [BR96] with the GQ based COP, results in a very efficient and provably-secure credential system that can be easily implemented using commonly used cryptographic libraries. Security of the system relies, in the random oracle model, on security of GQ as an identification protocol, which is, in turn, proved in [BP02] assuming *one-more RSA inversion* is hard (see Definition 2.2.2 on page 16). An interesting open question is construction of secure COPs for other traditional signature schemes.

3.2 Defining Credential Ownership Proofs

A credential is of the form (m, σ) , where m is the text of the credential and σ is the issuer's signature on m , generated using the standard signature scheme $S = (\text{KeyGen}, \text{Sign}, \text{Verify})$. To prove ownership of such a credential, we associate an interactive proof protocol with the signature scheme S , through which the credential-holder (prover) convinces the verifier that she owns a credential signed by an issuer that employs the signature S . Let ε denote the empty string.

Definition 3.2.1 (Credential Ownership Proof) *Associated with a standard signature S , we define a credential ownership proof (equiv. COP) protocol $S\text{-COP} = (P, V)$, consisting of a pair of algorithms: the prover P and the verifier V . Prover's private input is a signature σ and protocol's public inputs are the signature verification key pk and a message m . After the interaction with the prover, the verifier outputs a binary decision d . The protocol run is denoted by:*

$$(\varepsilon, d) \leftarrow [P(\sigma) \leftrightarrow V](pk, m) .$$

The working scenario for S-COP is as follows: When the credential-holder wants to ‘securely’ show her credential (m, σ) to a verifier, she first sends to the verifier the text of the credential (i.e., the message) m , on which she claims to have the issuer’s signature. The credential-holder and the verifier then interact with each other running P and V , respectively. At the end of this interaction, either the verifier accepts the prover’s claim of credential ownership or not. This is reflected in the verifier’s output d as a 1 if the verifier is convinced, and a 0 otherwise.

We require that an honest credential-holder can always convince the verifier, i.e., it must be guaranteed that if the signature σ is a valid signature on m with respect to pk , then the verifier must return 1. Hence, we define the correctness of the credential ownership proof protocol as follows.

Definition 3.2.2 (Correctness of a Credential Ownership Proof) *A credential ownership proof S-COP is said to be correct for a signature S if for any triple (pk, m, σ) , if $\text{Verify}(pk, m, \sigma) = 1$, then the probability that the verifier outputs 1 in a credential ownership proof protocol run with prover’s input σ and public input (pk, m) is 1.*

3.2.1 Defining Credential Ownership Proof Security

Intuitively, a secure credential ownership proof must not allow an adversary to prove knowledge of a credential it does not possess. The reasonable resources that one might consider to be provided to the adversary are being able to eavesdrop multiple credential ownership proof communications or even to play the role of a verifier for multiple times. These intuitions suggest that the proper formalization for credential ownership proof security is security against *impersonation* attacks. Inspired by definitions of security against impersonation attacks for identification schemes, we formalize security of credential ownership proofs as follows.

A COP-IMP-ATK adversary $A = (CV, CP)$ for $ATK \in \{PA, AA, CA\}$ is a pair of randomized polynomial-time algorithms: the *cheating verifier* and the *cheating prover*, respectively. The attack is carried out in two phases. Throughout both phases, the adversary, whether passive, active, or concurrent, is provided with credential initiation and corruption oracles that enable it to initiate a credential on any message of its choice and have the credential, respectively. Furthermore, a passive adversary is provided with a transcript oracle that enables it to have a transcript of a credential ownership proof on any message of its choice. Active and concurrent adversaries, instead, are provided with a prover oracle that enables them to interact with a single clone or multiple clones, respectively, of a prover of a credential on any message of their choice.

At the beginning of the first phase, the cheating verifier is given the public key, which has been generated through the signature key generation. Then it starts querying the oracles provided to it. When the initiation oracle is queried on a message m , a credential with message m is issued, that is message m is signed. When the corruption oracle is queried on a message m , the issued credential on m , i.e., the signature on m , is revealed to the adversary. When the transcript oracle is queried on a message m , the transcript of a credential ownership proof with prover input the signature on m and public input the public key and the message m is returned to the adversary. When the prover oracle is queried on a message m , a session identifier s , and an incoming message M_{in} , the new prover clone identified by (m, s) is either replaced with the current active prover clone (in the case of an

active attack) or added to the set of current active prover clones (in the case of a concurrent attack); then the prover clone code is run on input the signature on m and the incoming message M_{in} and the output of the prover clone, outgoing message M_{out} , is returned to the adversary. At some point, the cheating verifier declares that the first phase is complete and decides on an impersonation target credential message.

In the second phase, a credential ownership proof between the cheating prover and an honest verifier is run on public input the public key and the target credential message. The cheating prover is also given access to the same oracles as did the cheating verifier during the first phase with the natural restriction that it cannot query the corruption or the prover oracles on the target credential message. The adversary is said to win if at the end of this interaction the honest verifier is convinced that the adversary is in possession of a signature on the target message, given the condition that the target credential message does not belong to a corrupted credential.

Formally, we define credential ownership proof security against impersonation under concurrent attacks as follows.

Definition 3.2.3 (Credential Ownership Proof Security against Impersonation) *For a credential ownership proof $S\text{-COP} = (P, V)$, associated with a signature $S = (\text{KeyGen}, \text{Sign}, \text{Verify})$, and an adversary $A = (CV, CP)$, we define the advantage of A in credential ownership proof impersonation under passive, active, and concurrent attacks as the probabilities below, where the experiments $\text{Expt}_{S\text{-COP}, A}^{\text{COP-IMP-ATK}}(k)$ for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles are defined as in Figure 3.1.*

$$\text{Adv}_{S\text{-COP}, A}^{\text{COP-IMP-ATK}}(k) \triangleq \Pr [\text{Expt}_{S\text{-COP}, A}^{\text{COP-IMP-ATK}}(k) = 1]$$

We say that COP is COP-IMP-ATK-secure for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ if the maximum advantage a polynomial-time adversary can get in an COP-IMP-ATK-attack is negligible in k .

Credential-Holder Protection. The COP-IMP-ATK security notions guarantee that an adversary eavesdropping the communications of or interacting with multiple credential-holders is not able to impersonate any of them. This property is reflected in the defined experiments if the adversary chooses to see the transcripts of communications or interact with the prover clone(s) holding the credential on the target message during the first phase of the attack. Indeed, in Section 3.2.2 we show how the above notions are generally stronger than notions of security in an earlier work by Baek et al. [BSS05] aiming to capture credential-holder protection.

Credential-Issuer Protection. The credential-issuer is protected in the definition as the adversary that can have the signatures on arbitrary messages of its choice and interact with arbitrary credential-holders of its choice, cannot even prove ownership of a *new* credential, let alone forging one. This property is reflected in the defined experiment if the adversary chooses not to interact with a clone holding the credential on the target message during the attack and the target message presents a new credential. We note that COP-IMP-ATK security implies existential unforgeability of the underlying signature scheme under chosen-message attacks (i.e., EUF-CMA-security). This is true because if the signature is not EUF-CMA-secure, the adversary will be able to forge a new message-signature pair (i.e., a new credential) by properly querying the initiation and corruption oracles (using the combination

<p>Experiment $\text{Expt}_{\text{S-COP,A}}^{\text{COP-IMP-ATK}}(k)$</p> <p>$PS, UC, CC, AC \leftarrow \emptyset$</p> <p>$(pk, sk) \leftarrow \text{KeyGen}(1^k)$</p> <p>If $\text{ATK} = \text{PA}$</p> <p> Then let \mathcal{O} denote $\mathcal{T}r$</p> <p> Else let \mathcal{O} denote $\mathcal{P}rov$</p> <p>$(m, St_A) \leftarrow \text{CV}(1^k, pk : \text{Init}, \text{Crpt}, \mathcal{O})$</p> <p>If $m \notin UC$ then return 0</p> <p>$AC \leftarrow \{m\}$</p> <p>$(\varepsilon, d) \leftarrow [\text{CP}(St_A : \text{Init}, \text{Crpt}, \mathcal{O}) \leftrightarrow \mathcal{V}](pk, m)$</p> <p>Return d</p> <hr/> <p>Oracle $\text{Init}(m)$</p> <p>If $m \in UC \cup CC \cup AC$ then return \perp</p> <p>$\sigma[m] \leftarrow \text{Sign}(sk, m)$</p> <p>$UC \leftarrow UC \cup \{m\}$</p> <p>Return 1</p>	<p>Oracle $\text{Crpt}(m)$</p> <p>If $m \notin UC \setminus AC$ then return \perp</p> <p>$CC \leftarrow CC \cup \{m\}$</p> <p>$UC \leftarrow UC \setminus \{m\}$</p> <p>Return $\sigma[m]$</p> <hr/> <p>Oracle $\mathcal{T}r(m)$</p> <p>If $m \notin UC$ then return \perp</p> <p>$Tr \leftarrow \text{Tr}[\mathcal{P}(\sigma[m]) \leftrightarrow \mathcal{V}](pk, m)$</p> <p>Return Tr</p> <hr/> <p>Oracle $\mathcal{P}rov(m, s, M_{\text{in}})$</p> <p>If $m \notin UC \setminus AC$ then return \perp</p> <p>If $(m, s) \notin PS$ then</p> <p> If $\text{ATK} = \text{AA}$ then $PS \leftarrow \{(m, s)\}$</p> <p> If $\text{ATK} = \text{CA}$</p> <p> Then $PS \leftarrow PS \cup \{(m, s)\}$</p> <p> $St_P[m, s] \leftarrow (\sigma[m]; \rho)$ for random ρ</p> <p> $(M_{\text{out}}, St_P[m, s]) \leftarrow \mathcal{P}(M_{\text{in}}, St_P[m, s])$</p> <p>Return M_{out}</p>
--	--

Figure 3.1: The experiments used to define credential ownership proof COP-IMP-ATK-security for $\text{ATK} \in \{\text{PA}, \text{AA}, \text{CA}\}$ and the relevant oracles. PS denotes the set of active prover sessions that, in the case of an active attack, is limited to only *one* active session at a time. ε denotes the empty string. s denotes the session identifier. UC , CC , and AC denote the sets of uncorrupted, corrupted, and attacked credentials, respectively.

as a signing oracle) in the first phase of the COP-IMP-CA attack, and successfully prove ownership of the credential in the second phase.

It is easy to see that a zero knowledge proof of knowledge of a member of the NP language $L_{(pk, m)} = \{\sigma : \text{Verify}(pk, m, \sigma)\}$ is a COP-IMP-CA-secure credential ownership proof given that the underlying signature is EUF-CMA-secure. To prove this, one can construct a signature forger out of a credential ownership proof impersonator as follows. Honest credential-holders can be simulated without knowing a signature in the first phase since the protocol is zero knowledge. In the second phase, a valid signature can be extracted from the cheating prover since the protocol is a proof of knowledge. The obtained signature constitutes a forgery for the signature scheme. In this chapter though, we seek more efficient ways to realize COP-IMP-ATK-secure, specially COP-IMP-CA-secure, credential ownership proofs.

3.2.2 Security Treatment of Baek et al.

In an attempt to formalize the same concept as our credential ownership proofs, Baek et al. define a *universal designated-verifier signature proof* UDVSP [BSS05] as a sextuple of polynomial-time algorithms that consists of three algorithms of a signature scheme and the extra three algorithms below. A *transform* algorithm that generates a transformed secret key and a transformed signature as follows:

$$(\tilde{\sigma}, \tilde{sk}) \leftarrow \text{Transform}(pk, \sigma) .$$

Let ε denote the empty string. A pair of algorithms, a prover and a verifier, forming the *interactive verification protocol*, that interact with each other as follows:

$$(\varepsilon, d) \leftarrow [\text{P}(\tilde{sk}) \leftrightarrow \text{V}](pk, \tilde{\sigma}, m) .$$

In addition to unforgeability against chosen message attacks, Baek et al. propose the following two notions of security for a universal designated-verifier signature proof.

Definition 3.2.4 (Security against Impersonation under Type- i Attack) *For a universal designated verifier signature proof UDVSP = (KeyGen, Sign, Verify, Transform, P, V) and an adversary A = (CV, CP), the advantage of A in impersonation under type- i for $i \in \{1, 2\}$ attack is defined as the probability below, where the experiment $\text{Expt}_{\text{UDVSP}, A}^{\text{IM-TYPE-}i}(k)$ for $i \in \{1, 2\}$ is defined as in Figure 3.2.*

$$\text{Adv}_{\text{UDVSP}, A}^{\text{IM-TYPE-}i}(k) \triangleq \Pr [\text{Expt}_{\text{UDVSP}, A}^{\text{IM-TYPE-}i}(k) = 1]$$

UDVSP is said to be IM-TYPE- i -secure for $i \in \{1, 2\}$ if the maximum advantage a polynomial-time adversary can get in an IM-TYPE- i -attack is negligible in k .

<p>Experiment $\text{Expt}_{\text{UDVSP}, A}^{\text{IM-TYPE-1}}(k)$</p> <p>$St_A \leftarrow \varepsilon$</p> <p>$(pk, sk) \leftarrow \text{KeyGen}(1^k)$</p> <p>$\sigma \leftarrow \text{Sign}(sk, m)$ for random m</p> <p>$(\tilde{\sigma}, \tilde{sk}) \leftarrow \text{Transform}(pk, \sigma)$</p> <p>For $i = 1$ to n do</p> <p> $(\varepsilon, St_A) \leftarrow [\text{P}(\tilde{sk}) \leftrightarrow \text{CV}(St_A)](pk, \tilde{\sigma}, m)$</p> <p>$(\varepsilon, d) \leftarrow [\text{CP}(St_A) \leftrightarrow \text{V}](pk, \tilde{\sigma}, m)$</p> <p>Return d</p>	<p>Experiment $\text{Expt}_{\text{UDVSP}, A}^{\text{IM-TYPE-2}}(k)$</p> <p>$(pk, sk) \leftarrow \text{KeyGen}(1^k)$</p> <p>$\sigma \leftarrow \text{Sign}(sk, m)$ for random m</p> <p>$(\tilde{\sigma}', \tilde{sk}') \leftarrow \text{CV}(pk, m)$</p> <p>$(\varepsilon, d) \leftarrow [\text{CP}(\tilde{sk}') \leftrightarrow \text{V}](pk, \tilde{\sigma}', m)$</p> <p>Return d</p>
--	--

Figure 3.2: The experiments used to define universal designated verifier signature proof IM-TYPE- i security for $i \in \{1, 2\}$ by Baek et al. ε denotes the empty string. n is polynomial in k .

Let us compare the type-1 and type-2 attacks. In a type-1 attack, the adversary has the extra ability to interact with a prover in the first phase of the attack, but is limited to mount its impersonation on an honestly-generated transformed signature $\tilde{\sigma}$ in the second phase. In contrast, a type-2 adversary does not get to interact with a prover in the first phase of the attack, but is allowed to mount its impersonation on a maliciously-generated transformed signature $\tilde{\sigma}'$ in the second phase. Let us define

a hypothetical type-3 adversary as an adversary that both gets to interact with a prover in the first phase of the attack and is allowed to mount its impersonation on a maliciously-generated transformed signature in the second phase. The relevant experiment for a type-3 attack is shown in Figure 3.3. Clearly, an IM-TYPE-3-secure universal designated verifier signature proof is both IM-TYPE-1-secure and IM-TYPE-2-secure.

Experiment $\text{Expt}_{\text{UDVSP},A}^{\text{IM-TYPE-3}}(k)$

$St_A \leftarrow \varepsilon$

$(pk, sk) \leftarrow \text{KeyGen}(1^k)$

$\sigma \leftarrow \text{Sign}(sk, m)$ for random m

$(\tilde{\sigma}, \tilde{sk}) \leftarrow \text{Transform}(pk, \sigma)$

For $i = 1$ to n do

$(\varepsilon, St_A) \leftarrow [P(\tilde{sk}) \leftrightarrow CV(St_A)](pk, \tilde{\sigma}, m)$

$(\tilde{\sigma}', \tilde{sk}') \leftarrow CV(pk, m)$

$(\varepsilon, d) \leftarrow [CP(\tilde{sk}', St_A) \leftrightarrow V](pk, \tilde{\sigma}', m)$

Return d

Figure 3.3: The experiments used to define the hypothetical IM-TYPE-3 security. ε denotes the empty string.

Let us now compare the IM-TYPE-3 security notion with our COP-IMP-ATK security notions. Type-3 attacks are inherently comparable to our active attacks since they also provide the adversary with the ability to interact with a prover in the first phase of the attack. Generally speaking, an adversary in a COP-IMP-AA attack gets to use more attack resources than it does so in a type-3 attack. In a COP-IMP-AA attack, the adversary has extra capabilities of corrupting credential-holders of its choice, interacting with credential-holders of its choice, and deciding on the credential text that it is going to impersonate. A type-3 adversary, in contrary, is limited to interact with a fixed random credential holder and has to impersonate the same credential holder to win. Hence, COP-IMP-AA security is generally a stronger notion of security than IM-TYPE-3 security. Thus, COP-IMP-AA security is generally a stronger notion of security than IM-TYPE-1 and IM-TYPE-2 security.

3.3 Generic Construction from IBE

It is known that a secure signature scheme can be constructed based on a secure *identity-based encryption* (IBE). In this section, we extend this construction and give a generic construction of a COP associated with the mentioned signature scheme based on any IBE scheme and prove it COP-IMP-CA-secure assuming *one-wayness* of the IBE under *chosen ciphertext attacks* (denoted by OWE-ID-CCA). OWE-ID-CCA is a new security notion for IBE schemes which is weaker than *indistinguishability* under *chosen ciphertext attacks* (i.e. IND-ID-CCA), a widely-accepted security notion for IBE schemes formalized in [BF01, BF03]. Our results in this section provides constructions for signature schemes and associated COPs provably-secure based on *standard assumptions* such as BDH, in the *standard model*

(i.e. not in the *random oracle model*).

3.3.1 IBE and Its Security

An identity-based encryption scheme $\text{IBE} = (\text{MKeyGen}, \text{UKeyGen}, \text{Encrypt}, \text{Decrypt})$, consists of four algorithms where

the *master key generation* algorithm MKeyGen takes input the security parameter k and returns the master public key mpk and the master key msk , denoted as $(mpk, msk) \leftarrow \text{MKeyGen}(k)$,

the *user key generation* algorithm UKeyGen is given input msk and an identity I and outputs the decryption key usk corresponding to I , denoted as $usk \leftarrow \text{UKeyGen}(msk, I)$,

the *encryption* algorithm Encrypt on input mpk , I , and some plaintext P outputs the ciphertext C , denoted as $C \leftarrow \text{Encrypt}(mpk, I, P)$, and finally,

the *decryption* algorithm Decrypt takes inputs the user secret key usk and C and outputs P , denoted as $P \leftarrow \text{Decrypt}(usk, C)$.

We introduce a new notion of security for IBE schemes in analogy with the notions in [BF01] and [BF03]. Boneh and Franklin define three notions of security for IBEs: one-wayness (under chosen plaintext attacks) $\text{OWE-ID}(-\text{CPA})$, indistinguishability under chosen plaintext attacks IND-ID-CPA , and indistinguishability under chosen ciphertext attacks IND-ID-CCA . The resources of the adversary are the same in the first two notions: having access to a corruption oracle (a.k.a. extraction oracle). On the other hand, the goal of the adversary is the same in the last two notions: distinguishing the ciphertexts of two chosen plaintexts. We introduce a new notion OWE-ID-CCA in which the resources of the adversary are the same as the last notion: having access to both corruption and decryption oracles, and the goal of the adversary is the same as the first notion: decrypting a challenge ciphertext. Table 3.1 shows how the new notion compares to the previous three. It is clear that the security level guaranteed by this notion is higher than $\text{OWE-ID}(-\text{CPA})$, but lower than IND-ID-CCA .

Table 3.1: Comparison of security notions for IBE with the new notion highlighted

		adversary resource(s)	
		corruption oracle	corruption and decryption oracles
adversary	distinguishing	IND-ID-CPA	IND-ID-CCA
goal	decrypting	$\text{OWE-ID}(-\text{CPA})$	OWE-ID-CCA

Definition 3.3.1 (Identity-based One-Wayness under Chosen Ciphertext Attack) For an identity-based encryption scheme $\text{IBE} = (\text{MKeyGen}, \text{UKeyGen}, \text{Encrypt}, \text{Decrypt})$ and an adversary A , we define the advantage of A in breaking the identity-based one-wayness under a chosen ciphertext attack as the probability below, where the experiment $\text{Expt}_{\text{IBE}, A}^{\text{OWE-ID-CCA}}(k)$ and the signing oracle are defined as in Figure 3.4.

$$\text{Adv}_{\text{IBE}, A}^{\text{OWE-ID-CCA}}(k) \triangleq \Pr [\text{Expt}_{\text{IBE}, A}^{\text{OWE-ID-CCA}}(k) = 1]$$

We say IBE is OWE-ID-CCA-secure if the maximum advantage a polynomial-time adversary can get in an OWE-ID-CCA-attack is negligible in k .

<p>Experiment $\text{Expt}_{\text{IBE}, \mathcal{A}}^{\text{OWE-ID-CCA}}(k)$</p> <p>$HU, CU, AU \leftarrow \emptyset$</p> <p>$(mpk, msk) \leftarrow \text{MKeyGen}(1^k)$</p> <p>$(I, St_A) \leftarrow \mathcal{A}(1^k, mpk : \text{Init}, \text{Dec}, \text{Crpt})$</p> <p>$AU \leftarrow \{I\}$</p> <p>$P \xleftarrow{\\$} PSp$</p> <p>$C \leftarrow \text{Encrypt}(mpk, I, P)$</p> <p>$R \leftarrow \mathcal{A}(St_A, C : \text{Init}, \text{Dec}, \text{Crpt})$</p> <p>If $I \in HU$ and $C \notin C[I]$ and $P = R$</p> <p style="padding-left: 20px;">Then return 1</p> <p style="padding-left: 20px;">Else return 0</p> <hr/> <p>Oracle $\text{Dec}(I, C)$</p> <p>If $I \notin HU \setminus AU$ then return \perp</p> <p>$C[I] \leftarrow C[I] \cup \{C\}$</p> <p>Return $\text{Decrypt}(usk[I], C)$</p>	<p>Oracle $\text{Init}(I)$</p> <p>If $I \in HU \cup CU \cup AU$ then return \perp</p> <p>$usk[I] \leftarrow \text{UKeyGen}(msk, I)$</p> <p>$C[I] \leftarrow \emptyset$</p> <p>$HU \leftarrow HU \cup \{I\}$</p> <p>Return 1</p> <hr/> <p>Oracle $\text{Crpt}(I)$</p> <p>If $I \notin HU \setminus AU$ then return \perp</p> <p>$CU \leftarrow CU \cup \{I\}$</p> <p>$HU \leftarrow HU \setminus \{I\}$</p> <p>Return $usk[I]$</p>
--	---

Figure 3.4: The experiments used to define identity-based encryption scheme OWE-ID-CCA-security and the relevant oracles. HU , CU , and AU denote the sets of honest, corrupted, and attacked users, respectively. PSp denotes the plaintext space. $M[I]$ denotes the set of messages queried to the signing oracle for identity I .

3.3.2 Naor Transform

Boneh and Franklin attribute the following observation to Naor [BF01, p. 226]: A secure signature scheme can be constructed from any secure IBE. This observation can be formalized as the following transform.

Definition 3.3.2 (Naor Transform) *Let $\text{IBE} = (\text{MKeyGen}, \text{UKeyGen}, \text{Encrypt}, \text{Decrypt})$ be an identity-based encryption scheme. The Naor transform of IBE is defined as the signature scheme $\text{Naor}(\text{IBE}) = (\text{KeyGen}, \text{Sign}, \text{Verify})$, where the key generation, signing, and verification algorithms are defined as in Figure 3.5.*

The security claim can be formalized as follows.

Theorem 3.3.1 *$\text{Naor}(\text{IBE})$ is EUF-CMA assuming that IBE is IND-ID-CCA.*

Algorithm $\text{KeyGen}(1^k)$ $(sk, pk) \leftarrow \text{MKeyGen}(1^k)$ Return (sk, pk)	Algorithm $\text{Sign}(sk, m)$ $\sigma \leftarrow \text{UKeyGen}(sk, m)$ Return σ	Algorithm $\text{Verify}(pk, m, \sigma)$ Pick a random P $C \leftarrow \text{Encrypt}(pk, m, P)$ $P' \leftarrow \text{Decrypt}(pk, \sigma, C)$ If $P = P'$ then return 1 else return 0
--	--	---

Figure 3.5: The Naor transform of an identity-based encryption scheme

3.3.3 IBE-Based COP

As one may notice, the verification algorithm in $\text{Naor}(\text{IBE})$ inherently has a challenge-response structure. We use this structure to define the IBE-based credential ownership proof IBE-COP as in Figure 3.6. In other words, the verifier sends the prover a challenge ciphertext, obtained using encryption of a random plaintext, and expects the prover to be able to decrypt it using the signature she knows and reply with a response equal to the plaintext. We prove that this construction is COP-IMP-CA-secure .

Protocol $[\text{IBE-COP.P}(\sigma) \leftrightarrow \text{IBE-COP.V}](pk, m)$ $St_P \leftarrow (\sigma, pk, m), \quad St_V \leftarrow (pk, m)$		
<hr/>		
Algorithm $\text{IBE-COP.P}(C, St_P) \leftarrow (C) \leftarrow$ $R \leftarrow \text{Decrypt}(pk, \sigma, C)$ Return R		Algorithm $\text{IBE-COP.V}(\varepsilon, St_V)$ $P \xleftarrow{\$} \{0, 1\}^*$ $C \leftarrow \text{Encrypt}(pk, m, P)$ $St_V \leftarrow P$ Return (C, St_V)
	$\neg(R) \rightarrow$	Algorithm $\text{IBE-COP.V}(R, St_V)$ $d \leftarrow (P = R)$ Return d

Figure 3.6: The identity-based encryption scheme based credential ownership proof

Theorem 3.3.2 *IBE-COP is a COP-IMP-CA-secure credential ownership proof if IBE is an OWE-ID-CCA identity-based encryption.*

Proof. We construct an OWE-ID-CCA adversary A_{IBE} attacking IBE from a COP-IMP-CA adversary $A = A.(CP, CV)$ attacking IBE-COP and show that A_{IBE} is able to mount a successful attack if A succeeds.

In the first phase of the OWE-ID-CCA attack, A_{IBE} is given the master public key of IBE and oracle

access to the initiation, corruption and decryption oracles. At the end of this phase, A_{IBE} is expected to decide on the identity it is going to attack. A_{IBE} does this running A as a subroutine.

A_{IBE} first runs $A.CV$ on the master public key given to it as input. While running, $A.CV$ will make three kinds of requests: initiation oracle queries, corruption oracle queries, and requests to interact with a clone of honest prover holding a signature on a message of $A.CV$'s choice. A_{IBE} answers the initiation and corruption oracle queries using the initiation and corruption oracles provided to it, as signing in IBE-COP correspond with user key generation in IBE.

When $A.CV$ asks to have an interaction with a signature holder clone on a message m_i , A_{IBE} sets $I_i \leftarrow m_i$ and waits for the first message of interaction. $A.CV$ will send an encrypted challenge and expect to receive the correct decryption. A_{IBE} answers to the challenge C_i using its decryption oracle, that is, it queries (I_i, C_i) to the decryption oracle and relays the oracle response back to $A.CV$. This way, $A.CV$ will receive what it expects, i.e. the decryption of C_i .

After $A.CV$ gathers enough information to be able to impersonate, it decides on a target message m , declaring that it wants to impersonate a prover holding a signature on the message m . A_{IBE} now sets $I \leftarrow m$, outputs I as the target identity it wishes to attack, and declares that the first phase of the OWE-ID-CCA attack is over.

In the second phase of the OWE-ID-CCA attack, A_{IBE} is given a challenge ciphertext C and oracle access to the initiation, corruption, and decryption oracles and is expected to be able to decrypt the challenge. To do so, A_{IBE} runs $A.CP$ in the second phase of the (simulated) COP-IMP-CA attack. At this stage, the successful $A.CP$ is supposed to be able to impersonate, i.e. to be able to properly respond to a challenge given to it. A_{IBE} simply sends the challenge ciphertext C to $A.CP$ and outputs the obtained response R . If $A.CP$ is successful in impersonation, this response must be the correct decryption of C , which in turn means that A_{IBE} will be successful in the OWE-ID-CCA attack.

A_{IBE} only queries the corruption oracle when A asks a signing oracle query. Therefore, the success probability of A_{IBE} is equal to that of A . Furthermore, the running times of A_{IBE} and A are equal. Hence, A_{IBE} is also poly-time and we have:

$$\text{Adv}_{IBE, A_{IBE}}^{\text{OWE-ID-CCA}}(k) = \text{Adv}_{IBE\text{-COP}, A}^{\text{COP-IMP-CA}}(k) .$$

Denoting the COP-IMP-CA adversary achieving the highest advantage by A^* and the corresponding OWE-ID-CCA adversary constructed from A^* by A_{IBE}^* we get

$$\begin{aligned} \max_{A \in \text{Poly}(k)} [\text{Adv}_{IBE\text{-COP}, A}^{\text{COP-IMP-CA}}(k)] &= \text{Adv}_{IBE\text{-COP}, A^*}^{\text{COP-IMP-CA}}(k) \\ &= \text{Adv}_{IBE, A_{IBE}^*}^{\text{OWE-ID-CCA}}(k) \\ &\leq \max_{A_{IBE} \in \text{Poly}(k)} [\text{Adv}_{IBE, A_{IBE}}^{\text{OWE-ID-CCA}}(k)] , \end{aligned}$$

which completes the proof. ■

We note that this theorem is also of theoretical interest. Waters [Wat05], constructed efficient IBE schemes secure under the standard BDH assumption and without random oracles. Combining this result with Theorem 3.3.2 implies that a secure signature with associated COP can be constructed in standard model, without requiring random oracles or strong assumptions (such as one-more RSA).

As noted before, COP-IMP-CA security implies EUF-CMA. Hence, we have the following as a corollary. Note that this result is stronger than the previous observation, which claimed EUF-CMA security for Naor(IBE) given that the underlying IBE is IND-ID-CCA.

Corollary 3.3.3 *Naor(IBE) is EUF-CMA assuming that IBE is OWE-ID-CCA.*

3.4 Equivalence with IBI

An *identity-based identification scheme (IBI)* is a scheme through which an entity can identify herself to a verifier who only knows the claimed identity and a public key of a key generation center (KGC). The widely-accepted framework of security for such schemes is security against *impersonation* under *passive*, *active*, or *concurrent* attacks defined in Section 2.5. Bellare, Namprempre, and Neven show that there exists a trivial general construction of IBI schemes with building blocks consisting of standard identification and signature schemes, called *certificate-based IBI* [BNN04]. They also prove that if the underlying standard identification and signature schemes are secure, then the resulted certificate-based IBI is also secure. Hence, IBIs achieving high levels of security such as security against impersonation under concurrent attacks (which we denote by ID-IMP-CA) can be constructed.

In independent work, Kurosawa and Heng [KH04] introduced a new general construction of IBI using signature schemes with honest-verifier zero-knowledge protocols for proof of knowledge of a signature on a mutually-known message. However, their IBI only achieves security against impersonation under *passive* attacks assuming that the underlying signature scheme is secure and the protocol is HVZK-PoK. Interestingly, in this section we show that security against impersonation under *concurrent* attacks (ID-IMP-CA) can be achieved if their construction is applied to a signature with an associated credential ownership proof (henceforth S+COP), instead of a signature with their requirements! Furthermore, we show that S+COPs and IBIs are actually equivalent, i.e. each of them can be employed instead of the other only by renaming the entities, algorithms and parameters in use. Such equivalence implies yet another generic construction for secure COPs.

The Equivalence between the Schemes. An S+COP is a scheme through which a credential-issuer generates signatures on messages and later on, a credential-holder proves to a verifier, who only knows the credential-issuer's public key, that she is in possession of a signature on a mutually-known message. Similarly, an IBI is a scheme through which a KGC generates user secret keys for user identities and later on, a user proves to a verifier, who only knows the KGC's master public key, that she is in possession of a user secret key of a mutually-known identity. From this simple comparison, the equivalence shown in Table 3.2 between the entities, algorithms and parameters in the two schemes, i.e. S+COP scheme $S.(\text{KeyGen}, \text{Sign}, \text{Verify}) + \text{S-COP}.(P, V)$ and IBI scheme $\text{IBI}.(\text{MKeyGen}, \text{UKeyGen}, (P, V))$ becomes apparent. Note that, similar to signatures, a user secret key is also publicly verifiable (at least by simulating the identification protocol).

We call the transform which uses Table 3.2 to rename entities, algorithms and parameters in a given S+COP to convert it to an IBI scheme, COP-2-IBI transform. The corresponding reverse transform is likewise denoted IBI-2-COP transform. In what follows, we show that if these transforms are applied

Table 3.2: Equivalence between S+COPs and IBIs

scheme	entity			algorithm				parameter			
S+COP	issuer	holder	verifier	KeyGen	Sign	P	V	pk	sk	σ	m
III	III	III	III	III	III	III	III	III	III	III	III
IBI	KGC	user	verifier	MKeyGen	UKeyGen	P	V	mpk	msk	usk	I

to secure input schemes, they will yield secure output schemes. This fact enables us to construct each of the schemes from an implementation of the other.

Theorem 3.4.1 *The scheme COP-2-IBI (S, S-COP) is an ID-IMP-CA-secure identity-based identification assuming that S-COP is a COP-IMP-CA-secure credential ownership proof, and vice versa, i.e. the construction IBI-2-COP (IBI) is an EUF-CMA signature with an associated COP-IMP-CA-secure credential ownership proof assuming that IBI is an ID-IMP-CA-secure identity-based encryption.*

Proof. Security in a S+COP scheme guarantees that no poly-time adversary is able to impersonate a credential-holder, even if it can have a signature on any message it wishes (i.e. corrupt any credential holder it wants) and can interact concurrently with clones of credential-holders on messages of its choice. Likewise, security in an IBI scheme guarantees that no poly-time adversary is able to impersonate a user, even if it can have the user secret key of any identity it wishes (i.e. corrupt any identity it wants) and can interact concurrently with clones of users with identities of its choice. Hence, the two security definitions are intuitively equivalent. Indeed, if the entities, algorithms, and parameters are properly renamed according to Table 3.2, then the definition of our COP-IMP-CA notion in Definition 3.2.3 and the ID-IMP-CA notion in Section 2.5 convert to each other. Hence, the two security notions are equivalent and this completes the proof. ■

Note that the second part of Theorem 3.4.1 particularly enables one to construct several COP-IMP-CA-secure COP protocols (plus several EUF-CMA signatures) out of the many ID-IMP-CA-secure IBI schemes proposed to date, based on a range of different computational assumptions. For a collection of provably-secure IBI schemes, please refer to [BNN04]. Another implication of this equivalence is the construction of secure IBI schemes from secure IBE schemes.

3.5 Efficient COP from GQ

The mentioned two generic constructions result in several COP protocols based on a range of different security assumptions. However, the cryptography involved in implementing IBE and IBI schemes is complex. For instance, a notable proportion of such schemes requires implementation of bilinear maps. In this section we show that the GQ identification scheme [GQ88] yields a COP-IMP-CA-secure credential ownership proof protocol (that we call RSA-COP) for the popular RSA-FDH signature [BR96]. Such a construction only exploits simple RSA cryptography and can be implemented efficiently. Particularly, RSA-COP can be easily integrated into credential systems already using the popular RSA-FDH signature to issue credentials. First we briefly review the GQ scheme and then prove that it yields a secure COP protocol. Finally, a simple comparison is shown between RSA-COP

and some ZK solutions to our motivating problem, in terms of computational and communicational complexity. The comparison provides clear justification why COPs are preferable to ZK solutions.

3.5.1 The GQ Identification Scheme

This scheme was proposed by Guillou and Quisquater [GQ88], and proved to have both the *honest verifier zero knowledge (HVZK)* and the *proof of knowledge (PoK)* properties. The scheme enables the prover to prove knowledge of x to the verifier such that $X = x^e \bmod N$ holds for some mutually-known $pk_{GQ} = (N, e, X)$. To identify herself, the prover first sends a *commitment* Y to the receiver which is then replied by a *challenge* c from the verifier. Finally, the prover answers with a *response* z . The verifier then makes the decision d by testing whether or not the equation $z^e = Y \cdot X^c \bmod N$ holds. The scheme is transcribed in Figure 3.7.

	<p>Protocol $[\text{GQ.P}(sk_{GQ}) \leftrightarrow \text{GQ.V}](pk_{GQ})$</p> <p>$St_P \leftarrow (sk_{GQ}, pk_{GQ})$</p> <p>$St_V \leftarrow pk_{GQ}$</p>						
<p>Algorithm $\text{GQ.KeyGen}(k)$</p> <p>$(N, e, d) \leftarrow \text{Gen}_{\text{RSA}}(k)$</p> <p>$x \xleftarrow{\\$} \mathbb{Z}_N^*, X \xleftarrow{N} x^e$</p> <p>$pk_{GQ} \leftarrow (N, e, X)$</p> <p>$sk_{GQ} \leftarrow (N, x)$</p> <p>Return (pk_{GQ}, sk_{GQ})</p>	<table> <tr> <td> <p>Algorithm $\text{GQ.P}(\varepsilon, St_P)$</p> <p>$y \xleftarrow{\\$} \mathbb{Z}_N^*, Y \xleftarrow{N} y^e$</p> <p>$St_P \leftarrow (sk_{GQ}, y)$</p> <p>Return (Y, St_P)</p> </td> <td>$\neg(Y) \rightarrow$</td> <td> <p>Algorithm $\text{GQ.V}(Y, St_V)$</p> <p>$c \xleftarrow{\\$} \{0, 1\}^{\ell(k)}$</p> <p>$St_V \leftarrow (pk_{GQ}, Y, c)$</p> <p>Return (c, St_V)</p> </td> </tr> <tr> <td> <p>Algorithm $\text{GQ.P}(c, St_P)$</p> <p>$z \xleftarrow{N} y \cdot x^c$</p> <p>Return z</p> </td> <td>$\leftarrow(c) \rightarrow$</td> <td> <p>Algorithm $\text{GQ.V}(z, St_V)$</p> <p>$d \leftarrow (z^e \stackrel{N}{=} Y \cdot X^c)$</p> <p>Return d</p> </td> </tr> </table>	<p>Algorithm $\text{GQ.P}(\varepsilon, St_P)$</p> <p>$y \xleftarrow{\\$} \mathbb{Z}_N^*, Y \xleftarrow{N} y^e$</p> <p>$St_P \leftarrow (sk_{GQ}, y)$</p> <p>Return (Y, St_P)</p>	$\neg(Y) \rightarrow$	<p>Algorithm $\text{GQ.V}(Y, St_V)$</p> <p>$c \xleftarrow{\\$} \{0, 1\}^{\ell(k)}$</p> <p>$St_V \leftarrow (pk_{GQ}, Y, c)$</p> <p>Return (c, St_V)</p>	<p>Algorithm $\text{GQ.P}(c, St_P)$</p> <p>$z \xleftarrow{N} y \cdot x^c$</p> <p>Return z</p>	$\leftarrow(c) \rightarrow$	<p>Algorithm $\text{GQ.V}(z, St_V)$</p> <p>$d \leftarrow (z^e \stackrel{N}{=} Y \cdot X^c)$</p> <p>Return d</p>
<p>Algorithm $\text{GQ.P}(\varepsilon, St_P)$</p> <p>$y \xleftarrow{\\$} \mathbb{Z}_N^*, Y \xleftarrow{N} y^e$</p> <p>$St_P \leftarrow (sk_{GQ}, y)$</p> <p>Return (Y, St_P)</p>	$\neg(Y) \rightarrow$	<p>Algorithm $\text{GQ.V}(Y, St_V)$</p> <p>$c \xleftarrow{\\$} \{0, 1\}^{\ell(k)}$</p> <p>$St_V \leftarrow (pk_{GQ}, Y, c)$</p> <p>Return (c, St_V)</p>					
<p>Algorithm $\text{GQ.P}(c, St_P)$</p> <p>$z \xleftarrow{N} y \cdot x^c$</p> <p>Return z</p>	$\leftarrow(c) \rightarrow$	<p>Algorithm $\text{GQ.V}(z, St_V)$</p> <p>$d \leftarrow (z^e \stackrel{N}{=} Y \cdot X^c)$</p> <p>Return d</p>					

Figure 3.7: The GQ identification scheme

Bellare and Palacio prove that this identification scheme is secure against concurrent impersonation attacks, i.e. IMP-CA-secure [BP02], provided that the *challenge length* is super-logarithmic and the *one-more RSA inversion problem* is hard (see Definition 2.2.2 on page 16). In particular, they prove the following theorem.

Theorem 3.5.1 *In the random oracle model, GQ is IMP-CA-secure if one-more RSA inversion problem is hard for moduli generated by Gen_{RSA} and the challenge length is super-logarithmic in the security parameter. Quantitatively speaking, we have*

$$\max_{A_{GQ} \in \text{Poly}(k)} \left[\text{Adv}_{GQ, A_{GQ}}^{\text{IMP-CA}}(k) \right] \leq 2^{-\ell(k)} + \sqrt{\max_{A_{RSA} \in \text{Poly}(k)} \left[\text{Adv}_{\text{Gen}_{\text{RSA}}, A_{RSA}}^{\text{RSA-OMI}}(k) \right]},$$

where ℓ is the challenge length in GQ.

3.5.2 RSA-FDH Credential Ownership Proof

The RSA-FDH signature scheme, based on the RSA system [RSA78], is proposed and proved existentially unforgeable under chosen message attack by Bellare and Rogaway [BR96]. Briefly, the scheme uses an RSA modulus generator Gen_{RSA} to generate keys, assigns a signature of the form $[H(m)]^d \bmod N$ to a message m , and verifies a candidate signature σ by checking whether or not $\sigma^e = H(m) \bmod N$. The complete transcription of the scheme comes in Figure 3.8. The scheme assumes oracle access to a *full-domain hash* function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$, where $N = pq$ is an RSA modulus. Here, Gen_{RSA} is an RSA modulus generator, which produces an RSA modulus N and two values e and d such that $e \cdot d = 1 \bmod \varphi(N)$, where φ is the Euler function.

<p>Algorithm RSA-FDH.KeyGen(k)</p> <p>$(N, e, d) \leftarrow \text{Gen}_{\text{RSA}}(k)$</p> <p>$pk \leftarrow (N, e)$</p> <p>$sk \leftarrow (N, d)$</p> <p>Return (pk, sk)</p>	<p>Algorithm RSA-FDH.Sign(sk, m)</p> <p>$\sigma \xleftarrow{N} [H(m)]^d$</p> <p>Return σ</p> <hr/> <p>Algorithm RSA-FDH.Verify(pk, m, σ)</p> <p>$b \leftarrow (\sigma^e \stackrel{N}{=} H(m))$</p> <p>Return b</p>
---	--

Figure 3.8: The RSA-FDH signature scheme

We define the COP protocol RSA-COP as follows.

[RSA-COP.P(σ) \leftrightarrow RSA-COP.V](pk, m)

$pk_{\text{GQ}} \leftarrow (N, e, H(m))$

$sk_{\text{GQ}} \leftarrow (N, \sigma)$

$b \leftarrow [\text{GQ.P}(sk_{\text{GQ}}) \leftrightarrow \text{GQ.V}](pk_{\text{GQ}})$

Return b

Note that in analogy with GQ, in RSA-COP the prover proves knowledge of a value σ to the verifier such that $H(m) = \sigma^e \bmod N$ holds for some mutually-known N , $H(m)$, and e .

Proving completeness of the protocol is straightforward: Completeness of GQ translates into the equation $\sigma^e = H(m) \bmod N$, which holds given the validity of the RSA-FDH signature. We prove security of the protocol against COP-IMP-CA attacks.

Theorem 3.5.2 *RSA-COP is COP-IMP-CA-secure in the random oracle model assuming that GQ is secure against concurrent impersonation attack. Quantitatively speaking, we have*

$$\max_{A \in \mathbb{P}\text{oly}(k)} [\text{Adv}_{\text{RSA-COP}, A}^{\text{COP-IMP-CA}}(k)] \leq O(q_s) \cdot \max_{A_{\text{GQ}} \in \mathbb{P}\text{oly}(k)} [\text{Adv}_{\text{GQ}, A_{\text{GQ}}}^{\text{IMP-CA}}(k)] ,$$

where q_s is the number of credentials the issuer signs.

Proof. We prove that if a successful COP-IMP-CA adversary A for RSA-COP exists, then, in the random oracle model, a successful IMP-CA adversary A_{GQ} for GQ can be constructed. Description of such a construction follows.

In the proof, we make use of Coron's method [Cor00]. This method can be used for many proofs in the random oracle model to improve the reduction by a factor of q_h , the number of random oracle queries. The idea is to embed the problem instance in many of the random oracle responses, instead of just one of them. This way, one loses the ability to answer other adversary queries if they involve an embedded hash value, but on the other hand, gains the possibility to succeed in the reduction in many cases where adversary's output involve an embedded hash value. Overall, the reduction success probability can be calculated and optimized with respect to the embedding probability. In many cases, such as this reduction, this leads to an improvement by a factor of q_h .

Our assumption, i.e. the existence of a successful COP-IMP-CA adversary A for RSA-COP, means that there exists a pair of algorithms $A = A.(CV, CP)$, which is able to carry out a successful COP-IMP-CA attack on RSA-COP protocol. A_{GQ} uses these algorithms to mount an IMP-CA attack on GQ. Besides, working in the random oracle model implies that A_{GQ} must also simulate hash oracle query responses.

In the first phase of the IMP-CA attack, A_{GQ} is given pk_{GQ} and can request to interact concurrently with different clones of honest prover $GQ.P$. In the second phase, A_{GQ} is supposed to play the role of a prover and convince an honest verifier $GQ.V$ in an interaction.

To answer new hash oracle queries, A_{GQ} does the following. To answer the i th new query m_i , it picks a random value r_i from \mathbb{Z}_N^* and answers with hash value $r_i^e \bmod N$ with probability p_0 and with hash value $X \cdot r_i^e \bmod N$ with probability $(1 - p_0)$. In the former case we say X is *not* embedded in the hash of m_i and in the latter case we say X is embedded in the hash of m_i . Note that p_0 is a fixed probability which will be determined later, X is a value obtained from parsing pk_{GQ} , and all repeated queries will be answered the same as was answered before.

In the first phase of the attack, A_{GQ} must play the role of a cheating verifier $A_{GQ}.CV$ to extract needed information out of concurrent interactions with clones of honest GQ prover. A_{GQ} does this using the cheating verifier $A.CV$ as a subroutine. Given the public key $pk_{GQ} = (N, e, X)$, A_{GQ} runs $A.CV$ on input $pk = (N, e)$. $A.CV$ will then adaptively make three kinds of requests: requests to have an arbitrary message signed using RSA-FDH (*Init* oracle queries) requests to have RSA-FDH signature on an arbitrary message (*Crpt* oracle queries), and requests to interact concurrently with clones of RSA-COP prover. We describe below how to properly respond to these requests (*Prov* oracle queries).

On an *Init* oracle query followed by a *Crpt* oracle query m_i , A_{GQ} is only able to answer the queries, in the hash of which X is not embedded, with simulated signature r_i (because $\sigma = [H(m_i)]^d = (r_i^e)^d = r_i \bmod N$). Otherwise, i.e. if X is embedded in the hash of m_i , A_{GQ} will not be able to answer the query and fails as a result. If the *Init* oracle query is not followed by a *Crpt* oracle query on the same message, A_{GQ} just ignores it.

On an *Prov* oracle query (m, sid, M_{in}) , A_{GQ} simulates the interaction as follows. First, it queries the hash oracle on m , and then, distinguishes the following two cases:

1. If X is not embedded in the hash of m (i.e. $H(m) = r^e \bmod N$), A_{GQ} has an easy work ahead. The signature on m is r (because $\sigma = [H(m)]^d = (r^e)^d = r \bmod N$). So A_{GQ} can play the role

of a prover in possession of the signature by just running the algorithm RSA-COP.P.

2. On the other hand, if X is embedded in the hash of m (i.e. $H(m) = X \cdot r^e \bmod N$), A_{GQ} uses its ability to request interaction with $GQ.P$ to simulate the interaction with an honest RSA-COP prover, as follows. If (m, sid) is new (i.e. $A.CV$ is asking for the clone to begin the interaction), A_{GQ} issues an interaction query $(\varepsilon, m||sid)$ to its GQ interaction oracle and receives a Y as response. It simply outputs Y as the response to $A.CV$'s query. On the other hand, if (m, sid) is not new (i.e. $A.CV$ is asking for the clone to respond to a challenge $c = M_{in}$), A_{GQ} issues an interaction query $(M_{in}, m||sid)$ to its GQ interaction oracle and receives a z as response, such that

$$z^e = Y \cdot X^c \bmod N .$$

A_{GQ} then outputs $\zeta \leftarrow r_i^c \cdot z \bmod N$ as the response to $A.CV$'s query. Using the above equation and considering the fact that $H(m) = X \cdot r^e \bmod N$ we will have:

$$\zeta^e = r^{ce} \cdot z^e = r^{ce} \cdot Y \cdot X^c = Y \cdot (X \cdot r^e)^c = Y \cdot [H(m)]^c \bmod N ,$$

which means that ζ is the convincing response with respect to the commitment Y given previously to $A.CV$ by the clone with ID (m, sid) and the challenge $c = M_{in}$ received from $A.CV$. This, in turn means that A_{GQ} has simulated the interaction for $A.CV$ correctly.

A_{GQ} continues to simulate the responses to the oracle queries of $A.CV$ as above until at some point $A.CV$ halts and outputs a pair (m, St_A) . A_{GQ} will be able to impersonate the GQ prover in the next phase if X is embedded in the hash value of m and fails otherwise.

In the second phase, A_{GQ} must play the role of a GQ prover and convince an honest GQ verifier that it knows the GQ secret key. To achieve this goal, A_{GQ} uses $A.CP$ as a subroutine. A_{GQ} runs $A.CP$ on input St_A , simulating $A.CP$'s initiation, corruption, and interaction oracle queries as in phase one. At some point, $A.CP$ outputs a commitment Y . A_{GQ} simply sends Y as the first message to the honest verifier. The honest verifier then chooses a challenge c and sends it back. A_{GQ} relays this c as challenge to $A.CP$. Then $A.CP$ will output a ζ such that:

$$\zeta^e = Y \cdot [H(m)]^c \bmod N .$$

A_{GQ} calculates $z \leftarrow \zeta / r^c \bmod N$ and sends it to the honest verifier as the final message. Considering that X is embedded in the hash value of m (i.e. $H(m) = X \cdot r^e \bmod N$), We have:

$$z^e = \frac{\zeta^e}{r^{ce}} = \frac{Y \cdot [H(m)]^c}{r^{ce}} = \frac{Y \cdot (X \cdot r^e)^c}{r^{ce}} = Y \cdot X^c \bmod N ,$$

which means that the honest verifier will be convinced with the response z . Hence, whenever $A.CV$ selects an m , in hash of which X is embedded, A_{GQ} will be able to impersonate.

It can be easily seen that if A is a polynomial time algorithm, then so is A_{GQ} . In fact, A_{GQ} 's running time is equal to that of A plus at most an exponentiation for each hash, sign, and interaction query. Moreover, the success probability of the constructed adversary can be calculated as follows. A_{GQ} succeeds if A never asks a sign oracle query on a message in the hash of which X is embedded, chooses the target message m such that in the hash of it X is embedded, and succeeds in impersonating a COP prover. The first condition happens with probability at least $p_0^{q_s}$, where q_s is the total number

of issued credentials (which is in turn bigger than the total number of signing oracle queries). The second condition is met with probability at least $(1 - p_0)$, which in turn leads us to the equation:

$$\text{Adv}_{\text{GQ}, \text{A}_{\text{GQ}}}^{\text{IMP-CA}}(k) \geq p_0^{q_s} \cdot (1 - p_0) \cdot \text{Adv}_{\text{RSA-COP}, \text{A}}^{\text{COP-IMP-CA}}(k) .$$

The optimum p_0 which maximizes the success probability of A_{GQ} is then calculated as

$$p_0^* = 1 - \frac{1}{q_s + 1} ,$$

which in turn yields the following equation:

$$\text{Adv}_{\text{GQ}, \text{A}_{\text{GQ}}}^{\text{IMP-CA}}(k) \geq \frac{1}{q_s} \cdot \left(1 - \frac{1}{q_s + 1}\right)^{q_s + 1} \cdot \text{Adv}_{\text{RSA-COP}, \text{A}}^{\text{COP-IMP-CA}}(k) .$$

Considering the fact that

$$\frac{q_s}{\left(1 - \frac{1}{q_s + 1}\right)^{q_s + 1}} = O(q_s) ,$$

we will obtain:

$$\text{Adv}_{\text{RSA-COP}, \text{A}}^{\text{COP-IMP-CA}}(k) \leq O(q_s) \cdot \text{Adv}_{\text{GQ}, \text{A}_{\text{GQ}}}^{\text{IMP-CA}}(k) .$$

We have shown that for any COP adversary A , a GQ adversary A_{GQ} exists, such that the above equation holds. Denoting the COP adversary with the highest advantage by A^* and the corresponding constructed GQ adversary by A_{GQ}^* we get

$$\begin{aligned} \max_{\text{A} \in \mathbb{P}_{\text{Poly}}(k)} [\text{Adv}_{\text{RSA-COP}, \text{A}}^{\text{COP-IMP-CA}}(k)] &= \text{Adv}_{\text{RSA-COP}, \text{A}^*}^{\text{COP-IMP-CA}}(k) \\ &\leq O(q_s) \cdot \text{Adv}_{\text{GQ}, \text{A}_{\text{GQ}}^*}^{\text{IMP-CA}}(k) \\ &\leq O(q_s) \cdot \max_{\text{A}_{\text{GQ}} \in \mathbb{P}_{\text{Poly}}(k)} [\text{Adv}_{\text{GQ}, \text{A}_{\text{GQ}}}^{\text{IMP-CA}}(k)] , \end{aligned}$$

which proves the claimed bound. ■

Combining this theorem and Theorem 3.5.1, we get the following.

Corollary 3.5.3 *In the random oracle model, RSA-COP is COP-IMP-CA-secure if one-more RSA inversion problem is hard for moduli generated by Gen_{RSA} and the challenge length is super-logarithmic in the security parameter. Quantitatively speaking, we have*

$$\max_{\text{A} \in \mathbb{P}_{\text{Poly}}(k)} [\text{Adv}_{\text{RSA-COP}, \text{A}}^{\text{COP-IMP-CA}}(k)] \leq O(q_s) \cdot \left(2^{-\ell(k)} + \sqrt{\max_{\text{A}_{\text{RSA}} \in \mathbb{P}_{\text{Poly}}(k)} [\text{Adv}_{\text{Gen}_{\text{RSA}}, \text{A}_{\text{RSA}}}^{\text{RSA-OMI}}(k)]} \right) ,$$

where q_s is the number of credentials the issuer signs and ℓ is the challenge length in RSA-COP.

3.5.3 Efficiency of the Scheme

We have showed that GQ provides a COP-IMP-CA-secure credential ownership proof for RSA-FDH credentials. As mentioned before, GQ is only known to be HVZK. This can be considered as a clue that RSA-COP fulfills our initial motivation to design COP protocols that can be implemented more efficiently than ZK proofs. A thorough examination of the scheme reveals that this indeed *is* the case. In Table 3.3 RSA-COP is compared with some of the most efficient constructions of ZK for widely-used

Table 3.3: Comparison of RSA-COP Costs with other ZK Solutions

protocol	rounds	prover cost (group op.)	verifier cost (group op.)
ZK-PoK-RSA from [DK99]	5	5	5
ZK-PoK-DL from [DK99]	5	4	5
ZK-PoK-DL from [CDM00]	4	4	6
RSA-COP	3	2	2

cryptographic relations, to the best of our knowledge. The compared protocols are two ZK proofs of knowledge of discrete logarithm (denoted ZK-PoK-DL) and one ZK proof of knowledge of e th root (denoted ZK-PoK-RSA) from [DK99, CDM00]. As the table shows, ZK solution to our problem takes up to four rounds of interaction and costs the credential holder up to four group operations, while employing GQ as RSA-COP reduces the interaction rounds to three and credential holder cost to only two group operations. This property makes our solution desirable for light-weight implementations of credential systems while guaranteeing our formalized security requirements at the same time.

3.6 Concluding Remarks

In this chapter we formalized a new cryptographic protocol that we called credential ownership proof. A credential ownership proof is a protocol that enables a credential holder to virtually show their credential to a verifier without actually showing it. A secure credential ownership proof guarantees that the verifier is not able to impersonate the credential holder later in a credential ownership proof. This is a reasonable guarantee that is sufficient for many transactions. Credential ownership proofs are aimed to relax the security requirements of the existing zero knowledge methods. Zero knowledge proofs guarantee that no information other than holding a valid credential is conveyed to the verifier. However, credential ownership proofs only require that the ability to carry out a successful credential ownership proof is not transferred to the verifier. By relaxing the security requirement, credential ownership proofs are able to provide more efficient protocols in terms of both computational and communicational costs.

We showed generic constructions of credential ownership proofs based on identity-based encryption schemes and identity-based identification schemes. These provide a range of multiple constructions for credential ownership proofs in different settings and based on different assumptions. These include credential ownership proofs based on factoring, RSA, pairings, and discrete logarithms. Our IBE construction is of theoretical interest as it shows that credential ownership proofs are realizable in the Standard Model. Our IBI-COP equivalence is of theoretical interest since combined with the IBE construction it provides a generic construction for identity-based identification schemes from identity-based encryption schemes.

A desirable feature for credential ownership proofs is to have minimal communication cost. That is, to have non-interactive credential ownership proofs. This can be a future direction for research. The Fiat-Shamir transform (see Section 4.2.2) can be used to construct non-interactive credential ownership

proofs. However, the security of such schemes are only known to be provable in the Random Oracle Model. Hence, we note the design of non-interactive credential ownership proofs secure in the Standard Model as a future research direction.

Chapter 4

Generic Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures

4.1 Introduction

Digital signature is one of the main cryptographic primitives, first introduced in [DH76], realized in [RSA78], and formalized in [GMR88]. There have been many proposals for efficient and secure signature schemes and numerous extensions and variants to accommodate requirements of different applications. In this chapter, we focus on generic constructions of two important extensions of ordinary signatures, universal designated-verifier signatures (UDVS) and identity-based signatures (IBS), and their variants including multi-receiver UDVS and hierarchical IBS.

In a *designated-verifier signature (DVS) scheme*, the signature is only verifiable by a designated verifier, hence allowing the signer to sign a document in a non-transferable way: that is, the designated verifier is only able to convince himself and not anyone else about the validity of the signature. The main idea is that the signature can be constructed by both the signer and the verifier and so when a verifier receives such a signature, since he knows that he has not constructed it himself, he is convinced that it is produced by the signer. The verifier, however, cannot convince a third party by showing him what he has received, since it could have been generated by the verifier himself. A *universal designated-verifier signature (UDVS) scheme* allows the holder of a signature, for example the owner of a certificate or a signed credential, to prove the ownership of the signed document to a designated verifier in a non-transferable way: that is, the verifier cannot convince anyone else about the signature of the third party. DVS and UDVS play important roles in privacy preserving applications such as electronic voting (see e.g. [BT94]), secret handshakes [BDS⁺03], and fair exchange (see e.g. [GJM99]). Both above schemes can be generalized to multiple designated verifiers. A *(universal) multi designated-verifier signature ((U)MDVS)* is hence a scheme that produces designated signatures that exclusively convince members of a specific set of (designated) verifiers.

Identity based cryptography has the very attractive property that the public key of the signer is his identity and so the need for public key certificates is removed. In *identity-based signatures (IBS)*, there is an authority with a key pair: a master secret key and a master public key, who generates for each user a user secret key based on the user's identity. A user can use its user secret key to sign messages.

Signatures can be verified against the identity of the signer and the master public key. IBS schemes are generalized to *hierarchical identity-based signatures* (HIBS), to provide support for hierarchical identities. In a HIBS, the authority in possession of the master secret key issues secret keys for a top level of entities and then each entity is able to produce secret keys for lower level entities with the help of her secret key. The identity of any entity is composed of a chain of identities of the entities on the secret key issuing path from the authority to the entity herself. There have been numerous constructions for UDVS and IBS (see sections 4.4.3 and 4.5.2 for examples).

In this chapter we propose generic constructions of the above primitives from a large class of ordinary signatures. Generic constructions are important because they provide a deeper insight into the extended functionalities and their relationship with the basic functionality of digital signatures and allow a systematic approach for building more complex primitives from simpler ones. They may also result in new constructions; in this case, we obtain a more flexible construction for UDVS with strengthened security properties.

4.1.1 Our Contributions

Defining Classes of Signatures and Public Keys. We define a class \mathbb{C} of signature schemes that includes most of the widely used signatures including RSA, Schnorr, Cramer-Shoup, and BLS signatures (see Tables 4.2 and 4.3 for more). Intuitively, a signature generated by a scheme in this class can be converted in a reversible way to a pair that consists of, (a) a publicly simulatable part, and (b) a second part for which there exists an efficient protocol for proving its knowledge. We call this pair a *converted signature*. We show that the above requirements can be used to construct a protocol to prove knowledge of a signature on a known message by revealing the first part of the converted signature and proving knowledge of the second part. The protocols do not provide honest-verifier zero-knowledge property, since the first part of the converted signature is revealed during the protocol, but we show that the requirements are sufficient to prove the security of the constructions.

The class \mathbb{C} includes almost all widely used signature schemes. Tables 4.2 and 4.3 list some schemes in this class. We show that these schemes belong to \mathbb{C} and provide concrete protocols for signature conversion and proofs of knowledge of the second part of the converted signatures. In some cases, we provide two alternatives for conversion and proof of knowledge, which result in different constructions with trade-off between the computational cost and the output size.

We define a class \mathbb{K} of public and private key pairs that will be used for verifiers. A key pair in this class has the property that there exists an honest-verifier zero-knowledge proof of knowledge protocol for the secret key that corresponds to a public key. The class includes the public and private key pairs that are the same as the keys in the RSA cryptosystem, the GQ identification scheme keys, and key pairs with private key being a discrete logarithm of the public key.

A Generic Construction for UDVS and its Variants. We show how to construct a universal designated-verifier signature scheme from signatures in \mathbb{C} . We use a definition of security that includes the original security notions of *unforgeability* and *non-transferability privacy*, and also an adaptation of the notion of *non-delegatability* introduced in [LWB05] for UDVS. Non-delegatability in

DVS ensures that a signer cannot delegate her key to a third party and is a crucial security property in applications such as e-voting. Among all known constructions of UDVS only the construction in [HSMW06] provides this property. We prove that if the underlying signature scheme is in \mathbb{C} and the designated verifier keys are in \mathbb{K} , then the proposed generic construction of UDVS has the above security properties.

The construction of UDVS uses Jakobsson et al’s approach to providing verifier designation [JSI96]: “Instead of proving Θ , Alice proves the statement: *Either Θ is true, or I am Bob.*” In UDVS schemes, Alice wants to prove validity of her certificate to Bob. Hence, a natural construction of UDVS is a non-interactive proof of the following statement by Alice: “*Either my certificate is valid, or I am Bob.*” Such a signature can be constructed by applying the Fiat-Shamir transform to an honest-verifier zero-knowledge (HVZK) protocol for proof of knowledge of Alice’s certificate *or* Bob’s secret key. However, efficient HVZK protocols for proof of knowledge of signatures are only known for a small group of signature schemes. We show that although our protocols for proof of knowledge of signatures in \mathbb{C} do not guarantee honest-verifier zero-knowledge property, they do provide security for UDVS schemes in the above construction. The proposed UDVS construction results in efficient schemes with computation cost and signature size comparable with the existing schemes. For a more detailed comparison see Table 4.4. There are, however, a couple of significant advantages that make the new construction of particular interest.

The construction is *generic*; that is, using the proposed techniques, most known signatures can be used to construct a UDVS. This is the first generic construction of UDVS.

The cryptographic keys of the signer and the verifier are decoupled. That is, signers and the verifiers can *independently* choose their cryptographic keys. The construction works for any choice of signer and verifier settings as long as the signature scheme is a member of class \mathbb{C} and the verifier key belongs to the class \mathbb{K} . All previous UDVS constructions have fixed and predefined combinations of signature schemes and verifier key pairs.

The above two properties enable our construction to be easily integrable in the existing systems in an on-the-fly basis. Almost any signature scheme used by a signer in the system can be extended to a UDVS without any new setup or rekeying needed.

The construction provides provable security, where security properties includes the traditional unforgeability and deniability, and also the new property of *non-delegatability*.

We prove designated signature unforgeability using the same assumption used for proving unforgeability of the original signature scheme. Providing the extra property of designated signature unforgeability, in addition to the normal signature unforgeability, usually requires stronger assumptions.

The construction can be extended to universal multi-designated-verifier signatures.

A Generic Construction for IBS and IBDVS and their Variants. We propose a construction of IBS schemes from a signature in Class \mathbb{C} and prove that the construction is secure against adaptive chosen message and chosen identity attacks. In our construction, a user’s secret key is a signature of the authority on the user’s identity. An identity-based signature is generated as follows: the user

constructs a proof of knowledge of her (converted) secret key (i.e. the authority's (converted) signature on her identity) and then transforms it into a signature on a message using the Fiat-Shamir transform.

The IBS construction has the following properties:

It allows *nesting* in the sense that a user can act as a new key generation authority and so can issue keys for other users. This effectively enables construction of *hierarchical* identity-based signatures from any signature scheme in Class \mathbb{C} .

The construction provides a framework that unifies descriptions and security proofs of a noticeable number of existing identity-based signature schemes. In fact, to our knowledge, all random oracle based IBS schemes in the literature are instances of this construction. In all such schemes, a signature scheme is used for the generation of the user's key, and a Fiat-Shamir proof of knowledge of the user's secret key is used to form the identity-based signature.

The proposed method results in new concrete constructions. Four new schemes that are obtained through this construction based on discrete logarithms, RSA, and pairings are shown in Table 4.6.

The security proofs for the constructed IBS is based on the same assumption used for proving unforgeability of the original signature scheme.

- We extend the construction to identity-based (multi) designated-verifier signatures from any signature in \mathbb{C} . These are the first generic constructions of identity-based (multi) designated-verifier signatures.

We note that all security proofs for the UDVS and IBS constructions, are in the Random Oracle Model (ROM) [BR93].

A Generic construction for identity-based ring signatures. Ring signatures provide signer anonymity by constructing a signature that can be generated by any member of an ad-hoc group that the signer chooses. The signer uses his secret key and the public keys of the members of the ad hoc group, to produce a ring signature that can be verified against the public keys of the members of the ring, while the identity of the signer remains information theoretically secure within the group. We give a generic construction of identity-based ring signatures from any signature in class \mathbb{C} .

Combined Constructions. We show that the techniques in the above generic constructions can be combined to obtain constructions with arbitrary combinations of the functionalities that U(M)DVS, (H)IBS, IB(M)DVS, and IBRS schemes offer. That is, we can construct the following generic schemes from any signature in Class \mathbb{C} : U(M)DVS, (H)IB(R)S, (H)IBU(M)DV(R)S, (H)IB(M)DV(R)S. Figure 4.1 shows how each of these generic constructions can be realized using our extensions / transformations. Security proofs of these schemes can be constructed by combining our proof techniques for appropriate extensions / transformations.

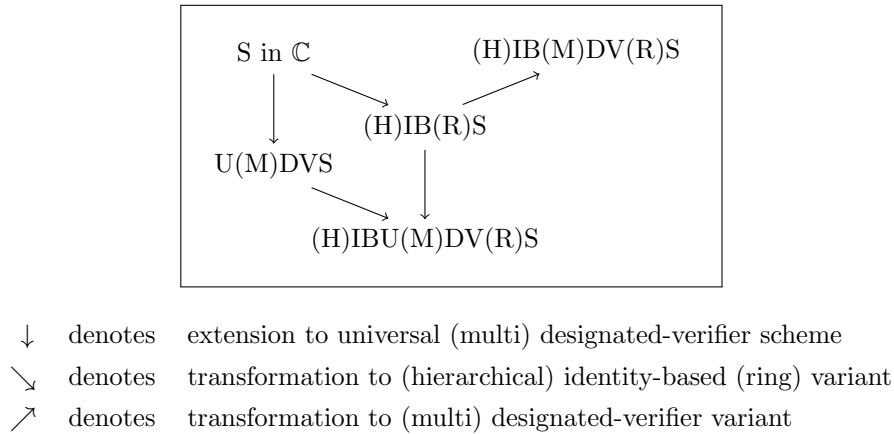


Figure 4.1: Schemes constructed generically in this chapter based on signature S in \mathbb{C}

4.1.2 Related Work

Designated verifier signatures (DVS) and universal designated-verifier signatures (UDVS) were proposed by Jakobsson et al. [JSI96] and Steinfeld et al. [SBWP03], respectively. Multi-designated-verifier signature was originally proposed by Jakobsson et al. [JSI96] and later formalized by Laguillaumie and Vergnaud [LV04b]. Universal multi-designated-verifier signatures was first proposed by Ng et al. [NSM05].

In [LWB05] Lipmaa et al. argued that the security definitions for DVS and UDVS schemes did not sufficiently capture the verifier-designation property and introduced the new security notion of *non-delegatability*. It has been shown that multiple DVS and UDVS schemes are delegatable (see e.g. [LWB05, LLP05, HSMW06]). None of the many UDVS schemes proposed to date, except a recent scheme of Huang et al. [HSMW06], has treated non-delegatability as a security requirement.

Identity-based cryptography was proposed by Shamir in [Sha84], where he also proposed an *identity-based signature* (IBS) scheme. Identity-based versions of DVS and UDVS were proposed in [SZM04] and [ZSMC05], respectively.

Ring signatures were proposed by Rivest et al. [RST01]. Identity-based versions were proposed in [ZK02].

All previous UDVS constructions only work for a specific combination of signature schemes and verifier key pairs. A recent work of Tso et al. [TGO⁺07] is the only exception in this regard. Although they require the signer to use (EC)DSA for signing, their construction gives the verifier the freedom to use either RSA-based or discrete-log-based keys.

The only other known non-delegatable UDVS scheme, due to Huang et al. [HSMW06], is in fact an instance of our construction.

Our constructions are close to Goldwasser and Waisbard's generic constructions of *designated conformer signatures* in [GW04]. They also use protocols for proof of knowledge of a signature as a building block of their constructions. Furthermore, they present such protocols for a number of signature schemes including Goldwasser-Micali-Rivest [GMR88], Gennaro-Halevi-Rabin [GHR99], and Cramer-Shoup [CS00] signatures. This shows that these signatures are in class \mathbb{C} .

A number of generic constructions of (H)IBS schemes from signatures is reported in the literature. One is the folklore *certification-based* construction that, as also noted in [GS02], constructs an HIBS scheme from any signature scheme. This construction is however inefficient as it produces long signatures. The construction in [KH04] has similar ideas to ours, but requires efficient zero-knowledge protocols for proof of knowledge of a signature, which is a restricting requirement. We show that zero-knowledge (and even honest-verifier zero-knowledge for that matter) is not strictly required and hence the construction we propose can be seen as a generalization of theirs. The construction in [BNN04] starts with an identification scheme as the building block and is different from the construction presented in this chapter.

Our identity-based signature can also be seen as the signature counterpart of *hidden credentials* [HBSO03]. In a hidden credential scheme, Alice encrypts a message in a way that Bob can only decrypt it if he has a certain credential from Chris, i.e. the credential acts as the private decryption key. In our identity-based signatures, Bob receives a signature which guarantees that an entity who has a certain credential from Chris has signed it, i.e. the credential acts as the private signing key.

Galindo et al. extended Bellare et al's generic IBS constructions to IBS schemes with additional properties, but as the authors also note [GHK06, p. 186], their approach cannot be extended to identity-based ring signatures and identity-based designated-verifier signatures.

4.2 Preliminaries

4.2.1 Proofs of Knowledge

Consider an *NP problem* P . The set of all the pairs (Ins, Sol) , each consisting of an *instance* Ins of P and its corresponding *solution* Sol , form a *relation* which we call an *NP relation*. Now, consider an NP relation Rel . Membership of this relation can be decided in polynomial time. Let Rel be the corresponding membership deciding algorithm. Then, a pair (Pub, Sec) belongs to Rel if and only if $Rel(Pub, Sec)$. Following the works of Camenisch and Stadler [CS97], we use the following notation for showing a protocol for *proof of knowledge* (See Definition 2.3.5 on page 20)

$$\text{PoK} \{Sec : Rel(Pub, Sec)\} \quad \text{or} \quad \text{PoK} \{Sec\} \quad \text{for short,}$$

where the prover proves knowledge of her secret Sec corresponding to a publicly known Pub , such that $(Pub, Sec) \in Rel$. The short version is used when the relation is clear from the context. Technically speaking, Sec is the private input to the prover algorithm and Pub is the public input of the protocol. We follow the convention that all the secret inputs are collectively denoted by Sec and shown before the colon ($:$) and all the remaining variables, functions, sets, etc. appearing after the colon are assumed to be the public inputs, collectively denoted by Pub .

Consider a three-move public-coin protocol written in the canonical form, as shown in Figure 2.4 on page 19, with prover's secret input $w = Sec$ and protocol public input $x = Pub$. The protocol is said to have the *special soundness* property (SpS from now on) as defined in [CDS94], if there also exists an algorithm that is able to *extract* the secret from two transcripts of the protocol with the same commitment and different challenges. A three-move public-coin protocol with both the honest-verifier

zero-knowledge (HVZK) (see Definition 2.3.4 on page 20) and SpS properties is usually called a Σ protocol. Examples of Σ protocols for proof of knowledge and the notation we use to denote them are shown in Table 4.1. The GQ protocol has also been discussed before in Figure 3.7 at page 44.

Table 4.1: Examples of Σ protocols for proof of knowledge

Protocol	Notation
GQ protocol [GQ88]	$\text{GQ} : \text{PoK}\{x : g^x = X\}$
Schnorr protocol [Sch91]	$\text{SchP} : \text{PoK}\{x : x^e = X \bmod N\}$
Feige-Fiat-Shamir protocol [FFS88]	$\text{FFS} : \text{PoK}\{x : x^2 = X \bmod N\}$
Okamoto protocol [Oka92]	$\text{OkaP} : \text{PoK}\{(x_1, x_2) : g_1^{x_1} g_2^{x_2} = X\}$
Cha-Cheon protocol [CC03, BNN04]	$\text{CCP} : \text{PoK}\{x : e(x, r) = e(s, t)\}$

We call a witness-indistinguishable proof of knowledge of one out of two quantities a *proof of disjunctive knowledge* of the two. In line with the above, we use the following notation to show such proofs: to show a protocol for proof of knowledge of a value Sec_1 such that $\text{Rel}_1(\text{Pub}_1, \text{Sec}_1)$ or a value Sec_2 such that $\text{Rel}_2(\text{Pub}_2, \text{Sec}_2)$, we use the notation

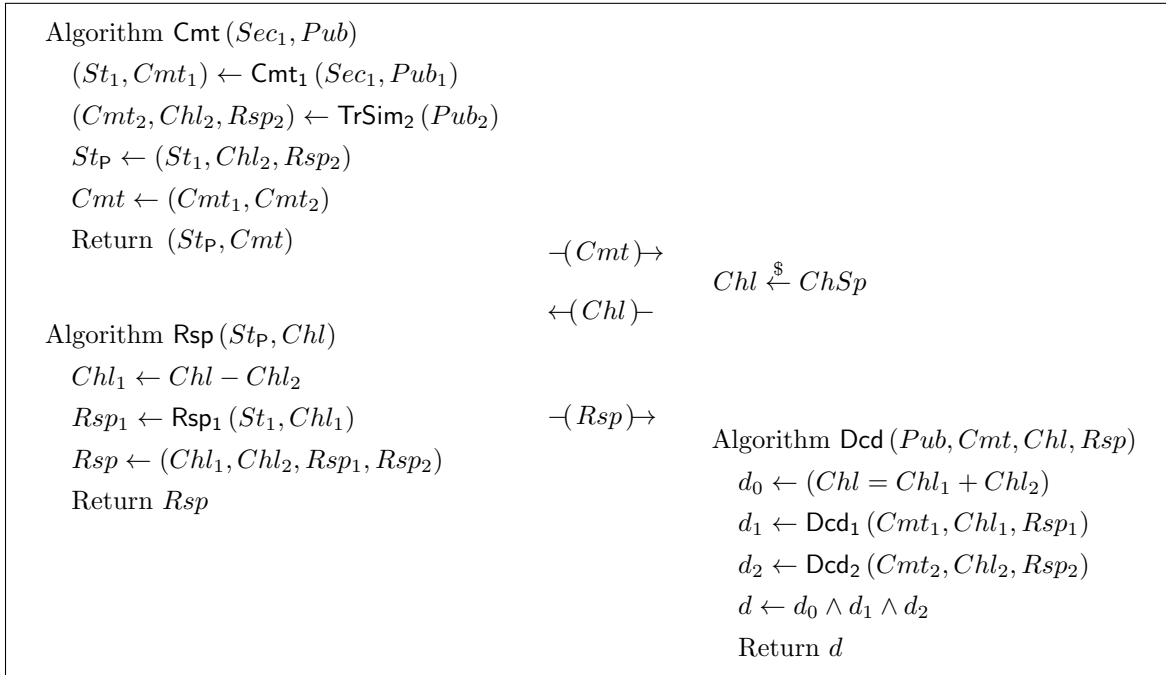
$$\text{PoK}\{(\text{Sec}_1, \text{Sec}_2) : \text{Rel}_1(\text{Pub}_1, \text{Sec}_1) \vee \text{Rel}_2(\text{Pub}_2, \text{Sec}_2)\} \text{ or } \text{PoK}\{\text{Sec}_1 \vee \text{Sec}_2\} \text{ for short.}$$

The short version is used when the relations are clear from the context. Consider two Σ protocols: one for proof of knowledge of Sec_1 corresponding to Pub_1 and another for proof of knowledge of Sec_2 corresponding to Pub_2 , both in the canonical form as in Figure 2.4 on page 19. Combining these two protocols, the Σ protocol for proof of knowledge of Sec_1 or Sec_2 corresponding to $\text{Pub} = (\text{Pub}_1, \text{Pub}_2)$ can be constructed as in Figure 4.2, assuming without loss of generality that the prover knows Sec_1 . This construction is due to Cramer et al. [CDS94]. Both HVZK and SpS properties are inherited by the constructed proof of disjunctive knowledge.

4.2.2 The Fiat-Shamir Transform

Fiat and Shamir proposed a method for transforming (interactive) three-move public-coin protocols into non-interactive schemes [FS86]. The idea is to replace the verifier with a hash function and the rationale behind it is that all the verifier does in such a protocol is providing an unpredictable challenge that can be mimicked by a Random Oracle hash function. This idea can be applied in two different ways, depending on what one includes in the hash function argument.

One way is to set the challenge as the hash of the concatenation of the public inputs and the commitment, i.e. $\text{Chl} \leftarrow H(\text{Pub} \parallel \text{Cmt})$. This way we get a *non-interactive proof of knowledge*. If such a transform is applied to the protocol in Figure 2.4 on page 19 using the Random Oracle hash function $H : \{0, 1\}^* \mapsto \text{ChSp}$, the resulting non-interactive proof scheme is as in Figure 4.3, with algorithms NIPoK and NIVoK for non-interactive proof and verification of knowledge, respectively. Here, π is a non-interactive proof that can be verified off-line and publicly. HVZK and SpS properties for non-interactive proofs are defined similarly to their counterparts for interactive proofs. Pointcheval and

Figure 4.2: A canonical Σ protocol for proof of disjunctive knowledge

Stern's *Forking Lemma* [PS00a] can be used to prove in the Random Oracle Model that the Fiat-Shamir construction has both the HVZK and SpS properties if the original interactive proof has the corresponding properties.

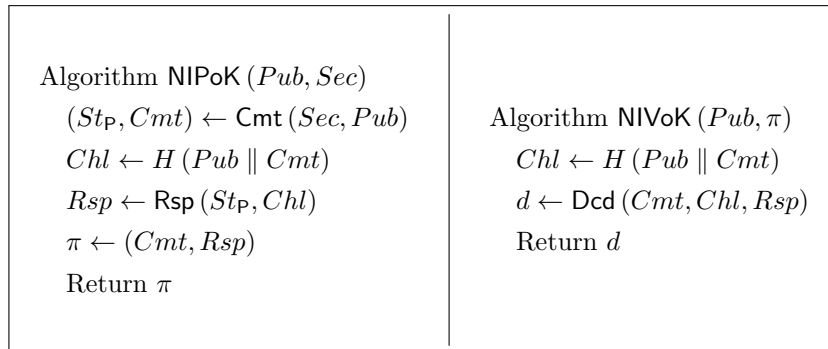


Figure 4.3: The non-interactive proof from applying Fiat-Shamir to the protocol in Figure 2.4

The other way of applying the Fiat-Shamir method is to set the challenge as the hash of the concatenation of the public inputs, the commitment, and an arbitrary message m , i.e. $Chl \leftarrow H(Pub \parallel Cmt \parallel m)$. This gives us a *signature scheme*. The resulting signature from applying such a transform to the protocol in Figure 2.4 using the Random Oracle hash function $H : \{0, 1\}^* \mapsto ChSp$, is as in Figure 4.4, with algorithms **Sign** and **Verify** for signing a message and verification of a candidate signature, respectively. Similarly, σ is a signature that can be verified publicly. The resulting signature scheme is existentially unforgeable under chosen message attack if the original protocol is a Σ protocol [PS00a]. Security of the signature can be also proved assuming other requirements (see e.g. [OO98]) or even weaker requirements on the protocol, namely IMP-PA-security as discussed in Theorem 2.4.1 at page 24. We do not get into those details since we are not going to use those results directly.

<p>Algorithm $\text{Sign}(Pub, Sec, m)$</p> <p>$(St_P, Cmt) \leftarrow \text{Cmt}(Sec, Pub)$</p> <p>$Chl \leftarrow H(Pub \parallel Cmt \parallel m)$</p> <p>$Rsp \leftarrow \text{Rsp}(St_P, Chl)$</p> <p>$\sigma \leftarrow (Cmt, Rsp)$</p> <p>Return σ</p>	<p>Algorithm $\text{Verify}(Pub, m, \sigma)$</p> <p>$Chl \leftarrow H(Pub \parallel Cmt \parallel m)$</p> <p>$d \leftarrow \text{Dcd}(Cmt, Chl, Rsp)$</p> <p>Return d</p>
--	---

Figure 4.4: The signature scheme from applying Fiat-Shamir to the protocol in Figure 2.4

For notational convenience, we use the term *signature of knowledge* (SoK) for both the NIPoK and Sign algorithms and the term *verification of knowledge* (VoK) for both the NIVoK and Verify algorithms, resulting from applying Fiat-Shamir transform to a Σ protocol as mentioned above. Assuming the original protocol to be $\text{PoK}\{Sec : \text{Rel}(Pub, Sec)\}$, we define SoK and VoK as follows:

$$\begin{aligned}
\text{SoK}\{Sec : \text{Rel}(Pub, Sec)\} &\triangleq \text{NIPoK}(Pub, Sec), \\
\text{VoK}\{Sec : \text{Rel}(Pub, Sec)\}(\pi) &\triangleq \text{NIVoK}(Pub, \pi), \\
\text{SoK}\{Sec : \text{Rel}(Pub, Sec)\}(m) &\triangleq \text{Sign}(Pub, Sec, m), \quad \text{and} \\
\text{VoK}\{Sec : \text{Rel}(Pub, Sec)\}(m, \sigma) &\triangleq \text{Verify}(Pub, m, \sigma).
\end{aligned}$$

4.2.3 On Public-Private Key Pairs

Key pairs are usually generated via a *key generation* algorithm KeyGen that takes a security parameter as input and outputs the key pair. It must be hard to compute the secret key corresponding to a given public key. We call the hard problem of computing the secret key for a given public key for a key pair the *underlying problem* of that key pair. Each public key is an *instance* of the underlying problem and the corresponding secret key is the corresponding *solution*. If key pairs are poly-time verifiable, i.e. one can efficiently verify if a given secret key corresponds to a given public key, the key generation algorithm KeyGen defines an NP relation KeyPair consisting of all the possible key pairs, i.e.

$$\text{KeyPair}_k = \{(pk, sk) : (pk, sk) \leftarrow \text{KeyGen}(k)\}.$$

We are interested in key pairs for which there exists a Σ protocol to prove knowledge of a secret key corresponding to a given public key. Let us call the set of these key pairs \mathbb{K} . A Σ protocol for a key pair in \mathbb{K} , omitting the security parameter (where it is clear from the context), can be shown as

$$\text{PoK}\{sk : \text{KeyPair}(pk, sk)\}.$$

Some key pairs that have Σ protocols as above are listed in the following. These include popular key pairs like the ones of the GQ identification scheme, discrete-log-based key pairs, and key pairs of the RSA cryptosystem. We use the term *key pair family* to refer to these different families of keys. For instance, we denote the keys for the GQ identification scheme by the term ‘GQ key pair family’.

There are quite a few different families of key pairs used in cryptographic schemes. Three of the simplest and most popular families are RSA family, GQ family, and DL family key pairs. The key generation algorithms for these families of keys are shown in Figure 4.5. The RSAGen and DLGen algorithms are respectively the *prime exponent RSA parameter generator* and the *DL parameter generator* algorithms that generate system parameters with respect to the security parameter taken as input.

Algorithm RSAKeyGen(k) $(N, e, d) \leftarrow \text{RSAGen}(k)$ $sk \leftarrow d$ $pk \leftarrow (N, e, d)$ Return (pk, sk)	Algorithm GQKeyGen(k) $(N, e, d) \leftarrow \text{RSAGen}(k)$ $sk \xleftarrow{\$} \mathbb{Z}_N^*; \quad X \xleftarrow{N} sk^e$ $pk \leftarrow (N, e, X)$ Return (pk, sk)	Algorithm DLKeyGen(k) $(p, g) \leftarrow \text{DLGen}(k)$ $sk \xleftarrow{\$} \mathbb{Z}_p; \quad X \leftarrow g^{sk}$ $pk \leftarrow (p, g, X)$ Return (pk, sk)
--	--	--

Figure 4.5: RSA, GQ and DL family key generation algorithms

The underlying problem of the RSA family keys is finding the private exponent d of an RSA system corresponding to the values (N, e, d) . The authors are not aware of any direct Σ protocols for proof of knowledge of the private exponent d corresponding to (N, e, d) . However, Rivest, Shamir and Adleman [RSA78], based on a previous work by Miller [Mil75], prove that there exists a *probabilistic* polynomial-time equivalence between computing d and factoring N . Furthermore, Coron and May [CM07] improve this results and present a *deterministic* polynomial-time algorithm that on input (N, e, d) outputs the factors of N . These results show that the knowledge of the private exponent is equivalent to the knowledge of the factorization of N . Thus, instead of proving knowledge of d , one can use the existing Σ protocols for proof of knowledge of the factorization of N , for example the protocols by Poupard and Stern [PS00b]. Therefore, RSA keys belong to \mathbb{K} .

The underlying problem of GQ and DL key families are the RSA and DL problems, respectively. Knowledge of the secret key corresponding to a public key for these two families of keys can be proved via GQ and Schnorr protocols, respectively, which are both Σ protocols. So these two families of keys also belong to \mathbb{K} .

Note that (Ins, Sol) , (Pub, Sec) , and (pk, sk) are three ways of showing the same object, i.e. a member of an NP relation, depending on how we are looking at the pair. We use this intuition later to interchange notation between NP problems, proofs of knowledge and key pairs.

4.2.4 The Forking and Reset Lemmas

The *Forking Lemma* was originally proposed by Pointcheval and Stern [PS00a]. Recently, Bellare and Neven proposed a general version of the Forking Lemma in [BN06]. We use the results and formulations from the latter in our proofs. For completeness, we transcribe the General Forking Lemma in the following.

Lemma 4.2.1 (General Forking Lemma) *Let $q \geq 1$ and H be a set such that $|H| \geq 2$. Let A be*

a randomized algorithm that has two outputs, the first of which is an integer in $\{0, 1, \dots, q\}$. Let also Coins be the set of all possible coins for \mathbf{A} . We define the accepting probability of \mathbf{A} with respect to an input generator IG as follows

$$\text{acc} \triangleq \Pr \left[J \geq 1 : x \leftarrow \text{IG}; h_1, \dots, h_q \xleftarrow{\$} H; (J, \sigma) \leftarrow \mathbf{A}(x, h_1, \dots, h_q) \right] .$$

The forker algorithm $\mathbf{F}_\mathbf{A}$ is defined as follows

Algorithm $\mathbf{F}_\mathbf{A}(x)$
 $\rho \xleftarrow{\$} \text{Coins}; \quad h_1, \dots, h_q \xleftarrow{\$} H$
 $(J, \sigma) \leftarrow \mathbf{A}(x, h_1, \dots, h_q; \rho)$
 If $J = 0$ then return $(0, \varepsilon, \varepsilon)$
 $h'_J, \dots, h'_q \xleftarrow{\$} H$
 $(J', \sigma') \leftarrow \mathbf{A}(x, h_1, \dots, h_{J-1}, h'_J, \dots, h'_q; \rho)$
 If $(J = J' \text{ and } h_J \neq h'_J)$ then return $(1, \sigma, \sigma')$
 else return $(0, \varepsilon, \varepsilon)$

We also define the success probability of the forker $\mathbf{F}_\mathbf{A}$ with respect to an input generator IG as follows

$$\text{frk} \triangleq \Pr [b \geq 1 : x \leftarrow \text{IG}; (b, \sigma, \sigma') \leftarrow \mathbf{F}_\mathbf{A}(x)] .$$

Then we have

$$\text{frk} \geq \text{acc} \cdot \left(\frac{\text{acc}}{q} - \frac{1}{|H|} \right) .$$

The forker algorithm $\mathbf{F}_\mathbf{A}$ basically attempts to find two sets of random inputs of the form (h_1, \dots, h_q) that have their first $J - 1$ elements in common and \mathbf{A} accepts and outputs index J on both of them. That is, a bifurcation happens on the sequence of inputs h_i on index $J - 1$ as shown in Figure 4.6. In the case of signature schemes, this translates to two runs of a forger algorithm on two sets of random oracle query responses that have their first $J - 1$ responses in common and the forger succeeds and outputs a forgery involving the J -th response on both runs.

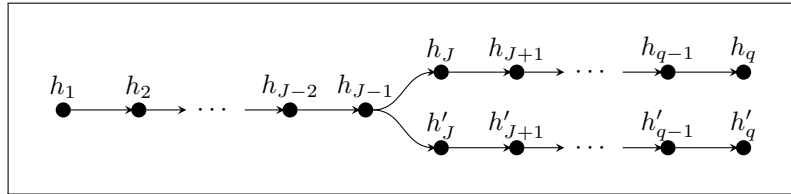


Figure 4.6: The bifurcation in the h_i inputs of the forked algorithm

The *Reset Lemma* was proposed by Bellare and Palacio [BP02]. We use this lemma in our proof. For completeness, we transcribe the Reset Lemma in the following.

Lemma 4.2.2 (Reset Lemma) *Let $\mathbf{P} = (\text{Cmt}, \text{Rsp})$ be a prover in a three-move public-coin canonical protocol, as shown in Figure 2.4, with a verifier represented by $\mathbf{V} = (\text{Chl}, \text{Dcd})$ with the challenge space ChSp , and let (w, x) be the private and public inputs to the protocol, respectively. Let $\text{acc}(w, x)$ be the probability that the verifier accepts in its interaction with the prover, namely the probability that*

the experiment *Experiment* $\text{Expt}_{\mathbf{P},\mathbf{V}}^{\text{ACC}}(w, x)$ in Figure 4.7 returns 1. Let $\text{res}(w, x)$ be the probability that the reset experiment *Experiment* $\text{Expt}_{\mathbf{P},\mathbf{V}}^{\text{RES}}(w, x)$ in Figure 4.7 returns 1. Then

$$\text{acc}(w, x) \leq \frac{1}{|\text{ChSp}|} + \sqrt{\text{res}(w, x)} .$$

<p>Experiment $\text{Expt}_{\mathbf{P},\mathbf{V}}^{\text{ACC}}(w, x)$</p> <p>$(St_{\mathbf{P}}, Cmt) \leftarrow \text{Cmt}(w, x)$</p> <p>$Chl \xleftarrow{\\$} \text{ChSp}$</p> <p>$Rsp \leftarrow \text{Rsp}(St_{\mathbf{P}}, Chl)$</p> <p>$d \leftarrow \text{Dcd}(x, Cmt, Chl, Rsp)$</p> <p>Return d</p>	<p>Experiment $\text{Expt}_{\mathbf{P},\mathbf{V}}^{\text{RES}}(w, x)$</p> <p>$(St_{\mathbf{P}}, Cmt) \leftarrow \text{Cmt}(w, x)$</p> <p>$Chl_1 \xleftarrow{\\$} \text{ChSp}$</p> <p>$Rsp_1 \leftarrow \text{Rsp}(St_{\mathbf{P}}, Chl_1)$</p> <p>$d_1 \leftarrow \text{Dcd}(x, Cmt, Chl_1, Rsp_1)$</p> <p>$Chl_2 \xleftarrow{\\$} \text{ChSp}$</p> <p>$Rsp_2 \leftarrow \text{Rsp}(St_{\mathbf{P}}, Chl_2)$</p> <p>$d_2 \leftarrow \text{Dcd}(x, Cmt, Chl_2, Rsp_2)$</p> <p>If $d_1 = d_2 = 1$ and $Chl_1 \neq Chl_2$ then return 1</p> <p>Else return 0</p>
--	--

Figure 4.7: The experiments used in the Reset Lemma.

The Reset Lemma basically upper bounds the probability that a (cheating) prover can convince the verifier to accept as a function of the probability that a certain experiment based on resetting the prover yields two accepting protocol transcripts. Note that in the reset experiment, the prover is only reset to the state *after* calculating the challenge. This yields two protocol transcripts with the same challenge, probably different challenges, and hence probably different responses.

4.3 Defining the Class \mathbb{C} of Signatures

In this section we define our class \mathbb{C} of signatures on which our constructions are based. The definition requires the signature to be convertible to a pair consisting of a simulatable part and a second part that is a pre-image of a value I under a function f , both of which determined by the simulatable part. This definition enables a holder of a signature to efficiently prove knowledge of a signature on a known message to a verifier as follows:

1. The holder converts the signature.
2. The holder reveals the simulatable part of the converted signature to the verifier.
3. The verifier determines I and f using the simulatable part.
4. The holder and the verifier carry out the protocol for proof of knowledge of the pre-image of I under f .

A similar property for signature schemes has been observed before in the literature, often referred to as the *reduction* of the proof of knowledge of a signature to a proof of knowledge of a pre-image

under a one-way function (see e.g. [ASW00, CD00, GMY06]). The property was used to tamper with the structure of specific signature schemes and has not been formalized comprehensively. Note that the fact that any NP relation has a Σ protocol [CDV06] provides protocols for proving knowledge of a signature for any signature scheme, but such protocols are not necessarily efficient enough for practice. We observe that even existence of a Σ protocol for a converted version of the signature is enough for our constructions. Such a protocol is not necessarily HVZK with respect to the signature since it reveals the simulatable part of the converted signature.

Let $SS = SS.(\text{KeyGen}, \text{Sign}, \text{Verify})$ be a provably-secure (standard) signature scheme. Security of the scheme, i.e. its existential unforgeability under chosen message attacks (EUF-CMA) [GMR88], is based on the hardness of an *underlying* problem denoted here by P_{SS} . Let us also denote by $PKSp$ and MSp the *public key space* (i.e. the set of all possible public keys) and the *message space* of a standard signature scheme, respectively. We define a class \mathbb{C} of standard signature schemes as follows.

Definition 4.3.1 (Class \mathbb{C} of Signatures) \mathbb{C} is the set of all signature schemes SS for which there exists a pair of algorithms, *Convert* and *Retrieve*, where *Convert* gets the public key pk , a message m , and a valid signature σ on the message as input and converts the signature to a pair $\tilde{\sigma} = (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$ called converted signature as follows:

$$\tilde{\sigma} = (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) \leftarrow \text{Convert}(pk, m, \sigma) \quad , \quad \text{such that:}$$

- there exists an algorithm *AuxSim* for simulating $\tilde{\sigma}_{\text{aux}}$ such that for every $pk \in PKSp$ and $m \in MSp$ the output of $\text{AuxSim}(pk, m)$ is (information-theoretically) indistinguishable from $\tilde{\sigma}_{\text{aux}}$,
- there exists an algorithm *Compute* that on input the public key pk , a message m , and $\tilde{\sigma}_{\text{aux}}$ computes a description of a one-way function $f(\cdot)$ and an I in the range of f , such that I is the image of $\tilde{\sigma}_{\text{pre}}$ under the one-way function f , i.e. for a converted signature the output of the following algorithm is **true**. *Valid* is the verification algorithm for the converted signature $\tilde{\sigma}$.

Algorithm *Valid* ($pk, m, \tilde{\sigma}$)
 $(f, I) \leftarrow \text{Compute}(pk, m, \tilde{\sigma}_{\text{aux}})$
 $d \leftarrow (f(\tilde{\sigma}_{\text{pre}}) = I)$
 Return d

- there exists a Σ protocol for proof of knowledge of a $\text{Sec} = \tilde{\sigma}_{\text{pre}}$ corresponding to a $\text{Pub} = (pk, m, \tilde{\sigma}_{\text{aux}})$ such that $\tilde{\sigma}$ is valid with respect to pk and m , i.e. there exist a Σ protocol for the following proof of knowledge

$$\text{PoK} \{ \tilde{\sigma}_{\text{pre}} : \text{Valid}(pk, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})) \} \quad ,$$

and for any candidate converted signature satisfying $\text{Valid}(pk, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))$, a valid signature on the message m can be retrieved via the *Retrieve* algorithm as follows:

$$\sigma \leftarrow \text{Retrieve}(pk, m, \tilde{\sigma}) \quad .$$

4.3.1 On Simulatability of Signature Schemes

We need another requirement on the signature scheme to be able to prove our schemes secure. We require that in the security proof of the signature scheme, two separate algorithms be identifiable: an algorithm that given an instance Ins of the underlying problem P_{SS} , is able to *simulate* for the adversary a public key and signatures on the messages of its choice, and a second algorithm that given a forgery by the adversary (resp. two forgeries on the same message for schemes with proof of unforgeability based on the Forking Lemma), is able to *calculate* the solution Sol to the problem instance. We call these two algorithms Sim and Cal , respectively. Since this is true for all the conventional signature schemes, we do not see it as a restricting requirement. We discuss this requirement in detail in this section.

We require that there exists a pair of algorithms, Sim and Cal , such that given an instance Ins of the underlying hard problem P_{SS} , Sim is able to *simulate* a public key for the signature scheme and signatures for arbitrary chosen messages with a noticeable probability, and given a pair (resp. two pairs) consisting of a new message and a signature (resp. two signatures) on the message, valid with respect to the simulated public key, Cal is able to *calculate* a solution Sol to the problem instance with a noticeable probability.

Intuitively, this property requires that it is possible to simulate the public key and signatures for chosen messages for the signature scheme, without knowledge of the secret key, with a sufficiently good probability, in a way that a forgery enables us to solve an instance of a hard problem. This simulation might take place in the Random Oracle Model though. Proofs of unforgeability for most of the signature schemes, are constructed in a folklore standard way by first simulating the attack environment for the adversary and then using the adversary's forgery to solve a hard problem. These two are the algorithms we are looking for. Note that for a proof in the ROM, the simulator must also answer A 's random oracle queries as well as its signing oracle queries.

Now consider two families of signatures depending on whether or not their security proof is based on the Forking Lemma. If the security proof of SS is *not* based on the Forking Lemma (*non-FL-based signature* from now on), then only one forgery is enough for the Cal algorithm to compute the solution Sol . However, if the security proof of SS is based on the Forking Lemma (*FL-based signature* from now on), then Cal needs two signatures on the same message to be able to calculate the solution to the hard problem. Let us denote by $Adv_{Sim(A)}(k)$ the probability that the Sim succeeds in simulating the attack environment for A and gets a *suitable* forgery (definition of suitable is case-dependent). Let us also denote by $Adv_{Cal(Sim)}(k)$ the probability that given one (respectively two for FL-based schemes) valid signature(s) on a message, Cal succeeds in computing the solution Sol for the problem instance Ins given to Sim .

A depiction of the mechanism of the proof for these two families of schemes is shown in Figure 4.8. Note that random oracle queries are not shown in this figure. As one can follow the order of events in Figure 4.8 from top to bottom, for a non-FL-based scheme, first the problem instance Ins is given to Cal as input. The public key pk and answers σ_i to signing oracle queries m_i are then simulated by Sim . The forgery (m, σ) which is output by the adversary A is then used by Cal to calculate a solution Sol for the problem instance. For FL-based schemes, a forker algorithm Frk is introduced which runs

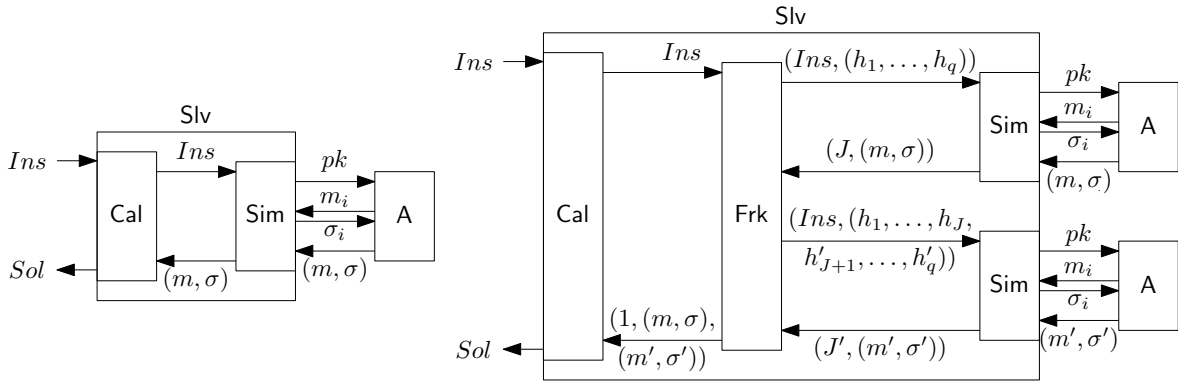


Figure 4.8: Mechanism of unforgeability proofs: non-FL-Based (left) and FL-Based signatures (right)

the simulator and the adversary twice, ordering the simulator to use different values as responses to the adversary's random oracle queries each time. Then the two signatures are given to the *Cal* that calculates and outputs *Sol*. For more details, see the General Forking Lemma in Section 4.2.4.

Let us also denote by *Slv* the combination of *Cal* and *Sim* for the case of non-FL-based signatures and the combination of *Cal*, *Frk*, and the two instances of *Sim* for the case of FL-based signatures. Using the notation defined above, we have the following results respectively for the non-FL-based and FL-based schemes:

$$\begin{aligned} \text{Adv}_{\text{Slv}}^{\text{Pss}}(k) &\geq \text{Adv}_{\text{Cal}(\text{Sim})}(k) \cdot \text{Adv}_{\text{Sim}(\text{A})}(k), \quad \text{and} \\ \text{Adv}_{\text{Slv}}^{\text{Pss}}(k) &\geq \text{Adv}_{\text{Cal}(\text{Sim})}(k) \cdot \text{Adv}_{\text{Sim}(\text{A})}(k) \cdot \left(\text{Adv}_{\text{Sim}(\text{A})}(k) - \frac{1}{|ChSp|} \right). \end{aligned}$$

4.3.2 Examples of Signatures in Class \mathbb{C}

Many of the signature schemes in use today fall in the class \mathbb{C} . Tables 4.2 and 4.3 list some examples and the corresponding algorithms for the above signatures and show why each of them belong to \mathbb{C} . RSA signature *RSA*, Rabin signature *Rab*, ElGamal signature *ElG*, and Schnorr signature *SchS* were proposed respectively in [RSA78], [Rab79], [ElG85], and [Sch91]. Security proofs for *RSA* and *Rab* were given in [BR93]. Security proofs for *SchS* and MEG (a modified version of *ElG*) were given in [PS00a]. Cramer-Shoup signature *CS*, BLS signature *BLS*, CL02 signature *CL02*, ZSS signature *ZSS*, BB signature *BB*, CL04 signature *CL04*, and Waters signature *Wat* were proposed and proved secure respectively in [CS00], [BLS01], [CL02], [ZSS04], [BB04b], [CL04], and [Wat05]. The table shows the respective signature anatomy and how a converted signature can be constructed for each scheme. All the pairing-based schemes, i.e. *BLS*, *ZSS*, *BB*, *CL04*, and *Wat*, admit to both *SchP* and *CCP* protocols for proving knowledge of converted signatures. Both methods of conversion are listed in the table for these schemes. To convert these schemes in the former method, the element z (r for *CL04*) is chosen randomly. The respective one-way function f and image I for each scheme is depicted next. Finally, a protocol for each scheme is shown. Using this protocol, the knowledge of a converted signature can be proved.

Table 4.2: Examples of signatures in \mathbb{C} , how to convert them, and how to prove knowledge of a converted signature. The scheme name comes in the first column followed by a brief description of the keys, the signature, and the verification equation in the upper section of the second column. The converted signature scheme comes in the lower section of the second column followed by the corresponding protocol for proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ in the third column.

Scheme	Keys	
	Signature: Verification	
	$\tilde{\sigma} = (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) : f(\tilde{\sigma}_{\text{pre}}) = I$	Protocol
RSA	$sk = d, pk = (N, e) : ed \stackrel{N}{=} 1$ $\sigma : \sigma^e \stackrel{N}{=} H(m)$	
	$\tilde{\sigma} = (\varepsilon, \sigma) : \tilde{\sigma}_{\text{pre}}^e \stackrel{N}{=} H(m)$	GQ
Rab	$sk = (p, q), pk = N : N = pq$ $\sigma : \sigma^2 \stackrel{N}{=} H(m)$	
	$\tilde{\sigma} = (\varepsilon, \sigma) : \tilde{\sigma}_{\text{pre}}^2 \stackrel{N}{=} H(m)$	FFS
MEG	$sk = x, pk = (g, h) : h = g^x$ $\sigma = (r, s) : g^{H(m r)} = h^r r^s$	
	$\tilde{\sigma} = (r, s) : \tilde{\sigma}_{\text{aux}}^{\tilde{\sigma}_{\text{pre}}} = g^{H(m \tilde{\sigma}_{\text{aux}})} / h^{\tilde{\sigma}_{\text{aux}}}$	SchP
SchS	$sk = x, pk = (g, h) : h = g^x$ $\sigma = (c, z) : c = H(g^z h^{-c} m)$	
	$\tilde{\sigma} = (g^z h^{-c}, z) : g^{\tilde{\sigma}_{\text{pre}}} = \tilde{\sigma}_{\text{aux}} h^{H(\tilde{\sigma}_{\text{aux}} m)}$	SchP
OkaS	$sk = (s_1, s_2), pk = (g_1, g_2, v) : v = g_1^{s_1} g_2^{s_2}$ $\sigma = (e, y_1, y_2) : e = H(g_1^{y_1} g_2^{y_2} v^e m)$	
	$\tilde{\sigma} = (g_1^{y_1} g_2^{y_2} v^e, (y_1, y_2)) : g_1^{\tilde{\sigma}_{\text{pre}1}} g_2^{\tilde{\sigma}_{\text{pre}2}} = v^{H(\tilde{\sigma}_{\text{aux}} m)} / \tilde{\sigma}_{\text{aux}}$	OkaP
CS	$sk = (p, q), pk = (n, h, x, e') : n = pq$ $\sigma = (e, y, y') : x \stackrel{n}{=} y^e h^{-H(y'^{e'} h^{-H(m)})}$	
	$\tilde{\sigma} = ((e, y'), y) : \tilde{\sigma}_{\text{pre}}^{\tilde{\sigma}_{\text{aux}1}} \stackrel{n}{=} x h^{H(\tilde{\sigma}_{\text{aux}2} h^{-H(m)})}$	GQ

As mentioned before, results in [GW04] show that both GMR [GMR88] and GHR [GHR99] signatures are also in \mathbb{C} . However, some schemes exist that do not seem to belong to \mathbb{C} , or at least do not seem to admit to efficient protocols, e.g. the PSS signature scheme from [BR96].

4.4 Universal Designated Verifier Signatures

In this section, we first review the definitions of the UDVS scheme and its security. Then we propose our generic construction of UDVS schemes from any signature scheme in \mathbb{C} and prove it secure.

4.4.1 Definition and Security

A UDVS is a signature scheme with an extra functionality: a holder of a signature can designate the signature to a particular verifier, using the verifier's public key. A UDVS can be described by adding

Table 4.3: Examples of signatures in \mathbb{C} , how to convert them, and how to prove knowledge of a converted signature (Continued). The scheme name comes in the first column followed by a brief description of the keys, the signature, and the verification equation in the upper section of the second column. The converted signature scheme comes in the lower section of the second column followed by the corresponding protocol for proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ in the third column.

Scheme	Keys	
	Signature: Verification	
	$\tilde{\sigma} = (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) : f(\tilde{\sigma}_{\text{pre}}) = I$	Protocol
BLS	$sk = x, pk = (g, y) : y = g^x$ $\sigma : e(\sigma, g) = e(H(m), y)$	
	$\tilde{\sigma} = (\sigma^z, z) : e(H(m), y)^{\tilde{\sigma}_{\text{pre}}} = e(\tilde{\sigma}_{\text{aux}}, g)$	SchP
	$\tilde{\sigma} = (\varepsilon, \sigma) : e(\tilde{\sigma}_{\text{pre}}, g) = e(H(m), y)$	CCP
CL02	$sk = (p, q), pk = (n, a, b, c) : n = pq$ $\sigma = (e, s, v) : v^e \stackrel{n}{=} a^{H(m)} b^s c$	
	$\tilde{\sigma} = ((e, s), v) : \tilde{\sigma}_{\text{pre}}^{\tilde{\sigma}_{\text{aux}1}} \stackrel{n}{=} a^{H(m)} b^{\tilde{\sigma}_{\text{aux}2}} c$	GQ
ZSS	$sk = x, pk = (g, y) : y = g^x$ $\sigma : e(g^{H(m)} y, \sigma) = e(g, g)$	
	$\tilde{\sigma} = (\sigma^z, z) : e(g, g)^{\tilde{\sigma}_{\text{pre}}} = e(g^{H(m)} y, \tilde{\sigma}_{\text{aux}})$	SchP
	$\tilde{\sigma} = (\varepsilon, \sigma) : e(g^{H(m)} y, \tilde{\sigma}_{\text{pre}}) = e(g, g)$	CCP
BB	$sk = (x, y), pk = (g, u_1, u_2) : (u_1, u_2) = (g^x, g^y)$ $\sigma = (\delta, l) : e(\delta, u_1 g^m u_2^l) = e(g, g)$	
	$\tilde{\sigma} = ((\delta^z, l), z) : e(g, g)^{\tilde{\sigma}_{\text{pre}}} = e(\tilde{\sigma}_{\text{aux}1}, u_1 g^m u_2^{\tilde{\sigma}_{\text{aux}2}})$	SchP
	$\tilde{\sigma} = (l, \delta) : e(\tilde{\sigma}_{\text{pre}}, u_1 g^m u_2^{\tilde{\sigma}_{\text{aux}}}) = e(g, g)$	CCP
CL04	$sk = (x, y), pk = (g, X, Y) : (X, Y) = (g^x, g^y)$ $\sigma = (a, b, c) : e(a, Y) = e(g, b) \wedge e(X, ab^{H(m)}) = e(g, c)$	
	$\tilde{\sigma} = ((a, b, c^r), r) : e(X, \tilde{\sigma}_{\text{aux}1} \tilde{\sigma}_{\text{aux}2}^{H(m)})^{\tilde{\sigma}_{\text{pre}}} = e(g, \tilde{\sigma}_{\text{aux}3})$	SchP
	$\tilde{\sigma} = ((a, b), c) : e(g, \tilde{\sigma}_{\text{pre}}) = e(X, \tilde{\sigma}_{\text{aux}1} \tilde{\sigma}_{\text{aux}2}^{H(m)})$	CCP
Wat	$sk = g_2^\alpha, pk = (g, g_1, g_2, W(\cdot)) : g_1 = g^\alpha$ $\sigma = (\sigma_1, \sigma_2) : e(\sigma_1, g) / e(\sigma_2, W(m)) = e(g_1, g_2)$	
	$\tilde{\sigma} = ((\sigma_1^z, \sigma_2), z) : [e(\tilde{\sigma}_{\text{aux}2}, W(m)) e(g_1, g_2)]^{\tilde{\sigma}_{\text{pre}}} = e(\tilde{\sigma}_{\text{aux}1}, g)$	SchP
	$\tilde{\sigma} = (\sigma_2, \sigma_1) : e(\tilde{\sigma}_{\text{pre}}, g) = e(\tilde{\sigma}_{\text{aux}}, W(m)) e(g_1, g_2)$	CCP

extra algorithms to the ones needed for description of the underlying signature scheme. Here, we briefly recall the definitions from Steinfeld et al. [SBWP03].

Definition 4.4.1 (Universal Designated-Verisier Signature) A universal designated verifier signature scheme is described by nine algorithms as follows:

CPGen is the common parameter generation algorithm that on input 1^k , where k is the security parameter, outputs a string consisting of common parameters cp publicly shared by all users,

SKeyGen (resp. VKeyGen) is the signer (resp. verifier) key generation algorithm that on input a

common parameter string cp , outputs a secret/public key-pair (sk_s, pk_s) (resp. (sk_v, pk_v)) for the signer (resp. verifier),

Sign and **PVer** are the Signing and public verification algorithms, where the former on input a signing secret key sk_s and a message m , outputs a signer's publicly-verifiable signature σ and the latter on input signer's public key pk_s and a pair (m, σ) consisting of a message and a (publicly-verifiable) signature, outputs a boolean verification decision d indicating the validity of the candidate signature with respect to the signer's public key and the message,

Desig and **DVer** are the Designation and designated signature verification algorithms, where the former on input a signer's public key pk_s , a verifier's public key pk_v , and a pair (m, σ) consisting of a message and a (publicly-verifiable) signature, outputs a designated(-verifier) signature $\hat{\sigma}$ and the latter on input a signer's public key pk_s , a verifier's secret key sk_v , and a pair $(m, \hat{\sigma})$ consisting of a message and a designated signature, outputs a boolean verification decision \hat{d} indicating the validity of the candidate designated signature with respect to the signer's public key, the verifier's secret key, and the message, and finally

(Reg, KRA) is the verifier key registration protocol between a verifier that runs **Reg** algorithm to register his public key and a key registration authority that runs **KRA** algorithm and at the end of protocol run the authority either accepts or rejects the registration.

Steinfeld et al. identified two security requirements for universal designated-verifier signature schemes: *designated signature unforgeability* and *non-transferability privacy*. We also consider a third property proposed by Lipmaa et al. called *non-delegatability*. Intuitively, these three properties guarantee the following. Unforgeability captures the inability of the adversary to forge designated signatures for new messages, even if it can have signatures on chosen messages and can verify chosen pairs of messages and designated signatures. Non-transferability privacy captures the inability of the designated verifier to produce evidence to convince a third party that a message has actually been signed by the signer. Finally, non-delegatability captures the inability of all but the signature holder and the designated verifier to generate designated signatures and hence the signature holder and designated verifier's inability to delegate their ability to generate designated signatures without revealing their corresponding secrets, i.e., the signature or the designated verifier secret key, respectively.

In the following, we define the security requirements for universal designated-verifier signature schemes. We review the definitions of strong designated signature unforgeability from [SWP04] and the definition of non-transferability privacy from [SBWP03]. Lipmaa et al. defined non-delegatability for designated verifier signatures [LWB05]. We adopt a similar definition for the universal designated verifier signature case.

Unforgeability. We use Steinfeld et al's definition of strong existential designated signature unforgeability of universal designated-verifier signature schemes under chosen message attacks, denoted here by DV-EUF-CMA-security. In the unforgeability game, the adversary is given the security parameter, the signer's public key, the verifier's public key, and can request to have a signature on any message of its choice and verify any pair consisting of a message and a designated signature of its choice against any signer's public key of its choice. The adversary's goal is to forge a designated

signature on a new message, i.e., on a message that has not been queried to the signing oracle. The formal definition comes in the following.

Definition 4.4.2 (Designated-Verifier Unforgeability) For a universal designated verifier signature scheme UDVS and an adversary A , we define the advantage of A in existential designated-verifier signature forgery under a chosen message attack as the probability below, where the experiment $\text{Expt}_{\text{UDVS},A}^{\text{DV-EUF-CMA}}(k)$ and relevant oracles are defined as in Figure 4.9.

$$\text{Adv}_{\text{UDVS},A}^{\text{DV-EUF-CMA}}(k) \triangleq \Pr [\text{Expt}_{\text{UDVS},A}^{\text{DV-EUF-CMA}}(k) = 1]$$

We say UDVS is DV-EUF-CMA-secure if the maximum advantage a polynomial-time adversary can get in a DV-EUF-CMA-attack is negligible in k .

$\text{Expt}_{A(\text{UDVS})}^{\text{DV-EUF-CMA}}(k)$ $M \leftarrow \emptyset$ $cp \leftarrow \text{CPGen}(1^k)$ $(sk_s, pk_s) \leftarrow \text{SKeyGen}(cp)$ $(sk_v, pk_v) \leftarrow \text{VKeyGen}(cp)$ $(m, \hat{\sigma}) \leftarrow A(1^k, pk_s, pk_v : \text{Sign}, \text{DVer})$ If $(\text{DVer}(pk_s, sk_v, m, \hat{\sigma}) = 1 \wedge m \notin M)$ Then return 1 else return 0	Oracle $\text{Sign}(m)$ $\sigma \leftarrow \text{Sign}(sk_s, m)$ $M \leftarrow M \cup \{m\}$ Return σ <hr/> Oracle $\text{DVer}(pk_s, m, \hat{\sigma})$ Return $\text{DVer}(pk_s, sk_v, m, \hat{\sigma})$
--	---

Figure 4.9: The experiment used to define universal designated verifier signature scheme DV-EUF-CMA-security and the relevant signing and designated signature verification oracles. M denotes the set of messages queried to the signing oracle by the adversary.

Non-Transferability Privacy. Steinfeld et al. have formalized this property and proposed a definition capturing the requirement that possessing a designated signature does not add to the computational ability of the designated verifier. In their formalization, they require that whatever a designated verifier who has been given a designated signature can leak to a third party (even at the expense of disclosing his secret key), he would have been able to leak without the designated signature.

To formalize non-transferability privacy, we consider a *cheating designated verifier* CDV and a *cheating third party* CTP cooperating with each other as an adversarial team. The cheating designated verifier is assumed to be able to have signatures on messages of its choice, register verifier keys of its choice, and communicate freely with the cheating third party. The cheating designated verifier can also choose a message to be signed by the signer, have the signature designated to any registered verifier public key of its choice, and have the designated signature. Its goal is to convince the cheating third party that the message is in fact signed by the signer. Now, let FS be a *forging strategy*. That is, FS is able to, given the CDV algorithm, forge designated signatures on the message in question without asking for the designated signatures. A second experiment is defined in which FS(CDV) interacts with the cheating third party. If the cheating third party cannot tell to which one among CDV or FS(CDV) it is communicating, non-transferability privacy is achieved, since whatever CDV is able to leak to a

cheating third party, can be leaked by $\text{FS}(\text{CDV})$ without access to any designated signatures. Formally, non-transferability privacy is defined as follows.

Definition 4.4.3 (Non-Transferability Privacy) *For a universal designated verifier signature scheme UDVS, an adversary $A = (\text{CDV}, \text{CTP})$, and a forgery strategy FS , the convincing measure $C_{\text{FS}}(A)$ is defined as the probability below, where the experiment $\text{Expt}_{\text{UDVS}, (\text{X}, \text{CTP})}^{\text{PR}}(k)$ for $\text{X} \in \{\text{CDV}, \text{FS}(\text{CDV})\}$ and relevant oracles are defined as in Figure 4.10.*

$$C_{\text{FS}}(A) \triangleq \left| \Pr \left[\text{Expt}_{\text{UDVS}, (\text{CDV}, \text{CTP})}^{\text{PR}}(k) = 1 \right] - \Pr \left[\text{Expt}_{\text{UDVS}, (\text{FS}(\text{CDV}), \text{CTP})}^{\text{PR}}(k) = 1 \right] \right|$$

UDVS is said to achieve computational non-transferability privacy if there exists an efficient forgery strategy FS such that the maximum convincing measure a polynomial-time adversary can achieve in a PR-attack is negligible in k . Furthermore, the scheme is said to achieve perfect non-transferability privacy if there exist an efficient forgery strategy FS such that the convincing measure in the PR-attack is always zero, regardless of the adversary's computational ability.

$\text{Expt}_{\text{UDVS}, (\text{X}, \text{CTP})}^{\text{PR}}(k)$ $M, VPK \leftarrow \emptyset$ $cp \leftarrow \text{CPGen}(1^k)$ $(sk_s, pk_s) \leftarrow \text{SKeyGen}(cp)$ Let \mathcal{O} denote $(\text{Sign}, \text{KRA})$ $((m, St_X), St_{\text{CTP}}) \leftarrow [\text{X}(\varepsilon : \mathcal{O}) \leftrightarrow \text{CTP}](1^k, pk_s)$ If $\text{X} = \text{CDV}$ then do $\sigma \leftarrow \text{Sign}(sk_s, m)$ Add Desig to \mathcal{O} $(\varepsilon, d) \leftarrow [\text{X}(St_X : \mathcal{O}) \leftrightarrow \text{CTP}(St_{\text{CTP}})](1^k, pk_s)$ If $m \in M$ then return \perp Return d	<div> Oracle $\text{Sign}(m)$ $\sigma \leftarrow \text{Sign}(sk_s, m)$ $M \leftarrow M \cup \{m\}$ Return σ </div> <hr/> <div> Oracle $\text{KRA}(pk_v)$ $(St_X, a) \leftarrow [\text{X}(St_X) \leftrightarrow \text{KRA}](pk_v)$ If $a = \text{accept}$ Then $VPK \leftarrow VPK \cup \{pk_v\}$ Return a </div> <hr/> <div> Oracle $\text{Desig}(pk_v)$ If $pk_v \in VPK$ Then return $\text{Desig}(pk_s, pk_v, m, \sigma)$ Else return \perp </div>
---	---

Figure 4.10: The experiments used to define universal designated verifier signature scheme PR-security for $\text{X} \in \{\text{CDV}, \text{FS}(\text{CDV})\}$ and the relevant signing, key registration, and signature designation oracles. M and VPK denote the set of messages queried to the signing oracle and the set of registered verifier public keys by the adversary. Note that the message m is signed and designated only for the cheating designated verifier CDV, and not for the forging strategy $\text{FS}(\text{CDV})$.

One can see that if designated signatures are simulatable by the verifier himself, then a designated signature adds no computational ability to the verifier and thus, we state and use the following lemma to prove our schemes secure.

Lemma 4.4.1 *A scheme UDVS achieves perfect non-transferability privacy if there exists an efficient forgery algorithm **Forge**, s.t. for any pairs (sk_s, pk_s) and (sk_v, pk_v) generated through key generation algorithms of UDVS and for any message m and any signature σ on m , the following two random variables have the same distribution:*

$$\text{Forge}(pk_s, sk_v, pk_v, m) \quad \text{and} \quad \text{Desig}(pk_s, pk_v, m, \sigma) \quad .$$

Note that there are two main differences between this lemma and Lemma 1 in [SBWP03, p. 531]. Firstly, their lemma is biconditional, but ours is not. Our lemma is a generalization of one direction of their lemma. They only state their lemma for deterministic designated signatures, but our lemma is stated for the general (possibly probabilistic) case.

Non-Delegatability. Lipmaa et al have defined the non-delegatability property for designated-verifier signatures. As they mention, their definition of κ -non-delegatability basically requires the designated signature to be a non-interactive *proof of knowledge* of the signer's or the designated verifier's secret key, with knowledge error κ as per Definition 2.3.5 on page 20. The reason behind such a definition is to guarantee that only the signer or the designated verifier are able to produce a designated signature, thus preventing them from being able to delegate their ability without revealing their secret key. In a UDVS scheme, we want only a person who holds a signature or the designated verifier to be able to produce a designated signature. Lipmaa et al's definition can be extended to the UDVS case as follows: κ -non-delegatability for UDVS schemes requires the designated signature to be a non-interactive proof of knowledge of a signature or the designated verifier's secret key, with knowledge error κ . This definition guarantees that only a signature holder or the designated verifier are able to construct valid designated signatures, and hence they are not able to delegate this ability to others without revealing the signature holder's certificate or the designated verifier's secret key.

To formalize the definition of non-delegatability, we model an entity that is able to produce designated signatures as a *designated-signature producer* oracle \mathcal{P} and denote by \mathcal{P}_m the oracle with input m , i.e. the oracle able to produce designated signatures on m . We propose the following definition.

Definition 4.4.4 (Non-Delegatability) *A UDVS scheme is said to be κ -non-delegatable if there exists a black-box knowledge extractor K that for any designated-signature producer oracle \mathcal{P} and any set of key pairs generated for the signer and the verifier through the common parameter and key generation algorithms of the scheme and for any message m , if \mathcal{P} produces a valid designated signature on m with probability $\epsilon > \kappa$, then on input m and access to the oracle \mathcal{P}_m , K outputs either a valid signature σ on m or the verifier's secret key sk_v in expected time at most τ such that $\tau \cdot (\epsilon - \kappa)$ is polynomially-bounded in the security parameter.*

We use an observation by Cramer et al. [CDM00, p. 359], that helps us simplify the non-delegatability proofs for our constructions.

Lemma 4.4.2 *A three-move public-coin protocol with SpS property and challenge space $ChSp$ is a proof of knowledge with knowledge error $\kappa = |ChSp|^{-1}$.*

The non-interactive version of this observation can be seen to hold in the Random Oracle Model using the Forking Lemma. That is, a Fiat-Shamir non-interactive proof of knowledge (i.e. our NIPoK) with

SpS property and challenge space $ChSp$ is a non-interactive κ -proof of knowledge in the the Random Oracle Model with knowledge error $\kappa = |ChSp|^{-1}$. Based on these observations, we propose the following lemma:

Lemma 4.4.3 *A scheme UDVS is κ -non-delegatable if a designated signature is a Fiat-Shamir non-interactive proof of knowledge of a signature or the secret key of the verifier, with SpS property and $|ChSp| \geq \frac{1}{\kappa}$.*

4.4.2 Generic Construction of UDVS And Its Security

We show how to extend any signature scheme in class \mathbb{C} to a universal designated verifier signature, by combining it with a key pair family for the verifier in \mathbb{K} . We use the building blocks we introduced before, namely proofs of disjunctive knowledge and the Fiat-Shamir transforms to construct our UDVS schemes. As mentioned before, our construction has the distinctive property that the verifier's key pair family can be chosen *independently* from the choice of the signer's signature. Our construction works for any combination of a signature in class \mathbb{C} and a verifier key pair family in \mathbb{K} .

To construct our UDVS schemes, we use Jakobsson et al's approach to providing verifier designation [JSI96]: “Instead of proving Θ , Alice proves the statement: *Either Θ is true, or I am Bob.*” In UDVS schemes, Alice wants to prove validity of her certificate to Bob. A natural construction of UDVS is a non-interactive proof of the following statement by Alice: “*Either my certificate is valid, or I am Bob.*” Such a signature can be constructed by applying the Fiat-Shamir transformed to a protocol for proof of knowledge of Alice's certificate *or* Bob's secret key. However, efficient protocols for honest-verifier zero-knowledge (HVZK) proof of knowledge of a signature on a message are only known for a small group of signature schemes. We show that for signatures in \mathbb{C} , a signature holder can first reveal $\tilde{\sigma}_{\text{aux}}$ and then prove knowledge of $\tilde{\sigma}_{\text{pre}}$ in order to prove knowledge of a signature. We prove that, although such a proof is *not* an HVZK proof of knowledge of a signature (since part of the signature is revealed), the above approach can be applied to it to construct secure UDVS schemes.

Let $SS = (\text{SKeyGen}, \text{Sign}, \text{Verify})$ be a standard signature scheme in class \mathbb{C} and KF be a verifier-chosen key family in \mathbb{K} . Denoting the signer- and verifier-related variables respectively by s and v indexes, the construction can be shown as follows:

- CPGen gets as input 1^k , where k is the security parameter, returns $cp = 1^k$ as the common parameter. The signer and the verifiers choose their own signature scheme and key pair families, respectively.
- To designate, the signature-holder first converts the signature and then constructs a signature of disjunctive knowledge of $\tilde{\sigma}_{\text{pre}}$ or the verifier's secret key. The DV-signature is a pair consisting of $\tilde{\sigma}_{\text{aux}}$ and this signature of knowledge.
- To verify the DV-signature, one verifies the validity of the signature of knowledge δ according to the message, the public keys of the signer and the verifier, and the value $\tilde{\sigma}_{\text{aux}}$ provided.

The scheme is transcribed in Figure 4.11.

Note that the designated verification algorithm in our construction is *public*, since pk_v is sufficient to

$\text{GUDVS. (SKeyGen, Sign, PVer)} \triangleq \text{SS. (SKeyGen, Sign, Verify)}$ $\text{VKeyGen} \triangleq \text{KeyGen}$
<hr/> <p>Algorithm GUDVS.Desig (pk_s, pk_v, m, σ)</p> $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) \leftarrow \text{Convert}(pk_s, m, \sigma)$ $\delta \leftarrow \text{SoK} \{(\tilde{\sigma}_{\text{pre}}, sk_v) : \text{Valid}(pk_s, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})) \vee \text{Pair}(pk_v, sk_v)\}$ $\hat{\sigma} \leftarrow (\tilde{\sigma}_{\text{aux}}, \delta)$ <p>Return $\hat{\sigma}$</p> <hr/>
<p>Algorithm GUDVS.DVer ($pk_s, pk_v, m, \hat{\sigma}$)</p> $d \leftarrow \text{VoK} \{(\tilde{\sigma}_{\text{pre}}, sk_v) : \text{Valid}(pk_s, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})) \vee \text{Pair}(pk_v, sk_v)\}(\hat{\sigma})$ <p>Return d</p>

Figure 4.11: Our generic construction of universal designated-verifier signatures

run the GUDVS.DVer algorithm. However, some authors have proposed a notion of *privacy of signer identity* in UDVS schemes that requires an only-verifier-executable designated verification [LV04a]. Our construction does not provide such property. In the same work, the authors show that if the designated signature is *encrypted* by the designator under an IND-CCA encryption and then sent to the verifier, then it will be verifiable only by the verifier and the scheme will preserve privacy of signer identity.

DV-Unforgeability. We use the Forking Lemma to prove DV-Unforgeability of our generic UDVS construction. We use the results and formulations of the General Forking Lemma (see Section 4.2.4) in our proof. Intuitively, our *SoK*-type constructions guarantees the ability to extract a signature or the verifier's secret key from a DV-forgery through forking. The extracted signature or secret key is later used to solve the underlying problem of the signature scheme or that of the verifier key pair, respectively. Thus, given a successful DV-forgery, we are able to solve at least one of the above underlying problems and we have the following theorem.

Theorem 4.4.4 *Let SS be a standard signature in \mathbb{C} and P_{SS} be its underlying problem. Also, let KF be a key family in \mathbb{K} and P_{KF} be its underlying problem. The construction GUDVS based on the combination of the signature SS and the verifier key family KF is DV-unforgeable if P_{SS} and P_{KF} are both hard.*

Proof. Let GUDVS be a UDVS scheme constructed via our constructions from a signature scheme SS in \mathbb{C} and a verifier key pair family KF in \mathbb{K} . Let also the underlying hard problem of the signature scheme be P_{SS} and the underlying hard problem of the verifier key pair family be P_{KF} . Given a DV-forgery A and two instances of the problems P_{SS} and P_{KF} , we show that at least one of the problem instances can be solved.

We show how to construct, given a DV-forgery A , two *solver* algorithms Slv_{KF} and Slv_{SS} for solving P_{KF} and P_{SS} instances, respectively. We also show that at least one of these two strategies succeeds in solving its given instance of the problem, if the DV-forgery manages to forge successfully. Given a successful DV-forgery A , Slv_{KF} succeeds only if the forgery produced by the A is of a certain family which is defined by an event. We also show that Slv_{SS} succeeds if the DV-forgery A is successful and another event occurs. Furthermore, we show that the events above cover the universe. It follows that, with the above two solvers, given a DV-forgery for GUDVS scheme, at least one of them solves the associated problem.

We construct our solvers in a modular way. We introduce four algorithms Sim_{KF} , Sim_{SS} , Cal_{KF} , and Cal_{SS} and use these four algorithms along with the adversary A and the Bellare-Neven forger algorithm (see Section 4.2.4) as modules of constructing the two solvers. We construct the solver Slv_{KF} as follows:

- the *simulator* algorithm Sim_{KF} runs A as a subroutine, simulating the attack environment (inputs and answers to queries) for A , and obtain a DV-forgery from A ,
- the *forker* algorithm Frk_{KF} runs Sim_{KF} as a subroutine, forking inputs to it, and obtain two different designated signatures from it, and
- the *solution calculator* algorithm Cal_{KF} runs Frk_{KF} as a subroutine and uses the two designated signatures output by it to solve the given instance of the problems P_{KF} .

The solver Slv_{SS} is also constructed in a similar way, using algorithms Sim_{SS} , Frk_{SS} , and Cal_{SS} . The algorithms Sim_{SS} and Cal_{SS} , in turn, run the Sim and Cal algorithms of the signature scheme SS , respectively. These algorithms are defined in Sections 4.3.1 and 4.3.2. The forker algorithms are based on the constructions introduced in the General Forking Lemma as discussed in Section 4.2.4. We describe each module in the following and discuss how they work and lead to the proof. First we describe Sim_{KF} , Frk_{KF} , and Cal_{KF} algorithms, which are used to construct the algorithm for solving a given P_{KF} problem instance. After a discussion on the success probability of our solver, we proceed to introduce our second set of algorithms Sim_{SS} , Frk_{SS} , and Cal_{SS} , which are used to construct the algorithm for solving a given P_{SS} problem instance. We denote the random oracles used in the signature scheme by \mathcal{H}_{SS} and the one used in the Fiat-Shamir transform to build a signature of knowledge by \mathcal{H}_{FS} .

Let us first set some notations. One can see easily from Figures 4.2 and 4.3 and our construction that our designated signatures be in the form $\hat{\sigma} = (\tilde{\sigma}_{\text{pre}}, \delta)$, where

$$\delta = (Cmt, Rsp) = ((Cmt_s, Cmt_v), (Chl_s, Chl_v, Rsp_s, Rsp_v)) \quad .$$

Furthermore, we have

$$Chl_s + Chl_v = Chl = H_{FS}(Pub \parallel Cmt) = H_{FS}((Pub_s, Pub_v) \parallel (Cmt_s, Cmt_v)) \quad .$$

Also note that, following our proof of knowledge notation convention, the notation

$$\delta \leftarrow \text{SoK} \{(\tilde{\sigma}_{\text{pre}} \vee sk_v) : \text{Valid}(pk_s, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})), \text{Pair}(pk_v, sk_v)\}$$

implies that

$$Sec_s = \tilde{\sigma}_{\text{pre}}, \quad Sec_v = sk_v, \quad Pub_s = (pk_s, m, \tilde{\sigma}_{\text{aux}}), \quad \text{and} \quad Pub_v = pk_v \quad .$$

We use the above notation throughout the proof.

Algorithm Sim_{KF} gets a P_{KF} instance Ins and a q -tuple (h_1, \dots, h_q) as input. It first runs the key generation algorithm of the corresponding signature to obtain a key pair (sk_s, pk_s) . Then Sim_{KF} runs A with inputs pk_s and $pk_v = Ins$. Note that, as we mentioned before, one can see the problem instance Ins as a public key pk_v , for which we are trying to find the solution Sol , i.e. corresponding secret key sk_v . During its run, A asks \mathcal{H}_{SS} , \mathcal{H}_{FS} , and Sign oracle queries. Sim_{KF} simulates the answers as follows:

- answers \mathcal{H}_{SS} queries randomly and records the answers.
- answers \mathcal{H}_{FS} queries by taking elements of the q -tuple (h_1, \dots, h_q) consecutively, i.e. answers the first query with h_1 , the second with h_2 and so on.
- answers Sign queries by running the Sign algorithm of the signature scheme. Note that the signing key sk_s is known to Sim_{KF} , so there is no need to simulate the signatures.

At last, A outputs a DV-forgery $(m, \hat{\sigma})$, where $\hat{\sigma} = (\tilde{\sigma}_{aux}, \delta)$. Sim_{KF} checks whether or not the adversary has been successful in forging, i.e. checks whether or not the message is new and the DV-forgery is valid by running the DVer algorithm. If $(m, \hat{\sigma})$ passes both tests, denoting $\delta = ((Cmt_s, Cmt_v), (Chl_s, Chl_v, Rsp_s, Rsp_v))$, Sim_{KF} looks up the *index* J s.t. $h_J = Chl_s + Chl_v$ and outputs $(J, (m, \hat{\sigma}))$. In the case that the adversary has not been successful or no matching index J is found, Sim_{KF} outputs $(0, \varepsilon)$.

Algorithm Frk_{KF} takes as input a P_{KF} instance Ins . It is defined as the Bellare-Neven forker algorithm in Section 4.2.4, with input Ins and access to algorithm Sim_{KF} , i.e. using the Bellare-Neven notation

$$\text{Frk}_{KF} \triangleq F_{\text{Sim}_{KF}}(Ins) \quad .$$

Frk_{KF} outputs either $(1, (m, \hat{\sigma}), (m', \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$, depending on whether the forking has been successful or not. Note that a successful forking implies same J th \mathcal{H}_{FS} oracle query and different corresponding answers. Since queries are of the form $(Pub_s, Pub_v) \parallel (Cmt_s, Cmt_v)$, where $Pub_s = (pk_s, m, \tilde{\sigma}_{aux})$, the message m is also part of the J th query and thus is the same for the two runs of the forked algorithm Sim_{KF} , hence $m = m'$. Therefore, from now on, we use m instead of m' .

Algorithm Cal_{KF} takes as input a P_{KF} instance Ins . It first runs Frk_{KF} on the same input Ins and obtains either $(1, (m, \hat{\sigma}), (m, \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$. Receiving the former means that forking by Frk_{KF} has been successful, i.e. $J = J'$ and $h_J \neq h'_J$ according to the General Forking Lemma. Note that h_J and h'_J are the two responses to the J th \mathcal{H}_{FS} oracle queries in the two runs of the forked algorithm Sim_{KF} . Thus $h_J = Chl = Chl_s + Chl_v$ and $h'_J = Chl' = Chl'_s + Chl'_v$. Hence we have the following event:

$$\mathbf{E} \triangleq [\quad J = J' \quad \wedge \quad Chl_s + Chl_v \neq Chl'_s + Chl'_v \quad] \quad . \quad (4.1)$$

Now, if $Chl_v \neq Chl'_v$, then Cal_{KF} simply runs the extraction algorithm for the protocol for proof of knowledge of the verifier's secret key and get a sk_v s.t. $\text{Pair}(pk_v, sk_v)$. Cal_{KF} outputs $Sol = sk_v$ as the solution to the P_{KF} problem instance Ins . If $Chl_v = Chl'_v$, Cal_{KF} declares failure and halts.

A graphical depiction of how modules are wired to interact in our solver is shown in Figure 4.12. Note that, again, random oracle queries are not shown in the figure. Let us denote by Slv_{KF} our solver, i.e.

the combination of all our modules: Cal_{KF} , Frk_{KF} , and the two instances of Sim_{KF} wired together as in Figure 4.12.

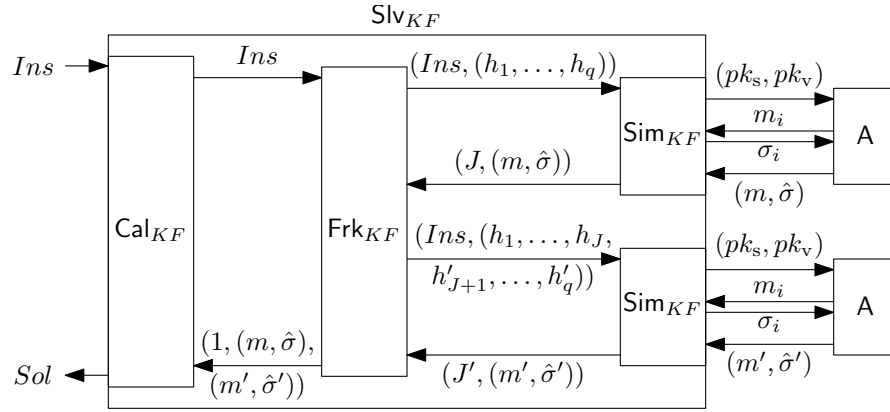


Figure 4.12: Mechanism of our DV-EUF-CMA-security proof for our GUDVS construction

Let us calculate the probability that our solver is successful in solving the P_{KF} problem instance Ins . We define the success probabilities for Sim_{KF} and Frk_{KF} similar to acc and frk , respectively, in the General Forking Lemma (see Section 4.2.4), i.e.

- $\text{Adv}_{\text{Sim}_{KF}(\mathbf{A})}(k)$ is defined as the probability that Sim_{KF} 's first output is not zero, given \mathbf{A} , a random problem instance of size k , and random choices of h_1, \dots, h_q , and
- $\text{Adv}_{\text{Frk}_{KF}(\text{Sim}_{KF})}(k)$ is defined as the probability that Frk_{KF} 's first output is one, given Sim_{KF} and a random problem instance of size k .

Now we observe that Sim_{KF} succeeds if \mathbf{A} succeeds in forging and the forgery uses a queried hash. We know that the probability that \mathbf{A} succeeds without using a queried hash is at most one over the size of the challenge space. Thus we have

$$\text{Adv}_{\text{Sim}_{KF}(\mathbf{A})}(k) \geq \text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1}{|\text{ChSp}|}.$$

On the other hand, we have Bellare and Neven's General Forking Lemma which gives us a lower bound for the success probability of the forker, i.e. Frk_{KF} , based on the success probability of the simulator, i.e. Sim_{KF} . Thus we have

$$\text{Adv}_{\text{Frk}_{KF}(\text{Sim}_{KF})}(k) \geq \text{Adv}_{\text{Sim}_{KF}(\mathbf{A})}(k) \cdot \left(\frac{\text{Adv}_{\text{Sim}_{KF}(\mathbf{A})}(k)}{q} - \frac{1}{|\text{ChSp}|} \right),$$

where q is the maximum number of \mathcal{H}_{FS} queries \mathbf{A} makes. We also see that Cal_{KF} is successful if Frk_{KF} succeeds and $\text{Chl}_v \neq \text{Chl}'_v$. So we get the following:

$$\text{Adv}_{\text{Cal}_{KF}(\text{Frk}_{KF})}^{P_{KF}}(k) = \Pr[\text{Chl}_v \neq \text{Chl}'_v | \text{Frk}_{KF} \text{ succeeds}] \cdot \text{Adv}_{\text{Frk}_{KF}(\text{Sim}_{KF})}(k).$$

Combining the three equations above, and applying the fact that Frk_{KF} succeeds *iff* \mathbf{E} happens, we can compute the overall probability of success of our solver in solving P_{KF} as follows:

$$\begin{aligned} \text{Adv}_{\text{Slv}_{KF}(\mathbf{A})}^{P_{KF}}(k) &\geq \frac{1}{q} \cdot \Pr[\text{Chl}_v \neq \text{Chl}'_v | \mathbf{E}] \cdot \\ &\quad \cdot \left(\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1}{|\text{ChSp}|} \right) \cdot \left(\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1+q}{|\text{ChSp}|} \right). \end{aligned}$$

Assuming that the size of the challenge space is super-logarithmic in the security parameter and the number of queries the adversary asks is polynomially-bounded in the security parameter, we can neglect the two fractions with $|ChSp|$ as denominator and simplify the above equation as follows:

$$\text{Adv}_{\text{Slv}_{KF}(\mathbf{A})}^{\text{P}_{KF}}(k) \geq \frac{1}{q} \cdot \Pr[Chl_v \neq Chl'_v | \mathbf{E}] \cdot \left[\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) \right]^2. \quad (4.2)$$

Now we describe the three algorithms Sim_{SS} , Frk_{SS} , and Cal_{SS} for solving an instance Ins of the underlying problem of the signature scheme P_{SS} . These modules are again wired together as shown in Figure 4.12, changing all the indexes from KF to SS .

Algorithm Sim_{SS} gets an instance Ins of the problem P_{SS} and a q -tuple (h_1, \dots, h_q) as input. It first runs the corresponding verifier key generation algorithm KeyGen to obtain a key pair (sk_v, pk_v) . Then Sim_{SS} runs the simulator algorithm Sim of the signature scheme SS on input Ins to get a public key pk_s for the signature scheme. It then runs \mathbf{A} on input (pk_s, pk_v) . \mathbf{A} asks \mathcal{H}_{SS} , \mathcal{H}_{FS} , and Sign oracle queries. Sim_{SS} responds as follows:

- forwards all \mathcal{H}_{SS} and Sign oracle queries to the signature simulator Sim and relays the answers given by Sim back to \mathbf{A} .
- answers \mathcal{H}_{FS} queries with the q -tuple it is provided with, i.e. answers the first query with h_1 , the second with h_2 and so on.

If Sim succeeds in simulating the \mathcal{H}_{SS} and Sign oracle queries, at last \mathbf{A} outputs a DV-forgery $(m, \hat{\sigma})$. Sim_{SS} checks whether or not the adversary has been successful in forging, i.e. checks whether or not the message is new and the DV-forgery is valid by running the DVer algorithm. If $(m, \hat{\sigma})$ passes both tests, Sim_{SS} looks up the *index* J s.t. $h_J = Chl_s + Chl_v$ and outputs $(J, (m, \hat{\sigma}))$. In the case that either Sim fails, the adversary fails in forging a valid forgery, or no matching index J is found, Sim_{SS} outputs $(0, \varepsilon)$.

Algorithm Frk_{SS} takes as input an instance Ins of the problem P_{SS} . It is defined as the Bellare-Neven forker algorithm, with input Ins and access to algorithm Sim_{SS} , i.e.

$$\text{Frk}_{\text{SS}} \triangleq \text{F}_{\text{Sim}_{\text{SS}}}(Ins) \quad .$$

Frk_{SS} outputs either $(1, (m, \hat{\sigma}), (m', \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$. Note that, with a similar reasoning as before, $m = m'$.

Algorithm Cal_{SS} takes as input an instance Ins of the problem P_{SS} . It runs Frk_{SS} on the same input and obtains either $(1, (m, \hat{\sigma}), (m', \hat{\sigma}'))$ or $(0, \varepsilon, \varepsilon)$. Again, receiving the former means that forking by Frk_{SS} has been successful, i.e. $J = J'$ and $h_J \neq h_{J'}$. Hence we have the same event \mathbf{E} as defined in Equation 4.1. Now, if $Chl_s \neq Chl'_s$, then Cal_{SS} simply runs the extraction algorithm for the protocol for proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ and get a $\tilde{\sigma}_{\text{pre}}$ s.t. $\text{Valid}(pk_s, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})_v)$. Then it runs the corresponding Retrieve algorithm on input $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$ and gets a valid σ . Now, Cal_{SS} feeds (m, σ) to the solution calculator algorithm Cal of the signature scheme SS and gets the solution Sol for the problem instance Ins of the problem P_{SS} if Cal is successful. If either $Chl_s = Chl'_s$ or Cal fails, Cal_{SS} declares failure and halts.

Let us calculate the probability that our solver is successful in solving the P_{SS} problem instance Ins . We can define the success probability for Sim_{SS} and Frk_{SS} similar to that of Sim_{KF} and Frk_{KF} . Notice

that Sim_{SS} succeeds if Sim succeeds in simulating, \mathbf{A} succeeds in forging, and the forgery uses a queried hash. Thus, with similar reasonings as before, the success probability of Sim_{SS} can be finally written as

$$\text{Adv}_{\text{Sim}_{\text{SS}}(\mathbf{A})}(k) \geq \text{Adv}_{\text{Sim}(\mathbf{A})}(k) \cdot \left(\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1}{|\text{ChSp}|} \right).$$

Furthermore, for the success probability of the forker algorithm Frk_{SS} , a similar equation to the one we had in the first part of the proof holds, i.e.

$$\text{Adv}_{\text{Frk}_{\text{SS}}(\text{Sim}_{\text{SS}})}(k) \geq \text{Adv}_{\text{Sim}_{\text{SS}}(\mathbf{A})}(k) \cdot \left(\frac{\text{Adv}_{\text{Sim}_{\text{SS}}(\mathbf{A})}(k)}{q} - \frac{1}{|\text{ChSp}|} \right),$$

where q is the maximum number of \mathcal{H}_{FS} queries made by \mathbf{A} . We also see that Cal_{SS} is successful if Frk_{SS} succeeds in forking, $\text{Chl}_v \neq \text{Chl}'_v$, and Cal succeeds in solving the problem instance Ins . Now let us define the following event:

$$\mathbf{F} \triangleq [\text{Frk}_{\text{SS}} \text{ succeeds} \quad \wedge \quad \text{Chl}_s \neq \text{Chl}'_s] .$$

In case of event \mathbf{F} , a valid signature σ on the message m can be computed. The probability of obtaining such a signature can be written as

$$\Pr[\mathbf{F}] = \Pr[\text{Chl}_s \neq \text{Chl}'_s | \text{Frk}_{\text{SS}} \text{ succeeds}] \cdot \text{Adv}_{\text{Frk}_{\text{SS}}(\text{Sim}_{\text{SS}})}(k) .$$

Now, using the notation defined in Section 4.3.1, for non-FL-based signatures, we have the following:

$$\text{Adv}_{\text{Cal}_{\text{SS}}(\text{Frk}_{\text{SS}})}^{\text{P}_{\text{SS}}}(k) \geq \text{Adv}_{\text{Cal}(\text{Sim})}(k) \cdot \Pr[\mathbf{F}] .$$

Combining the above equations and with a similar reasoning that lead us to Equation 4.2 plus the fact that \mathbf{E} happens *iff* Frk_{SS} succeeds, we get the following for overall probability of success of our solver in solving P_{SS} for non-FL-based signatures:

$$\begin{aligned} \text{Adv}_{\text{Siv}_{\text{SS}}}^{\text{P}_{\text{SS}}}(k) &\geq \frac{1}{q} \cdot \text{Adv}_{\text{Cal}(\text{Sim})} \cdot \Pr[\text{Chl}_s \neq \text{Chl}'_s | \mathbf{E}] \cdot [\text{Adv}_{\text{Sim}(\mathbf{A})}(k)]^2 \cdot \\ &\quad \cdot \left(\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1}{|\text{ChSp}|} \right) \cdot \left(\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) - \frac{1+q}{|\text{ChSp}| \cdot \text{Adv}_{\text{Sim}(\mathbf{A})}(k)} \right), \end{aligned}$$

where we also have exploited the fact that $\text{Adv}_{\text{Sim}(\mathbf{A})}(k) \leq 1$ to change the last numerator from $\text{Adv}_{\text{Sim}(\mathbf{A})}(k) + q$ to $1 + q$. Similarly, assuming that the size of the challenge space is super-logarithmic, the number of queries the adversary asks is polynomially-bounded, and $\text{Adv}_{\text{Sim}(\mathbf{A})}(k)$ is noticeable in the security parameter, we can simplify the above equation as follows:

$$\text{Adv}_{\text{Siv}_{\text{SS}}}^{\text{P}_{\text{SS}}}(k) \geq \frac{1}{q} \cdot \text{Adv}_{\text{Cal}(\text{Sim})} \cdot \Pr[\text{Chl}_s \neq \text{Chl}'_s | \mathbf{E}] \cdot [\text{Adv}_{\text{Sim}(\mathbf{A})}(k)]^2 \cdot \left[\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) \right]^2 . \quad (4.3)$$

Combining Equations 4.2 and 4.3 and applying the fact that

$$\Pr[\text{Chl}_v \neq \text{Chl}'_v | \mathbf{E}] + \Pr[\text{Chl}_s \neq \text{Chl}'_s | \mathbf{E}] \geq 1 ,$$

we get the following result for non-FL-based schemes:

$$\text{Adv}_{\text{Sim}_{KF}(\mathbf{A})}^{\text{P}_{KF}}(k) + \frac{1}{\text{Adv}_{\text{Cal}(\text{Sim})} \cdot [\text{Adv}_{\text{Sim}(\mathbf{A})}(k)]^2} \cdot \text{Adv}_{\text{Siv}_{\text{SS}}(\mathbf{A})}^{\text{P}_{\text{SS}}}(k) \geq \frac{1}{q} \cdot \left[\text{Adv}_{\mathbf{A}(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k) \right]^2 .$$

Thus, as long as the adversary A has a good advantage of forging, we are able to solve at least one of the problem instances of P_{KF} or P_{SS} with a good probability. This completes the proof for non-FL-based signatures.

On the other hand, for the FL-based signatures, since another forking is performed to get two valid signatures, we get the following:

$$\text{Adv}_{\text{CalSS}(\text{Frk}_{SS})}^{\text{P}_{SS}}(k) \geq \text{Adv}_{\text{Cal}(\text{Sim})}(k) \cdot \Pr[\mathbf{F}] \cdot \left(\frac{\Pr[\mathbf{F}]}{q} - \frac{1}{|R_{H_{SS}}|} \right),$$

where $R_{H_{SS}}$ is the range of the hash function H_{SS} . Again with a similar reasoning, assuming that the size of the challenge space and the size of $R_{H_{SS}}$ are both super-logarithmic, the number of queries the adversary asks is polynomially-bounded, and $\text{Adv}_{\text{Sim}(A)}(k)$ is noticeable in the security parameter, we get the following final result for overall probability of success of our solver in solving P_{SS} for FL-based signatures:

$$\text{Adv}_{\text{Sim}_{SS}}^{\text{P}_{SS}}(k) \geq \frac{1}{q^3} \cdot \text{Adv}_{\text{Cal}(\text{Sim})} \cdot (\Pr[\text{Chl}_s \neq \text{Chl}'_s | \mathbf{E}])^2 \cdot [\text{Adv}_{\text{Sim}(A)}(k)]^4 \cdot [\text{Adv}_{A(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k)]^4. \quad (4.4)$$

Similarly, combining Equations 4.2 and 4.4, we get the following result for FL-based schemes:

$$\text{Adv}_{\text{Sim}_{KF}(A)}^{\text{P}_{KF}}(k) + \frac{\sqrt{q}}{\sqrt{\text{Adv}_{\text{Cal}(\text{Sim})} \cdot [\text{Adv}_{\text{Sim}(A)}(k)]^2}} \cdot \sqrt{\text{Adv}_{\text{Sim}_{SS}(A)}^{\text{P}_{SS}}(k)} \geq \frac{1}{q} \cdot [\text{Adv}_{A(\text{GUDVS})}^{\text{DV-EUF-CMA}}(k)]^2,$$

which again, guarantees a lower band for the probability that our solvers are able to solve at least one of the two instances of respectively the problems P_{KF} and P_{SS} . This completes the proof for FL-based signatures. \blacksquare

Non-Transferability Privacy. Non-transferability privacy for our generic UDVS schemes is due to the very concept behind our construction. Our designated signatures consist of publicly-simulatable values of $\tilde{\sigma}_{\text{aux}}$ and *witness indistinguishable* signatures of knowledge of a valid converted signature or the verifier's secret key, both forgeable by the designated verifier himself indistinguishably from the real designated signatures. In fact, to forge a designated signature, the verifier can simply execute the following algorithm:

Algorithm $\text{GUDVS.Forge}(pk_s, sk_v, pk_v, m)$
 $\tilde{\sigma}_{\text{aux}} \leftarrow \text{AuxSim}(pk_s, m)$
 $\delta \leftarrow \text{SoK} \{(\tilde{\sigma}_{\text{pre}}, sk_v) : \text{Valid}(pk_s, m, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})) \vee \text{Pair}(pk_v, sk_v)\}$
 Return $(\tilde{\sigma}_{\text{aux}}, \delta)$

AuxSim 's ability to simulate $\tilde{\sigma}_{\text{aux}}$ and witness indistinguishability of the signature of knowledge together imply that the output of the algorithm GUDVS.Forge is indistinguishable from real designated signatures. Combining this with Lemma 4.4.1, we have the following theorem.

Theorem 4.4.5 *The construction GUDVS achieves non-transferability privacy for any combination of a signature in \mathbb{C} and a verifier key family in \mathbb{K} .*

Non-Delegatability. The very design of our UDVS construction is naturally geared to provide non-delegatability through the use of signatures of knowledge. However, to meet the requirements of Lemma 4.4.3, we must first prove that a designated signature in our scheme is a signatures of knowledge of a *signature* or the secret key of the verifier with SpS property. All we know now is that a designated signature in our scheme consists of a $\tilde{\sigma}_{\text{aux}}$ and a signature of knowledge of $\tilde{\sigma}_{\text{pre}}$ or the secret keys of the verifier with both HVZK and SpS properties.

One can see that a designated signature $(\tilde{\sigma}_{\text{aux}}, \delta)$ as a signature of knowledge has the SpS property in the Random Oracle Model. The reason is that two designated signatures with the same first-move message (i.e. Random Oracle query, which includes $\tilde{\sigma}_{\text{aux}}$ along with the commitment) and different challenges (i.e. Random Oracle responses) provides two δ s with the same commitment and different challenges, which in turn, gives us the secret, i.e. $\tilde{\sigma}_{\text{pre}}$ or sk_v . If the former is given, then one can retrieve a valid signature by running the Retrieve algorithm on input $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$. Thus, two designated signatures with the same Random Oracle query and different Random Oracle responses give us a signature or the verifier's secret key. Hence, the designated signature has the SpS property as well and by Lemma 4.4.3 we have the following theorem:

Theorem 4.4.6 *The construction GUDVS is κ -non-delegatable for any combination of a signature in \mathbb{C} and a verifier key family in \mathbb{K} for which $|ChSp| \geq \frac{1}{\kappa}$.*

Optimizing Designated Signature Length. Our construction leads to designated signatures of the form

$$\hat{\sigma} = \left(\tilde{\sigma}_{\text{aux}}, ((Cmt_s, Cmt_v), (Chl_s, Chl_v, Rsp_s, Rsp_v)) \right) .$$

We observe that in many cases the component (Cmt_s, Cmt_v) of the designated signature can be omitted. This reduces the size of the designated signature. More precisely, if both Dcd_s and Dcd_v provide algorithms to compute Cmt_s given pk_s, Chl_s, Rsp_s and to compute Cmt_v given pk_v, Chl_v, Rsp_v , respectively, then the signature of knowledge δ does not need to contain the component (Cmt_s, Cmt_v) . Technically, assume that Dcd_s and Dcd_v can be written as follows:

Algorithm $Dcd_s(pk_s, Cmt_s, Chl_s, Rsp_s)$ $Cmt'_s \leftarrow \text{CptCmt}_s(pk_s, Chl_s, Rsp_s)$ $d \leftarrow (Cmt_s \equiv Cmt'_s)$ Return d	Algorithm $Dcd_v(pk_v, Cmt_v, Chl_v, Rsp_v)$ $Cmt'_v \leftarrow \text{CptCmt}_v(pk_v, Chl_v, Rsp_v)$ $d \leftarrow (Cmt_v \equiv Cmt'_v)$ Return d
---	---

Then, it is sufficient to construct the signature of knowledge by only including Rsp . That is, the designated signature can be constructed as follows:

$$\hat{\sigma} = \left(\tilde{\sigma}_{\text{aux}}, (Chl_s, Chl_v, Rsp_s, Rsp_v) \right) .$$

The designated signature can be then verified as follows. One first computes Cmt'_s and Cmt'_v through CptCmt_s and CptCmt_v . Then, one checks if these values hash to $Chl_s + Chl_v$.

Note that, both GQ and Schnorr protocols have decision algorithms that can be written as above and hence, this method can be applied to almost all combinations of signatures and verifier key pairs in use today.

Multiple Designated Verifiers: Our constructions of UDVS schemes can be extended to *universal multi-designated-verifier signatures*, where a signature is designated to more than one verifier. This can be done by setting the designated signature to be a signature of knowledge of the (converted) signature or all the secret keys of the n verifiers. That is, the designated signature contains the auxiliary information $\tilde{\sigma}_{\text{aux}}$ and

$$\text{SoK} \left\{ \tilde{\sigma}_{\text{pre}} \vee \left(\bigwedge_{i=1}^n sk_{v_i} \right) \right\}.$$

Again, these schemes allow the signer and the verifiers to choose their settings independently, thus the verifiers might have different families of keys. The signature of knowledge based construction guarantees that the non-transferability privacy and non-delegatability requirements are met for these schemes. We expect that a similar proof of DV-unforgeability can be also written for such schemes but we do not elaborate on it since we consider it out of the scope of this thesis.

4.4.3 Comparison

We compare our constructions to the previous constructions in the literature. For the comparison to make sense, we choose instances of our constructions that match the signature scheme and verifier key family of each benchmark scheme. To simplify our comparisons and as evidenced in [MvOV97], we assume the cost of computing a product $a^x \cdot b^y \cdot c^z$ is equivalent to a single exponentiation. We use the following typical parameters for lengths of elements of different sets in use: for DL-based schemes, 1024 bits for group elements and 160 bits for exponents, for RSA-based schemes, 1024 bits for elements, for pairing-based schemes, 171 bits for elements of paired groups and 1024 bits for elements of the target group, and 160 bits for random salts and hash function outputs. Using the above values and the low RSA public exponent $e = 2^{16} + 1$, each low exponent exponentiation costs a tenth of a single exponentiation (see e.g. [SWP04]). To further simplify the comparison, we only consider the dominant term for the costs of computation, assuming that a pairing computation costs (much) more than an exponentiation, which itself costs (much) more than a multiplication, which itself costs (much) more than an addition.

In Table 4.4 the first column lists the signature scheme and designated verifier key family of the schemes. For instance, BLS+DL indicates the case where the signer and the verifier are respectively using the BLS signature and a DL pair family of keys. SchS, RSA, and Wat indicate Schnorr, RSA-FDH, and Waters signatures, respectively. DVSBM is from [SBWP03]. DVSBMH is from [LV04a]. UDVS-BLS and UDVS-BB are both from [Ver06]. SchUDVS₁, SchUDVS₂, and RSAUDVS are all from [SWP04]. ZFI is from [ZFI05]. LLQ₁ and LLQ₂ are both from [LLQ06]. HSMW is from [HSMW08]. TGOBO₁ and TGOBO₂ are both from [TGO⁺07]. Note that [HSMW06] is the same as our GUDVS_{SchP} scheme based on BLS+DL. Each section of the table lists schemes based on the same signature and designated verifier keys. The last schemes on each section are the GUDVS counterparts that are comparable to the listed schemes, that is they are based on the same signature and designated verifier keys. The GUDVS scheme is constructed using protocols for proof of knowledge of converted signatures as listed in Tables 4.2 and 4.3. For pairing-based signatures, two different methods of conversion and proof of knowledge, hence two resulting schemes are possible. Both these schemes are shown in the table. ‘ROM’ and ‘StM’ stand for random oracle model and standard model, respectively. ‘e’ and ‘p’

stand for exponentiation and pairing computations, respectively. SI and ND stand for signer—verifier setting independence and non-delegatability, respectively. A ✓, ✗, or ? shows that either the scheme has the property, it does not have the property, or it is not known whether or not it has the property, respectively. \approx refers to the verifier-key flexibility property of [TGO⁺07] that enables the verifier to have either an RSA-based or a DL-based key pair. Comparatively more desirable quantities are highlighted in boldface. Note that the values we obtain for the sizes of our designated signatures are the optimized values through methods discussed in page 80.

Table 4.4: Comparison of previous UDVS schemes in the literature with our corresponding GUDVS counterparts based on the original signature and the designated-verifier key family

SS + KF	Scheme	Assumption	Model	Cost				$ \hat{\sigma} $ (bits)	SI	ND
				Desig			DVer			
				off-line	on-line	total				
BLS + DL	DVSBM	BDH	ROM	nil	1 p	1 p	1 p	1024	X	X
	DVSBMH	GBDH	ROM	nil	1 p	1 p	1 p	160	X	X
	UDVS-BLS	SCBDH	ROM	1 e	1 e	2 e	2 p	342	X	X
	GUDVS _{SchP}	CDH	ROM	1 p	1 e	1 p	2 p	811	✓	✓
	GUDVS _{CCP}	CDH	ROM	1 e	2 e	3 e	2 p	662	✓	✓
SchS + DL	SchUDVS ₁	SDH	ROM	1 e	1 e	2 e	1 e	2048	X	X
	SchUDVS ₂	DL	ROM	2 e	1 e	3 e	2 e	1504	X	?
	GUDVS	DL	ROM	2 e	1 e	3 e	2 e	1664	✓	✓
RSA + DL	RSAUDVS	RSA+DL	ROM	1 e	2 e	3 e	2 e	11584	X	?
	GUDVS	RSA+DL	ROM	0.1 e	1 e	1 e	2 e	1504	✓	✓
BB + DL	ZFI	KEA+SDH	StM	1 e	1 p	1 p	2 p	1366	X	?
	UDVS-BB	KEA+SDH	StM	2 e	1 e	3 e	2 p	673	X	X
	GUDVS _{SchP}	SDH	ROM	2 e	1 e	3 e	1 p	971	✓	✓
	GUDVS _{CCP}	SDH	ROM	1 e	2 e	3 e	2 p	822	✓	✓
Wat + DL	LLQ ₁	GBDH	StM	nil	1 p	1 p	2 p	1195	X	X
	LLQ ₂	BDH	StM	nil	1 p	1 p	2 p	1355	X	X
	HSMW	GBDH	StM	nil	1 p	1 p	2 p	1024	X	X
	GUDVS _{SchP}	CDH	ROM	1 p	1 e	1 p	2 p	982	✓	✓
	GUDVS _{CCP}	CDH	ROM	2 e	2 e	4 e	3 p	833	✓	✓
EIG + DL	TGOBO ₁	DSA+DL	ROM	3 e	1 e	4 e	3 e	1664	≈	?
	GUDVS	DL	ROM	1 e	1 e	2 e	2 e	1664	✓	✓
EIG + RSA	TGOBO ₂	DSA+RSA	ROM	3 e	0.1 e	3 e	2 e	2528	≈	?
	GUDVS	DL+RSA	ROM	1 e	1 e	2 e	2 e	2528	✓	✓

We note that designation of a certain certificate can be performed in two phases: before choosing the designated verifier and after that. Thus, computations can be carried out in two phases accordingly. We use the terms *off-line* and *on-line* to denote the two phases, respectively. An interesting property of our constructions is that cost of the on-line phase of designation is usually low (typically 1 exponentiation). This makes our constructions desirable for the systems in which certificates are often needed to be verified by (and hence designated to) multiple different verifiers. Table 4.4 summarizes our

comparisons. As the table shows, our schemes generally have lower costs of on-line designation and yet are comparable in costs of designated verification and size of designated signatures. Our schemes provide provable non-delegatability and setting independence for the signer and verifier. Furthermore, our proof technique allows for security of our schemes to be typically based on weaker assumptions. Note that, as a side effect of using the Forking Lemma for proof of security, our security reductions are not *tight*.

The table shows that security proofs for our schemes are based on either weaker or similar assumptions than those of their counterparts. Also in all but the EIG+RSA based schemes, our constructions provide either faster or equal on-line designation times than those of their counterparts. Furthermore, in all but the BLS+DL and SchS+DL based schemes, our schemes even have either faster or equal designated signature verification times than those of their counterparts. Our RSA+DL and Wat+DL based schemes even have shorter designated signature lengths than those of their counterparts.

In Table 4.5 we compare each existing UMDVS scheme in the literature with a scheme resulting from applying our construction to the same underlying signature and designated verifier key family. We assume the same parameters as our previous comparisons. NSM is from [NSM05]. UDVS-BLS(n) and UDVS-BB(n) are from [Ver08]. SKS is from [SKS06]. n is the number of designated verifiers. *Public verifiability* is a desirable property in UMDVS schemes and hence included in the table as PV. If designated signatures are publicly verifiable, then each designated verifier can perform the designated signature verification by himself. Otherwise, all designated verifiers must collaborate to be able to verify a designated signature.

As the table shows, our schemes provide public verifiability, setting independence, and non-delegatability. Furthermore, security proofs for our schemes are based on weaker assumptions than those of their counterparts. In addition, our BLS+DL based schemes have much shorter designated signature lengths than those of their counterparts.

4.5 Identity-based Signatures

Identity-based cryptosystems were proposed to overcome the problem of lack of public-key infrastructure which the public-key cryptosystems face. In such systems, public-key certificates are no longer needed, and the identities of the users are used as their public keys. However, users lose their ability to construct their own secret keys by themselves and must depend on a *key-generation center* (KGC) to provide them with their respective private keys. Syntactical and security definitions for identity-based signatures are discussed in Section 2.5. In this section, we propose our generic construction of identity-based signature schemes from any signature scheme in \mathbb{C} and prove it secure.

4.5.1 Generic Construction of IBS and Its Security

In this section we show how to extend any signature in \mathbb{C} to an IBS scheme. The idea is to use the key pair generated for the signature scheme as the master key pair and use the signing algorithm as the user key generation in the following way: to generate a user secret key for an identity, the identity is signed and the signature on the identity is given to the user as the user secret key. Now, the user is

Table 4.5: Comparison of UMDVS schemes in the literature with their corresponding GUMDVS counterparts based on the original signature and the designated-verifier key family

Type	Scheme	Assumption	Model	Cost	$ \hat{\sigma} $ (bits)	PV	SI	ND
BLS+DL	NSM	BDH	ROM	...	1024	✗	✗	✗
	UDVS-BLS(n)	SCBDH	ROM	...	$160 + 171(n + 1)$	✓	✗	✗
	GUMDVS _{SchP}	CDH	ROM	...	$651 + 160n$	✓	✓	✓
	GUMDVS _{CCP}	CDH	ROM	...	$502 + 160n$	✓	✓	✓
BB+DL	UDVS-BB(n)	KEA+SDH	StM	...	673	✓	✗	✗
	SKS	?	StM	...	$171(n + 1)$	✗	✗	✗
	GUMDVS _{SchP}	SDH	ROM	...	$811 + 160n$	✓	✓	✓
	GUMDVS _{CCP}	SDH	ROM	...	$662 + 160n$	✓	✓	✓

↓

Type	Scheme	Cost			
		Desig			DVer
		off-line	on-line	total	
BLS+DL	NSM	nil	$1 \text{ p} + (n - 1) \text{ m}$	$1 \text{ p} + (n - 1) \text{ m}$	$(2n + 1) \text{ p}$
	UDVS-BLS(n)	1 e	$n \text{ e}$	$(n + 1) \text{ e}$	$2n \text{ p}$
	GUMDVS _{SchP}	1 p	$n \text{ e}$	$1 \text{ p} + (n + 1) \text{ e}$	$2 \text{ p} + n \text{ e}$
	GUMDVS _{CCP}	1 e	$(n + 1) \text{ e}$	$(n + 2) \text{ e}$	$2 \text{ p} + n \text{ e}$
BB+DL	UDVS-BB(n)	2 e	$1 \text{ e} + (n - 1) \text{ m}$	$3 \text{ e} + (n - 1) \text{ m}$	$4 \text{ p} + (n - 1) \text{ m}$
	SKS	nil	$n \text{ e}$	$n \text{ e}$	$1 \text{ p} + (n + 1) \text{ e}$
	GUMDVS _{SchP}	2 e	$n \text{ e}$	$(n + 2) \text{ e}$	$1 \text{ p} + (n + 1) \text{ e}$
	GUMDVS _{CCP}	1 e	$(n + 1) \text{ e}$	$(n + 2) \text{ e}$	$2 \text{ p} + (n + 1) \text{ e}$

able to prove her identity, since she can prove knowledge of a converted signature on her identity. The Fiat-Shamir transform can be used to transform this proof into a signature scheme. The resulting signature would be an identity-based signature.

The concrete description of the generic construction is as follows. Suppose that the standard signature $\text{SS} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ is in \mathbb{C} . The generic IBS scheme GIBS is constructed as follows:

- To generate a master key pair, the KCG runs the key generation algorithm of the signature scheme and outputs the generated public/secret key pair as the master public/secret key pair. To generate a user key pair, the KCG simply signs the identity of the user using his master secret key and outputs the generated signature coupled with the master public key and the identity of the user as the user secret key.
- An identity-based signature is constructed as a signature of knowledge of KGC's signature on the identity of the signer by first applying the corresponding conversion algorithm on input σ (which is contained in the user secret key of the signer) to obtain $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}})$. Then she constructs a proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ and transforms it into a signature of knowledge on m via the Fiat-Shamir transform. The signature is a pair consisting of $\tilde{\sigma}_{\text{aux}}$ and this signature of knowledge. Finally, to verify an identity-based signature σ , one verifies the validity of the

signature of knowledge δ according to the identity of the signer, the master public key, and the value $\tilde{\sigma}_{\text{aux}}$ provided.

The scheme is transcribed in Figure 4.13.

<p>Algorithm GIBS.MKeyGen(k) $(msk, mpk) \leftarrow \text{SS.KeyGen}(k)$ Return (msk, mpk)</p> <hr/> <p>Algorithm GIBS.UKeyGen(msk, id) $\sigma \leftarrow \text{SS.Sign}(msk, id)$ $usk \leftarrow (mpk, id, \sigma)$ Return usk</p>	<p>Algorithm GIBS.Sign(usk, m) $(\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) \leftarrow \text{Convert}(mpk, id, \sigma)$ $\delta \leftarrow \text{SoK}\{\tilde{\sigma}_{\text{pre}} : \text{Valid}(mpk, id, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\}(m)$ $\sigma \leftarrow (\tilde{\sigma}_{\text{aux}}, \delta)$ Return σ</p> <hr/> <p>Algorithm IBS.Verify(mpk, id, m, σ) $d \leftarrow \text{VoK}\{\tilde{\sigma}_{\text{pre}} : \text{Valid}(mpk, id, (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}))\}(m, \delta)$ Return d</p>
---	--

Figure 4.13: Our generic construction of identity-based signatures

This construction is a generalized version of Kurosawa and Heng’s construction [KH04]. They need stronger requirements on their signature schemes. It is also worth mentioning similarities between the idea behind Kurosawa and Heng’s and our constructions and that of Naor’s observation on how to transform any identity-based encryption to a standard signature scheme [BF01, p. 226]: in both, user secret keys are seen as the signature of the KGC on the user identity and vice versa. Our constructions can be seen as the other way of Naor’s observation, i.e. from the non-identity-based world to the identity-based world. A possible result of combining the two ideas is the construction of identity-based signatures from identity-based encryptions.

We propose the following theorem for the security of our construction.

Theorem 4.5.1 *Let SS be a standard signature in \mathbb{C} and P_{SS} be its underlying problem. The construction GIBS based on the signature SS is ID-EUF-CMA-secure if P_{SS} is hard.*

Proof. We prove that the interactive version of our IBS scheme, denoted by GIBI, is an identity-based identification scheme secure against impersonation under passive attacks (i.e. ID-IMP-PA in the sense of Definition def:id-imp-atk on page 26). This completes the proof since Bellare et al. have shown that any IMP-PA-secure IBI is transformed via Fiat-Shamir to an ID-EUF-CMA-secure IBS scheme [BNN04].

To prove IMP-PA security we need to be able to respond to two types of oracle queries: corruption oracle and conversation oracle queries. For the former, a user secret key for the given identity must be simulated and for the latter, a transcript of the interaction between a user with a given identity and a verifier. User secret keys can be simulated via the simulation algorithm for the signature scheme, since user secret keys are simply signatures on user identities. Transcripts of the interaction between a user with a given identity and a verifier can be simulated via first simulating the $\tilde{\sigma}_{\text{aux}}$ and then simulating a transcript for the proof of knowledge of the $\tilde{\sigma}_{\text{pre}}$ corresponding to the master public key,

the identity, and $\tilde{\sigma}_{\text{aux}}$.

At last, the successful impersonator can be used to extract two transcripts with the same $\tilde{\sigma}_{\text{aux}}$ and commitment message and two different challenges and responses to them. This allows first computing the $\tilde{\sigma}_{\text{pre}}$ and then, knowing both $\tilde{\sigma}_{\text{aux}}$ and $\tilde{\sigma}_{\text{pre}}$, computing a forgery for the signature scheme which, in turn, is given to the solution calculator algorithm to compute the solution to the given instance of the underlying problem P_{SS} .

Given an instance of the underlying problem P_{SS} , we run the *Sim* algorithm on this input. *Sim* gives us a pk that we relay to the adversary as the master public key. The adversary then starts to ask two types of oracle queries: corruption oracle queries and conversation oracle queries. On a corruption oracle query id , we forward id as a signing query to *Sim* and get the signature σ on it and then forward it along with the master public key and the input id as the response to the query id (i.e. the user secret key corresponding to id) to the adversary. On a conversation query id , we run the *AuxSim* algorithm on input the master public key and id and get a simulated $\tilde{\sigma}_{\text{aux}}$. Then we run the *TrSim* algorithm for the protocol for proof of knowledge of $\tilde{\sigma}_{\text{pre}}$ on input $(mpk, id, \tilde{\sigma}_{\text{aux}})$ to get a transcript $Tr = (Cmt, Chl, Rsp)$ for that protocol. Then we give the adversary the conversation $((Cmt, \tilde{\sigma}_{\text{aux}}), Chl, Rsp)$ as the response to the query id . For the signature schemes that use a random oracle in their construction, the adversary asks \mathcal{H}_{SS} queries as well. These queries are also relayed to *Sim* algorithm and the response is relayed back to the adversary.

At last, the adversary decides that the first phase is over and outputs a target identity id^* . We keep answering the queries as before in the second phase. The adversary is able to prove knowledge of the user secret key corresponding to id^* at this stage. Rewinding the adversary and asking for a new challenge gives us two transcripts with the same commitment and $\tilde{\sigma}_{\text{aux}}^*$ and different challenges and their respective responses. We are able to extract a $\tilde{\sigma}_{\text{pre}}^*$ corresponding to $\tilde{\sigma}_{\text{aux}}^*$ from the same commitment, different challenges, and their respective responses then, and at last run the *Retrieve* algorithm on input the master public key, id^* , and the pair $(\tilde{\sigma}_{\text{aux}}^*, \tilde{\sigma}_{\text{pre}}^*)$ to get a signature σ^* on the identity id^* . We finally run the *Cal* algorithm on input σ^* to get the solution to the problem instance.

Let us compute the probability that we are successful in solving the underlying problem instance. Let us denote the probability that we are able to successfully simulate the environment for the adversary and the adversary gives us a *suitable* forgery by acc . With a similar reasoning to the proof of Theorem 4.4.4, we get

$$acc(k) \geq \text{Adv}_{\text{Sim(A)}}(k) \cdot \text{Adv}_{\text{GIBI,A}}^{\text{IMP-PA}}(k) \quad .$$

Now, applying the Reset Lemma (see Section 4.2.4) we get the success probability of computing two suitable transcripts as follows

$$res(k) \geq \left(acc(k) - \frac{1}{|ChSp|} \right)^2 \quad .$$

Furthermore, again with a similar reasoning to the proof of Theorem 4.4.4, we are able to calculate the probability that our solution calculator Cal_{SS} is successful in solving the instance of the problem P_{SS} as the following for non-FL-based schemes:

$$\text{Adv}_{\text{Cal}_{\text{SS}}}^{P_{\text{SS}}}(k) \geq \text{Adv}_{\text{Cal(Sim)}} \cdot res(k) \quad .$$

Combining the above equations, we get the following final result for the success probability of our solver Slv of P_{SS} problem instances for non-FL-based schemes:

$$\text{Adv}_{\text{Slv}}^{\text{P}_{\text{SS}}}(k) \geq \text{Adv}_{\text{Cal}(\text{Sim})} \cdot \left(\text{Adv}_{\text{Sim}(\text{A})}(k) \cdot \text{Adv}_{\text{A}(\text{GIBI})}^{\text{IMP-PA}}(k) - \frac{1}{|\text{ChSp}|} \right)^2.$$

This completes the proof for non-FL-based signature schemes.

Furthermore, for FL-based signatures, the probability that the solution calculator is successful in solving the instance of the problem P_{SS} can be written as the following:

$$\text{Adv}_{\text{Cal}_{\text{SS}}}^{\text{P}_{\text{SS}}}(k) \geq \text{Adv}_{\text{Cal}(\text{Sim})} \cdot \left(\text{res}(k) \right)^2.$$

Thus the overall success probability of the solver is calculated as follows for FL-based signature schemes:

$$\text{Adv}_{\text{Slv}}^{\text{P}_{\text{SS}}}(k) \geq \text{Adv}_{\text{Cal}(\text{Sim})} \cdot \left(\text{Adv}_{\text{Sim}(\text{A})}(k) \cdot \text{Adv}_{\text{A}(\text{GIBI})}^{\text{IMP-PA}}(k) - \frac{1}{|\text{ChSp}|} \right)^4.$$

This completes the proof for FL-based signatures. ■

Hierarchical Identities: We observe that our construction of generic IBS schemes has the *nesting* property, meaning that if one extends the definition of class \mathbb{C} to identity-based signature schemes, then the construction **GIBS** belongs to the class \mathbb{C} itself. This is due the fact that a **GIBS** signature in the form $\sigma = (\tilde{\sigma}_{\text{aux}}, (\text{Cmt}, \text{Rsp}))$ can be converted to the converted signature below:

$$\tilde{\sigma} = (\tilde{\sigma}_{\text{aux}}, \tilde{\sigma}_{\text{pre}}) = ((\tilde{\sigma}_{\text{aux}}, \text{Cmt}), \text{Rsp}).$$

For all the signatures listed in Section 4.3.2, knowledge of Rsp can be proved via a Σ protocol. Hence, for all the constructions of IBS schemes from these signatures, the **GIBS** can be *nested* in the way that an identity based signer can act as a new KGC for a new user. This enables construction of *hierarchical* identity-based signature schemes.

Ring of Identity-based Signers: Another possible extension of the **GIBS** schemes is the construction of *identity-based ring signatures* from any signature scheme in \mathbb{C} . To generate a ring signature, the signer constructs a one-out-of- n signature of knowledge of the n user secret keys in the chosen ring, where each user secret key is a signature of the KGC on the corresponding user identity.

Identity-Based Designated-verifier Signatures: A special case of identity-based ring signatures is when the ring size is two. Such a signature is equivalent to an identity-based designated-verifier signature in which the ring consists of the signer and the verifier. hence, we get a construction of **IBDVS** signatures.

4.5.2 Comparison

Our construction provides a unified framework that captures and provides security proofs for many identity-based signature schemes in the literature. All random oracle based schemes that we have found in the literature are instances of our construction! That is, in all the schemes, the user secret

key is in fact a signature on the user identity by the KGC and the identity-based signature on a message is a Fiat-Shamir transformed proof of knowledge of the user secret key. Namely,

- the schemes from [Sha84] and [GQ88] use the RSA-FDH signature to issue user secret keys and different proofs of knowledge to calculate signatures;
- the scheme from [FS86] uses the Rabin signature **Rab** to issue user secret keys;
- the scheme from [Oka92] uses the **OkaS** signature to issue user secret keys and the **OkaP** proof of knowledge to calculate signatures;
- all the schemes from [SOK00, Pat02, Hes02, CC03, Yi03, HCW05] use the BLS signature to issue user secret keys but different proofs of knowledge to generate signatures;
- the BNN-IBS scheme from [BNN04] uses the Schnorr signature **SchS** to issue user secret keys; and finally,
- the scheme from [BLMQ05] uses the ZSS signature to issue user secret keys.

All the new convertible IBS schemes constructed in [BNN04], can be also seen as instances of our construction. Furthermore, as mentioned above, schemes such as that of [Oka92] and the BNN-IBS scheme from [BNN04] which are not captured by Bellare et al's framework, can be seen to follow our more generic construction.

Furthermore, our method provides new constructions based on almost any signature scheme. Four new schemes, based on discrete logarithms, RSA, and pairings, constructed from MEG, CS, and two from BB signatures, are briefly shown in Table 4.6. In the first column, the original scheme is listed. In the second, third, and fourth columns, the resulting IBS scheme is sketched. The IBS schemes are constructed using protocols for proof of knowledge of converted signatures as listed in Tables 4.2 and 4.3. For BB signature, since the scheme admits to both SchP and CCP protocols, both resulting schemes are listed. Note that the original scheme from the first column is used to generate user secret keys as signatures on the user identity. Finally, the security basis of the IBS scheme is listed for each resulting scheme. This basis is equal to P_{SS} as we proved in Theorem 4.5.1. Note that, again we can use techniques similar to those in page 80 to shorten the length of the signature for the first three schemes.

Schemes resulting from our construction have the desirable property of fast *on-line* signature computation. The reason is that, most of the computation needed to generate an identity-based signature in our construction, specially the costly parts, can be carried out without using the actual message to be signed. This allows a signer to precompute the quantities needed to sign messages beforehand in an *off-line* manner. Then, whenever the message to be signed is determined, the signer uses the pre-computed quantities and finalizes the signature computation. We call this the *on-line* phase. In our construction, the on-line computation can be as low-cost as performing one modular multiplication, for instance, when the SchP proof of knowledge is used.

Table 4.6: A sketch of four example new IBS schemes based on discrete logarithms, RSA, and pairings

Sch.	Keys	Signing	Verification	Sec.
MEG	$msk = x$ $mpk = (g, h)$ $h = g^x$ $usk = (r, s)$	Pick random t $Pub \leftarrow mpk \parallel id \parallel r$ $Chl \leftarrow H'(Pub \parallel r^t \parallel m)$ $\sigma \leftarrow (r, Chl, t + s \cdot Chl)$	$Pub' \leftarrow mpk \parallel id \parallel \sigma_1$ $Cmt' \leftarrow \sigma_1^{\sigma_3} \left(\frac{h^{\sigma_1}}{g^{H(id \parallel \sigma_1)}} \right)^{\sigma_2}$ $\sigma_2 \stackrel{?}{=} H'(Pub' \parallel Cmt' \parallel m)$	DL
CS	$msk = (p, q)$ $mpk = (n, h, x, e')$ $n = pq$ $usk = (e, y, y')$	Pick random t $Pub \leftarrow mpk \parallel id \parallel e \parallel y'$ $Chl \leftarrow H'(Pub \parallel t^e \parallel m)$ $\sigma \leftarrow (e, y', Chl, t \cdot y^{Chl})$	$Pub' \leftarrow mpk \parallel id \parallel \sigma_1 \parallel \sigma_2$ $Cmt' \leftarrow \frac{\sigma_4^{\sigma_1}}{\left(x h^{H(\sigma_2' h - H(id))} \right)^{\sigma_3}}$ $\sigma_3 \stackrel{?}{=} H'(Pub' \parallel Cmt' \parallel m)$	Strong RSA
BB	$msk = (x, y)$ $mpk = (g, u_1, u_2)$ $(u_1, u_2) = (g^x, g^y)$ $usk = (\delta, l)$	Pick random z, t $Pub \leftarrow mpk \parallel id \parallel \delta^z \parallel l$ $Chl \leftarrow H'(Pub \parallel e(g, g)^t \parallel m)$ $\sigma \leftarrow (\delta^z, l, Chl, t + z \cdot Chl)$	$Pub' \leftarrow mpk \parallel id \parallel \sigma_1 \parallel \sigma_2$ $Cmt' \leftarrow \frac{e(g, g)^{\sigma_4}}{e(\sigma_1, u_1 g^{id} u_2^{\sigma_2})^{\sigma_3}}$ $\sigma_3 \stackrel{?}{=} H'(Pub' \parallel Cmt' \parallel m)$	SDH
BB	$msk = (x, y)$ $mpk = (g, u_1, u_2)$ $(u_1, u_2) = (g^x, g^y)$ $usk = (\delta, l)$	Pick random t $Pub \leftarrow mpk \parallel id \parallel l$ $Chl \leftarrow H'(Pub \parallel g^t \parallel m)$ $\sigma \leftarrow (l, g^t, \delta^{t+Chl})$	$Pub' \leftarrow mpk \parallel id \parallel \sigma_1$ $Chl' \leftarrow H'(Pub' \parallel \sigma_2 \parallel m)$ $e(\sigma_3, u_1 g^{id} u_2^{\sigma_1}) \stackrel{?}{=} e(\sigma_2 g^{Chl'}, g)$	SDH

4.6 Combined Constructions

The functionalities for which constructions are given in the chapter can be mixed and matched together in an arbitrary fashion based on the application at hand. That is, one can combine our constructions of U(M)DVS, (H)IBS, IBDVS, and IBRS schemes with each other and get new constructions for schemes with combined functionalities.

An example is the construction of *identity-based universal designated verifier signatures* (IBUDVS) from any signature in \mathbb{C} . In such a scheme, a designator wishes to designate a certificate signed by an identity-based signer and the designated verifier is also identity-based. The designated verifier's secret key is a signature on his identity by the KGC. To designate, the designator simply constructs a disjunctive proof of knowledge of (a converted version of) her certificate *or* (a converted version of) the verifier's secret key.

The signature of knowledge based construction guarantees that the non-transferability privacy and non-delegatability requirements are met for these schemes. We expect that a proof of identity-based DV-unforgeability can be also written for such schemes, by combining the ideas used to prove the generic UDVS and IBS schemes secure, but we do not elaborate on it since we consider it out of the scope of this thesis.

In Table 4.7 we compare each existing IBUDVS scheme in the literature with a scheme resulting from applying our construction to the same underlying identity-based signature and designated verifier key family. We assume the same parameters as our previous comparisons. ZSMC₁ and ZSMC₂ schemes are from [ZSMC05] and are respectively based on Cha-Cheon IBS CC [CC03] and Hess IBS Hess

[Hes02]. Cao² scheme is from [CC08] and is based on Paterson-Schuldt IBS PS [PS06]. Note that, for the comparisons to make sense, we did not start from a signature to construct our IBUDVS schemes, rather we constructed our schemes based on the underlying IBS used in each existing scheme. As the table shows, security proofs for our schemes are based on either weaker or similar security assumptions than those of their counterparts.

Table 4.7: Comparison of IBUDVS schemes in the literature with their counterparts based on our construction

IBS	IBUDVS	Assumption	Model	Cost				$ \hat{\sigma} $ (bits)	SI	ND
				Desig			DVer			
				off-line	on-line	total				
CC	ZSMC ₁	BDH	ROM	nil	1 p	1 p	1 p	1195	✗	✗
	GIBUDVS	CDH	ROM	2 p	1 p	3 p	4 p	1153	✓	✓
Hess	ZSMC ₂	?	ROM	3 p	1 p	4 p	4 p	1526	✗	?
	GIBUDVS	CDH	ROM	2 p	1 p	3 p	4 p	2006	✓	✓
PS	Cao ²	CDH	StM	4 e + 160 m	1 e + 80 m	5 e + 240 m	4 p + 240 m	684	✗	✗
	GIBUDVS	CDH	ROM	3 p	1 p	4 p	5 p	1495	✓	✓

4.7 Concluding Remarks

In this chapter we proposed generic constructions for universal designated-verifier signature schemes and identity-based signature schemes and proved security of the constructions. We also extended our generic constructions to the multiple verifier variant of the universal designated-verifier signature schemes, the hierarchical, ring, and designated-verifier variants of the identity-based signature schemes, and combined constructions such as identity-based universal designated-verifier signature schemes.

Our generic construction of universal designated-verifier signature schemes offer fast on-line designation, provable non-delegatability, and signer—verifier setting independence while being comparable to previous schemes in other aspects. Specifically, the setting independence property allows for our constructions to be easily integrated with the existing settings with no additional cost of rekeying. Our generic construction provides new schemes that might be of specific application interest, such as an RSA-only based universal designated-verifier signature scheme which have not been achieved previously. However, our schemes are provably secure in the Random Oracle Model and designing universal designated-verifier signature schemes with any of the above properties, i.e., fast on-line designation, provable non-delegatability, and setting independence, is an open problem.

Our generic construction of identity-based signature schemes can be basically seen as a unifying framework for all the random oracle based schemes that were proposed previously in the literature. This framework not only provides unified intuitive construction and proof methods, but also enables construction of new schemes that might be of interest of specific applications. Our generic construction

is also able to provide schemes with fast on-line signing algorithms in which the computationally costly calculations can be carried out in an off-line manner. This makes those instances of our construction suitable for situations where fast signatures are needed such as real-time applications.

Our generic construction of the universal designated-verifier signature scheme and identity-based signature scheme variants and our combined constructions provide generic recipes for designing schemes that are expected to be provably secure in the Random Oracle Model. Among these constructions, all the universal designated-verifier, designated-verifier, and ring variants provide the setting independence property. All the universal designated-verifier and designated-verifier variants are also expected to be provably non-delegatable. We expect that writing the proofs for these schemes is not hard using the proof techniques we used in this chapter, but we leave it open as a possible extension of our work.

Chapter 5

Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems

5.1 Introduction

Attribute-based encryption is an extension of the notion of identity-based encryption in which user identity is generalized to a set of descriptive attributes instead of a single string specifying the user identity.

Inspired by the recent works on attribute based encryption, we introduce a new signature scheme that we call a *threshold attribute-based signature*, or t-ABS for short. In a t-ABS scheme, a central authority issues secret keys to the signers according to their *attributes*. A signer with an attribute set Att can use his secret key to sign using any subset $A \subset Att$ of attributes. A signature can be verified against a verification attribute set B and verification succeeds if $A \cap B$ has at least size d . Threshold attribute-based signatures have attractive applications that are outlined below.

As their name suggests, t-ABS schemes provide *threshold attribute-based verification*. Consider the following import policy for goods to a country: imports from certain ‘low-risk countries’ can be from any company that has a specific ISO or a particular FAO certificate, but imports from other countries must be from a company with both certificates. This policy can be enforced by verifying if the exporting company has two of the three following attributes: it is in a low-risk country, it is an ISO certificate holder, and it is a FAO certificate holder. Here, a t-ABS scheme with appropriate definition of attributes can be used to implement the verification functionality.

The threshold verification can also be used for applications such as identity-based signatures that use biometric information as user identities to provide strong authentication for users. Here, key generating authorities measure users’ biometrics and issue secret keys to them. Verifiers, measure users’ biometrics independently to obtain their identity in the system. The new measurement will not be exactly the same as the ones collected at the registration phase and hence *fuzzy verification* is required. Fuzzy verification ensures that two reads of the same biometric feature match despite having minor differences. Using a threshold attribute-based signature scheme with suitable choices of mapping and threshold, “close” biometric measurements map to attribute sets for signing and

verification that have sufficient overlap and result in successful verification of signature.

A t-ABS scheme also enables the signer to choose their “signing identity” (relevant attributes) at the time of signing. A professor in the School of Computer Science who is also Head of the School, may need to sign documents as either a Professor of Computer Science, or the Head of the School, or a Professor of Computer Science and the Head of the School. In general a person with n “atomic” identities can sign using an identity that is any of the $O(2^n)$ “combination” identities, assuming the combination is meaningful. This can be implemented using a t-ABS scheme with a signing key of size $O(n)$. Providing the same functionality using standard signatures or identity-based signatures requires $O(2^n)$ signing keys, one for each combined identity. More importantly, the naïve solution of $O(n)$ signing keys of a standard or an identity-based signature, is not secure because signatures can be “mixed and matched” with each other. A group of signers, for example, can collude to produce a signature corresponding to the union of their identities. A t-ABS scheme provides *collusion resistance*: that is, no group of colluding signers can produce a signature that could not be generated by a single member of the group.

A t-ABS scheme also enables *signature holders* to present for verification a signature that corresponds to a subset of signer’s attributes. A document signed by John Smith, a member of the 2008 Executive Committee of the Workers’ Union (WU), can be presented as signed by an Executive Committee member of the WU, without revealing the signer’s name or the year. The same document can also be presented as signed by a 2008 Executive Committee member of the WU, or simply by John Smith. This functionality gives the signature holder the flexibility to present the same signature in different ways at different occasions. To provide the same functionality, using a standard or an identity-based signature scheme, $O(2^n)$ signatures are required, each produced using one of the aforementioned $O(2^n)$ keys. Using a t-ABS scheme however, requires a signing key of size $O(n)$ and a signature of size $O(n)$ bits¹.

A t-ABS scheme supports *signer-attribute privacy* and in well-designed systems can provide privacy for users’ attribute. Consider a scenario where participation in a poll requires proof of residency in one of the 27 counties in a state. Assume these proofs are cards that are signed by either the local government or police authorities of the counties and are given to the long-time residents and foreign workers, respectively. For privacy reasons, card holders may want to be able to prove that they own a card but protect their other details. Proving possession of such a card is equivalent to proving possession of a card issued by one of the 2×27 possible entities. Using standard or identity-based signatures, a voter can prove possession of a valid signature with respect to one of the 2×27 public keys or identities through a proof of size 2×27 (see *proofs of partial knowledge* [CDS94]). A t-ABS can provide a proof of size $2+27$ and ensures that the attributes of the signer (and hence those of the card holder) are not revealed. Signers in such schemes will each have two attributes: one corresponding to ‘police’ or ‘local government’, and the other representing one of the 27 counties. Verification of residency is then equivalent to requiring that a card is verifiable with a threshold of $d = 2$ attributes from the set of all $2+27$ attributes above. This is because no signer can simultaneously be both police and government, or from two (or more) counties.

¹We note that in certain cases it may not be desirable to be able to break up the attribute set during signing a message or showing a signature, in which case traditional methods based on standard or identity-based signatures could be used.

The above example can be extended to the more general case where the universal set of attributes \mathbb{U} can be partitioned into n subsets \mathbb{U}_i for $i \in N = \{1, 2, \dots, n\}$ such that each signer has at most one attribute from each set. Now suppose providing service to signature holder requires the signer to possess an i -th attribute in the set B_i , for $i \in I \subseteq N$. Verification of such a property is equivalent to verifying if the signer has at least $|I| = d$ of the attributes in the attribute set \mathbb{U} . A t-ABS scheme can be used to implement the above system, with a verification cost of $O(\sum_{i \in I} |B_i|)$. However, the number of possible accepting issuers for such a verification policy and hence the cost of a proof of partial knowledge is $O(\prod_{i \in I} |B_i| \cdot \prod_{i \in N \setminus I} |\mathbb{U}_i|)$.

The above functionalities of t-ABS schemes become more attractive in cases where it is practically infeasible for the signature holder and the verifier to know all possible signers. For example, when a user wants to show that their document is signed by a notary public. It is impossible in practice for a verifier to know all the notary publics and thus standard or identity-based signatures cannot be used. Using a t-ABS however, allows the verifier to easily check if the signer has the attribute **notary public**.

As a prime example to show their application range, we show that t-ABS schemes, when augmented with additional protocols for signing and proof of signature ownership, can be used to construct *attribute-based anonymous credential systems*. In Section 5.5 we show that such credential systems inherit the above properties of t-ABS and provide a number of attractive features including *attribute privacy*, which is an essential property in anonymous credential systems, since verifying credentials against specific public keys reveals unnecessary information about the users and contradicts the “raison d’être” of such systems.

5.1.1 Our Contributions

We first introduce a new type of signature called *threshold attribute-based signature*, or t-ABS for short. In a t-ABS, a signer has an attribute set Att and receives a signing key from a key generating authority based on the set Att . The signer can sign a document using any subset $A \subset Att$ of attributes. A signature is successfully verified against an attribute set B as long as the signing set A and the verification set B have an intersection of size at least d . In other words, the verification succeeds if the attribute set used in the signature is sufficiently ‘close’ to the attribute set used for verification. We construct a t-ABS scheme in Section 5.3 and prove its security against selective forgery in the standard model based on the CDH assumption (see Definition 2.2.4 on page 17). We discuss how security and efficiency of the scheme can be improved and give a secure construction against existential forgery in the standard model.

We then introduce an algorithm for *converting* a signature and an interactive protocol for signature verification. The convert algorithm enables the signature holder to construct a signature that is verifiable with the attribute set $A \cap B$ and prevents the verifier from learning the signer’s attributes that are outside $A \cap B$. The interactive verification protocol enables the signature holder to prove possession of a valid signature without allowing the verifier to learn even $A \cap B$. The two protocols, depending on the requirements of the application, can be used to provide two levels of *signer-attribute privacy*. The weak privacy level is achieved by providing the verifier with the converted signature and

guarantees that the verifier does not find out any signer attribute outside $A \cap B$. The full privacy level is achieved by first converting the signature and then proving possession of the signature using the interactive verification protocol. This guarantees that the verifier learns nothing more than the fact that $|A \cap B| \geq d$. We denote a t-ABS with an efficient conversion algorithm and an efficient interactive verification protocol by t-ABS⁺. We provide efficient conversion and interactive verification for our proposed basic scheme and prove that it provides both levels of signer-attribute privacy. We also show how signatures in the second construction can be efficiently converted and interactively verified and similarly prove our second construction to be fully signer-attribute private.

A t-ABCS scheme consists of a t-ABS⁺ scheme, a commitment scheme, and three additional protocols for (i) obtaining a signature on a committed value, (ii) proving knowledge of a signature, and (iii) proving knowledge of a signature on a committed value. We give security definitions for t-ABCS schemes and a concrete efficient construction that is based on our basic t-ABS scheme and prove security of the construction. Consider a user with a secret key SK_U that has established a pseudonym N_{UI} with an issuer I and another pseudonym N_{UV} with a verifier V . Assume that these pseudonyms are both commitments to SK_U , but under different randomnesses. A t-ABCS scheme can be used by U to, on the pseudonym N_{UI} , obtain a credential that is associated with a signing attribute set A_I , and later provide proof of possession for the credential on the pseudonym N_{UV} against a verification attribute set B_V if $|A_I \cap B_V| \geq d$. The verifier cannot deduce any information about the d attributes in common between the signing attribute set A_I and the verification attribute set B_V ; hence the privacy of the attributes presented by the credential holder is guaranteed.

We define threshold attribute-based anonymous credential systems (t-ABACS) and formalize their security using the *simulation paradigm* that is commonly used for defining security of multiparty protocols. The approach uses a comparison of a real system model with an ideal system model. The ideal model for the t-ABACS captures the security and privacy properties of an anonymous credential system. We show how t-ABCS schemes can be used to realize this model. That is, we give a concrete t-ABACS system and prove its security in the sense that it remains indistinguishable from an ideal system from the viewpoint of any polynomial-time adversary. This results in a concrete t-ABCS scheme with security based on the CDH problem.

5.1.2 Related Work

Attribute-based encryption (ABE) was introduced by Sahai and Waters as an extension of identity-based encryption where each identity is considered as a set of descriptive attributes [SW05]. The scheme, called fuzzy identity-based encryption, allows a threshold attribute-based decryption of encrypted data as follows. Messages can be encrypted by specifying a set of decryptor attributes ω' during encryption. Such a ciphertext then can be decrypted by any user with the attribute set ω such that $|\omega \cap \omega'| \geq d$. Our attribute-based signature scheme can be viewed as the signature counterpart of Sahai and Waters's encryption scheme. In our scheme, a signer has a set of attributes A and the verifier specifies a verification attribute set B . A signature is verified as valid if $|A \cap B| \geq d$.

Attribute-based signatures extend the *identity-based signature* of Shamir [Sha84] by allowing identity of a signer to be a set of descriptive attributes rather than a single string representing the signer's

identity. A t-ABS provides threshold attribute-based verification. That is, the verification succeeds if at least d of the attributes specified by the verifier are the same as those of the signer. Thus, identity-based signature can be seen as a specific case of our schemes with identity size and threshold both equal to one.

Independent of our work, there have been other attempts to define and realize *attribute-based signatures* [Kha07b, Kha07a, YCD08, Kha08, GZ08, MPR08, LK08]. The schemes of [YCD08] and [GZ08] are direct applications of the known transform from identity-based encryptions to identity-based signatures [GS02]. In both works, authors do not consider any notion of privacy. The works of [Kha07b, Kha07a] and [LK08] capture weaker notions of anonymity where signers only remain anonymous within the group of entities possessing the same attributes and the verifiers must know which attributes are used to sign a message to be able to verify. Khader [Kha08] and Maji et al. [MPR08] treat attribute-privacy as a fundamental requirement of attribute-based signatures and provide schemes that provably satisfy this requirement for threshold trees and monotone boolean verification policies, respectively. However, both works have shortcomings that make their solutions unsuitable for the scenarios outlined in the introduction. Both schemes require the signer to know the verification policy at the time of signing. This is a major limitation for their proposed scheme, particularly in applications where other than the signer and the verifier, there are also ‘signature holders’. An example of such applications is a credential system where a credential holder needs to satisfy different verification policies depending on the occasion, and it is important for efficiency and useability purposes not to require different credentials for each verification policy. Also, security proofs of Khader and Maji et al. are in the random oracle or generic group models, while all our proofs are in the standard model and use the well-known assumption of computational Diffie-Hellman. Finally, none of the previous attribute-based signatures had been extended to credential systems.

Anonymous credential systems (a.k.a. *pseudonym systems*) were introduced by Chaum [Cha85, CE87] and more recently further formalized and studied in [LRSW99, Lys02]. A credential in such systems is issued and verified on a user pseudonym, which in turn is bound to the user’s secret identity. Users remain anonymous since their pseudonyms hide their secret identity. Besides, transactions involving the same user remain unlinkable.

5.2 Notation and Preliminaries

We use “ $A \dashv (X) \rightarrow B \mid C$ ” to denote that A sends X to B if condition C holds. The condition can be complex and include logical connectors. We use the proof of knowledge notation originated in [CS97]. For instance, “ $\text{ZK-PoK}\{(x, y, z) : a = g^x h^y \wedge b = c^y d^z\}$ ” denotes a zero knowledge proof of knowledge of (x, y, z) such that $a = g^x h^y$ and $b = c^y d^z$, where a, g, h, b, c , and d are public inputs to the protocol.

Lagrange interpolation for a polynomial $q(\cdot)$ over \mathbb{Z}_p of order $d - 1$ and a set $S \subset \mathbb{Z}_p$ with size $|S| = d$ is calculated as $q(x) = \sum_{i \in S} q(i) \Delta_{i,S}(x)$, where

$$\Delta_{i,S}(x) \triangleq 1 \cdot \prod_{j \in S, j \neq i} \frac{x - j}{i - j} \quad \text{for all } i \in S \quad (\text{and extendedly for all } i \in \mathbb{Z}_p) .$$

We use zero knowledge proofs of knowledge for conjunctive statements about discrete logarithms. Efficient versions of such proofs can be found in the literature. e.g., in the work by Cramer et al. [CDM00].

A *commitment scheme* consists of two algorithms **CmtKeyGen** and **Commit**, where the former is the key generation algorithm that generates a commitment public key cpk on input a security parameter and the latter is the commitment algorithm that on input a commitment public key cpk , a message to be committed m , and a random element r generates a commitment C on m . The scheme is said to be *hiding* if C hides the message m , that is, given cpk and C , it is hard to obtain any information about m . The scheme is said to be *binding* if C binds the committer to m , that is, Given cpk , it is hard to come up with two different pairs (m, r) and (m', r') such that $\text{Commit}(cpk, m, r) = \text{Commit}(cpk, m', r')$. The scheme is said to be secure if it has both of the above properties.

Camenisch and Lysyanskaya have shown that a tuple consisting of a signature, a commitment scheme, and efficient protocols for issuing and verifying signatures on committed values, is sufficient for realizing anonymous credential schemes [CL01, Lys02]. We denote such tuples by *C-signatures*.

A well-known paradigm that is used to define security of cryptographic protocols is the simulation (a.k.a. “ideal vs. real”) paradigm, originating from [GMW91]. The intuition behind such definitions is that a real system is secure if it emulates a certain ideal system designed to trivially guarantee security properties expected from the system. To formally define security using this paradigm, one first defines a protocol-independent *framework*, which contains the model of computation and communication and a notion of *emulation*. The former determines the types of entities involved in the real-life and ideal protocol execution, what each of them can see and do, and how they communicate. The latter defines what it means to say a real protocol execution emulates (i.e., is as secure as) the ideal protocol execution. Then, an *ideal model* specific to the protocol is defined to capture the expected security properties. Security is guaranteed by including a trusted entity that collects the inputs of different entities in each round, performs the necessary calculations locally, and hands each entity’s outputs for that round back to them. We use this paradigm to define a framework for analyzing security of our credential system.

5.3 Threshold Attribute-Based Signatures

We assume there is a universal set of attributes \mathbb{U} that is publicly known. Each signer is associated with a subset $Att \subset \mathbb{U}$ of attributes that is verified by a central authority. In the following, we assume that the signing attribute set A is equal to the signer attribute set Att and we use the terms interchangeably. This has minimal impact in our definitions of the schemes and their security. We define the signature for a fixed threshold d . We discuss later how flexible thresholds can be achieved.

5.3.1 Definition

A threshold attribute-based signature (t-ABS) is a quadruple of algorithms as follows:

Setup is the algorithm run by a central authority on input the security parameter k and outputs a master secret key msk and a master public key mpk .

KeyGen is the algorithm run by the central authority on inputs msk and a set of signer attributes A and generates a secret signing key ssk for the signer.

Sign is the algorithm run by a signer on inputs ssk and a message m and generates a signature σ on the message.

Verify is the algorithm run by a verifier on inputs mpk , a message signature pair (m, σ) , and a verification attribute set B and outputs 1 if σ is a valid signature by a signer who has at least d of the attributes in B , i.e., if $|A \cap B| \geq d$.

Correctness. A t-ABS scheme has to satisfy the correctness property, i.e., a signature generated by a signer with attributes A must pass the verification test for any B if $|A \cap B| \geq d$.

Unforgeability. For a t-ABS scheme defined as above we require that it is *existentially unforgeable* against *chosen message and attribute set attacks*. In particular, we define the following game between a challenger and the adversary.

Setup Phase: The challenger runs the **Setup** algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α , and
- a signature of a signer with any attribute set of its choice α on a message of its choice m .

Forgery Phase: The adversary outputs a triplet (μ, σ, β) , consisting of a message μ , a forged signature σ , and a verification attribute set β , and wins if σ is a valid signature with respect to (mpk, μ, β) and

- for all queried sets of attributes α , we have $|\alpha \cap \beta| < d$, and
- for all queried pairs (α, m) , we have $m \neq \mu$ or $|\alpha \cap \beta| < d$.

If no polynomial adversary has a considerable advantage in the above game, we say that the t-ABS scheme is existentially unforgeable against chosen message and attribute set attacks, or EUF-CMAA-secure for short. We also consider a weaker notion of security, *selective unforgeability* against chosen message and attribute set attacks (SUF-CMAA-security) in which the adversary should commit to the target forgery message and verification attribute set in the beginning of the attack. We define this notion in the following and discuss how an existentially unforgeable scheme can be constructed given a selectively unforgeable scheme. In particular, we define the following game between a challenger and the adversary.

Initiation Phase: The adversary declares a pair $(\tilde{\mu}, \beta)$ containing the message and the verification attribute set on which it wants to make a forgery.

Setup Phase: The challenger runs the **Setup** algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α as long as $|\alpha \cap \beta| < d$, and
- a signature of a signer with any attribute set of its choice α on a message of its choice \tilde{m} as long as $\tilde{m} \neq \tilde{\mu}$ or $|\alpha \cap \beta| < d$.

Forgery Phase: The adversary outputs a forged converted signature $\tilde{\sigma}$ on the message $\tilde{\mu}$ and the verification attribute set β and wins if $\text{Verify}(\text{mpk}, \tilde{\mu}, \tilde{\sigma}, \beta) = 1$.

If no polynomial adversary has a considerable advantage in the above game, we say that the t-ABS scheme is selectively unforgeable against chosen message and attribute set attacks, or SUF-CMAA-secure for short.

It is easy to see that any SUF-CMAA-secure t-ABS scheme is a EUF-CMAA-secure scheme, but the reduction is not efficient. The corresponding security penalty is linear in the product of the sizes of the message and verification attribute set spaces. To see this, consider a successful EUF-CMAA adversary for the scheme. One can use this adversary to SUF-CMAA-break the scheme as follows. First guess the adversary's choice of $(\tilde{\mu}, \beta)$ and declare it as the selected forging target. Then run the adversary. If the guess is correct, the SUF-CMAA attack succeeds and the probability of this event is one over the number of possible messages times the number of possible verification attribute sets. Using the Random Oracle Model [BR93], the security penalty can be reduced to the maximum number of oracle queries the adversary can make. In this case, the random oracle only needs to be 'programmed' in one point, making sure that the forgery target $(\tilde{\mu}, \beta)$ provided by the EUF-CMAA adversary is mapped to the guessed pair. A similar situation in the design of identity-based encryption schemes is elaborated by Boneh and Boyen in [BB04a].

Collusion Resistance. It is important to note that the above definition of unforgeability guarantees *collusion resistance* in the sense that no colluding group of users can generate a signature that is not generable by one of the colluders. This is because if a group of signers can construct a signature that none of them could have individually produced, then this is a forgery as per the above definition. Note that the adversary is able to query for secret keys for entities with chosen attributes, hence the game captures the scenario in which multiple signers are colluding. Furthermore, for the adversary to win, we must have $|\alpha \cap \beta| < d$ for *all* queried sets of attributes α . This ensures that none of the colluders could generate the forged signature by himself.

5.3.2 Additional Protocols

A signature holder can always check a signature σ against possible verification attribute sets to deduce information about the signer's attributes. To preserve privacy of signers we equip our t-ABS scheme with an additional algorithm for *converting* the signature to another signature that is verifiable against B and only reveals the d chosen attributes of the signer. The *converted signature* can be seen as a B -designated signature that contains a minimal subset of attributes from the original set of signer attributes, that allows the verification to succeed. Such a converted signature reveals only the verifier's chosen attribute set. Attribute privacy is obtained by using an *interactive verification protocol* $i\text{Verify}$, that allows the signature holder to prove possession of a valid converted signature without revealing the chosen d attributes in common between A and B .

We call our t-ABS scheme equipped with the above algorithm and protocol a t-ABS⁺ scheme, which contains the **Setup**, **KeyGen**, and **Sign** algorithms as defined in the t-ABS scheme plus the following two:

Convert is the algorithm run by a signature holder on inputs mpk , a message signature pair (m, σ) , and a verification attribute set B and generates a converted signature $\tilde{\sigma}$ on the message.

CvtVerify is the algorithm run by a verifier on inputs mpk , a message converted-signature pair $(m, \tilde{\sigma})$, and a verification attribute set B and outputs 1 if $\tilde{\sigma}$ is a valid converted signature by a signer who has at least d of the attributes in B , i.e., if $|A \cap B| \geq d$.

iVerify is an interactive verification protocol for proving knowledge of a converted signature on the *prover* side and verifying a converted signature on the *verifier* side. The public inputs are the t-ABS master public key mpk , a message m , and verifier's verification attribute set B . Prover's private input is a converted signature $\tilde{\sigma}$. The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B .

Correctness. Apart from correctness of the underlying t-ABS scheme, we require that any converted signature calculated from a valid signature (i) passes the **CvtVerify** test, and (ii) makes the verifier in the **iVerify** protocol accept.

Unforgeability. We require that converted signatures in our t-ABS⁺ scheme are also existentially unforgeable under chosen message and attribute set attacks. Since knowledge of a signature is sufficient for producing a converted signature (using algorithm **Convert**), converted signature unforgeability implies signature unforgeability, and hence, is a stronger notion of security. Converted signature existential unforgeability under chosen message and attribute set attacks (C-EUF-CMAA-security) is defined through a game, in which the setup and query phases are the same as the EUF-CMAA-security game above, but σ in the forgery phase is replaced with $\tilde{\sigma}$. That is, the adversary is given the same resources, but is expected to forge a nontrivial valid *converted* signature instead of a nontrivial valid signature. The C-EUF-CMAA-security game is defined as follows.

Setup Phase: The challenger runs the **Setup** algorithm and gives mpk to the adversary.

Query Phase: The adversary is allowed to ask queries for the following:

- a secret key of a signer with attributes of its choice α , and
- a signature of a signer with any attribute set of its choice α on a message of its choice m .

Forgery Phase: The adversary outputs a triplet $(\mu, \tilde{\sigma}, \beta)$, consisting of a message μ , a forged converted signature $\tilde{\sigma}$, and a verification attribute set β , and wins if $\tilde{\sigma}$ is a valid converted signature with respect to (mpk, μ, β) and

- for all queried sets of attributes α , we have $|\alpha \cap \beta| < d$, and
- for all queried pairs (α, m) , we have $m \neq \mu$ or $|\alpha \cap \beta| < d$.

The C-SUF-CMAA-security game can be defined accordingly.

Weak Signer-Attribute Privacy. A converted signature should not reveal any attribute of the signer other than the d of them common with B chosen by the signature holder at the time of conversion. Thus, we require that whatever a verifier can deduce about other attributes of the signer given a converted signature, can also be deduced given merely the d attributes as well. This ensures that only the d attributes of the signer that are chosen by the signature holder are revealed to the verifier given a converted signature. We call this property *weak signer-attribute privacy*.

Signer-Attribute Privacy. We also require that the iVerify protocol is a zero knowledge proof of knowledge of a valid converted signature with respect to the public inputs (mpk, m, β) . This ensures that (i) only provers in possession of a valid converted signature are indeed successful in proving so, and (ii) the proof reveals no information other than the validity of the prover's converted signature to the verifier. We call this property *(full) signer-attribute privacy*. Note that property (ii) guarantees that proofs of possession of signatures from different signers satisfying the verification policy remain indistinguishable for the verifier. Furthermore, it guarantees that multiple proofs of possession of even the same signature (from the same signer) remain unlinkable for the verifier.

Flexible Threshold. To achieve a flexible threshold, one can use either or a combination of the following two techniques based on the application at hand: (i) designing multiple schemes with different thresholds and (ii) using *dummy* attributes. The latter is specifically useful to enable thresholds less than the scheme threshold d and it works as follows. The key generation authority assumes that all signers in the system possess some dummy attributes and generates their signing key components accordingly. To enforce a verification policy with threshold $d' \leq d$, a verifier includes $d - d'$ dummy attributes in its verification attribute set. Possessing at least d attributes out of the “new” verification attribute set is equivalent to possessing at least d' attributes out of the “original” verification attribute set, since the $d - d'$ dummy attributes are possessed by all signers. Hence, a range of thresholds $[d_{\min}..d_{\max}]$ can be supported by a scheme with $d_{\max} - d_{\min}$ dummy attributes at the price of increasing the size of all signing keys and signatures by $d_{\max} - d_{\min}$ components.

5.3.3 Constructions

We propose a threshold attribute-based signature based on bilinear maps. We make use of some design techniques of earlier works of [SW05] and [BB04a]. Signer attributes are assumed to be sets of at most n elements of \mathbb{Z}_p . Although generally, identities can be sets of at most n arbitrary strings and a collision resistant hash function is used to map the strings to elements of \mathbb{Z}_p . We use $N = \{1, 2, \dots, n + 1\}$ to denote the set of possible attributes. In the following, a basic scheme with a fixed threshold d is introduced. We will later discuss how to extend our scheme for verifiers with different thresholds. Let $\mathbb{G}_1 = \langle g \rangle$ be a group of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be defined. Let $(e, \mathbb{G}_1, \mathbb{G}_2)$ be of public knowledge. We present the scheme for signing messages in \mathbb{Z}_p . Although, the message space can be expanded to contain arbitrary messages using a collision resistant hash function to map strings to \mathbb{Z}_p .

Setup(1^k): Pick y randomly from \mathbb{Z}_p and set $g_1 = g^y$. Pick random elements $g_2, h, t_1, t_2, \dots, t_{n+1}$

from \mathbb{G}_1 . Define and output the following:

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} \quad \text{and} \quad msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, t_1, t_2, \dots, t_{n+1}, h)$$

KeyGen(msk, A): Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0) = y$, choose random elements r_i in \mathbb{Z}_p for $i \in A$, and output

$$ssk = \left\langle \left\{ g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \right\}_{i \in A} \right\rangle$$

Sign(ssk, m): Parse the signing key as $ssk = \langle \{ssk_{1i}, sk_{2i}\}_{i \in A} \rangle$, pick random elements s_i in \mathbb{Z}_p for all $i \in A$, and output

$$\sigma = \left\langle A, \left\{ sk_{1i} (g_1^m \cdot h)^{s_i}, \quad sk_{2i}, \quad g^{s_i} \right\}_{i \in A} \right\rangle$$

Verify(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1) \quad (5.1)$$

Correctness. One can easily check the correctness of the scheme by verifying that the S components of the signature *interpolate* to result in $e(g_2, g_1)$. If σ is a correctly produced signature, we have:

$$\begin{aligned} \prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} &= \prod_{i \in S} \left(\frac{e(g_2^{q(i)} T(i)^{r_i} (g_1^m \cdot h)^{s_i}, g)}{e(T(i), g^{r_i}) \cdot e(g_1^m \cdot h, g^{s_i})} \right)^{\Delta_{i,S}(0)} \\ &= \prod_{i \in S} \left(e(g_2, g)^{q(i)} \right)^{\Delta_{i,S}(0)} = e(g_2, g)^{\sum_{i \in S} q(i) \Delta_{i,S}(0)} \\ &= e(g_2, g)^y = e(g_2, g_1) \end{aligned}$$

Thus, Equation 5.1 holds.

Unforgeability. Unforgeability is implied by converted-signature unforgeability that we prove later in Theorem 5.3.1.

Additional Protocols. The concrete conversion and converted-signature verification algorithms are as follows:

Convert(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{aligned} \text{for all } i \in S: \quad & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B \setminus S}(0)} \\ \text{for all } i \in B \setminus S: \quad & \tilde{\sigma}_{1i} \leftarrow (T(i) g_1^m h)^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} \end{aligned}$$

and output $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$.

CvtVerify($mpk, m, \tilde{\sigma}, B$): Parse the converted signature as $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$. Check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1) \quad (5.2)$$

Furthermore, the iVerify protocol flow is as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s'_i and r'_i for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.

$$\check{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r'_i} (g_1^m \cdot h)^{s'_i} \quad \text{and} \quad \check{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r'_i} \quad \text{and} \quad \check{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s'_i}$$

2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \check{\sigma}_{1i}^{1/\tau_i}$ and sends $\langle \{\hat{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$ to the verifier.
3. Both the signature holder and the verifier calculate the following for all $i \in B$:

$$\begin{aligned} u_0 &\leftarrow e(g_2, g_1) & u_{1i} &\leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} \\ u_2 &\leftarrow \prod_{i \in B} e(T(i), \check{\sigma}_{2i})^{\Delta_{i,B}(0)} & u_3 &\leftarrow \prod_{i \in B} e(g_1^m h, \check{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{aligned}$$

4. The signature holder performs the following ZK-PoK for the verifier:

$$\text{ZK-PoK} \left\{ (\{\tau_i\}_{i \in B}) : \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_3 \right\}$$

Correctness. The t-ABS⁺ scheme is correct since (i) the t-ABS scheme is correct, and (ii) the S components of the converted signature interpolate to result in $e(g_2, g_1)$ and the $B \setminus S$ components are *dummy* components, and (iii) the equation checked in the iVerify protocol is only a rearranged version of what is checked in the CvtVerify algorithm.

From the definition of the Lagrange coefficient, for any subset S of B and all $i \in B$ we have:

$$\Delta_{i,B}(x) = \Delta_{i,S}(x) \cdot \Delta_{i,B \setminus S}(x)$$

Considering this property, if $\tilde{\sigma}$ is a correctly produced converted signature, for the components in S we have:

$$\begin{aligned} \prod_{i \in S} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} &= \prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,B}(0)/\Delta_{i,B \setminus S}(0)} \\ &= \prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} \\ &= e(g_2, g_1) \end{aligned}$$

Furthermore, for the components of the converted signature not in S we have:

$$\begin{aligned}
\prod_{i \in B \setminus S} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m g_3^r \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} &= \prod_{i \in S} \left(\frac{e(T(i) g_1^m g_3^r h, g)}{e(T(i), g) \cdot e(g_1^m g_3^r \cdot h, g)} \right)^{\Delta_{i,B}(0)/\Delta_{i,B \setminus S}(0)} \\
&= \prod_{i \in S} \left(\frac{e(T(i) g_1^m g_3^r h, g)}{e(T(i), g) \cdot e(g_1^m g_3^r \cdot h, g)} \right)^{\Delta_{i,S}(0)} \\
&= \prod_{i \in S} 1^{\Delta_{i,S}(0)} = 1
\end{aligned}$$

Combining the two results above shows that the verification equation, i.e., Equation 5.2, holds.

Furthermore, if $\tilde{\sigma}$ is a correctly produced converted signature, one can easily check that the equation

$$\prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{33}$$

in the iVerify protocol, is just a rearrangement of Equation 5.2 and hence it holds.

Unforgeability. We prove the following theorem.

Theorem 5.3.1 *The above t -ABS⁺ scheme is C-SUF-CMAA-secure if the CDH problem is hard. As a direct corollary, the underlying t -ABS scheme is SUF-CMAA-secure if the CDH problem is hard.*

Proof. The proof can be derived from the proof of the Theorem 5.4.1 that comes next by setting $r = \rho = 0$. Note that in that case, there would be no type 2 forgery in the following proof, thus the part dealing with a type 2 forger can be ignored. We omit the proof to avoid repetition. ■

Existential unforgeability can be achieved in any of the following techniques:

General Reduction: Any selectively unforgeable t-ABS can be proved to be existentially unforgeable. This general reduction is discussed in page 99. This reduction is not efficient and introduces a large penalty factor to the security of the scheme, but it is carried out in the standard model.

Random Oracles: An alternative method is use random oracles to hash messages and signer attributes at the time of signing. This method is discussed in page 99 and provides an efficient reduction. The penalty factor here is substantially lower than that of the general reduction.

Waters's Technique: Waters proposed a technique [Wat05] that can be used here to achieve a *tight* existential unforgeability reduction in the standard model at the price of large public keys. To use this technique, we need to (i) add ℓ random elements h_1, h_2, \dots, h_ℓ from \mathbb{G}_1 to mpk in Setup algorithm, where ℓ is the (maximum) bit length of messages, and (ii) replace all instances of $g_1^m h$ in the above scheme with $W(m)$, where Waters function $W(\cdot)$ is defined as $W(m) = h \prod h_i^{m_i}$, where m_i denotes the i -th bit of m . The concrete scheme is transcribed in page 105 for completeness.

Of course, Waters's technique is preferred to the other two. However, we use the algebraic properties of our basic scheme to construct C-signatures in the next section and we do not know if C-signatures can be constructed based on the Waters modification of our basic t-ABS scheme. Thus, our construction of C-signatures admits only to the first two techniques.

Signer-Attribute Privacy. One can see that since $B \setminus S$ components of the converted signature are publicly simulatable, weak signer-attribute privacy is achieved. Furthermore, the iVerify protocol can be easily proved to be both a proof of knowledge and zero knowledge and hence full signer-attribute privacy is also achieved. We prove the following theorem.

Theorem 5.3.2 *The above $t\text{-ABS}^+$ scheme achieves both weak and full signer-attribute privacy.*

Proof. It is easy to see that the converted signature cannot reveal more than the d selected attributes, since it simply does not include any components corresponding to other attributes. The proof of our iVerify protocol being a zero knowledge proof of knowledge of a converted signature is a simpler version of the proof of the same properties for iHVerify and iCVerify protocols. The latter proof is part of the proof of Theorem 5.4.2 which comes next. We omit the proof to avoid repetition. ■

Alternatively, random oracles can be used to hash the attributes and reduce the security penalty substantially. However, we cannot use random oracles to hash the message, since as we see later, we will use the algebraic properties of our scheme to construct C-signatures and using a random oracle would not allow that.

Efficiency. The iVerify protocol is of size linear in the size of the verification attribute set. This is in contrast with the discussed partial proofs of knowledge, which require proofs of size linear in the number of possible signers.

A Standard-Model Existentially-Unforgeable Scheme. Signer attributes are assumed to be sets of at most n elements of \mathbb{Z}_p . Although generally, identities can be sets of at most n arbitrary strings and a collision resistant hash function is used to map the strings to elements of \mathbb{Z}_p . We use $N = \{1, 2, \dots, n+1\}$ to denote the set of possible attributes. Let $\mathbb{G}_1 = \langle g \rangle$ be a group of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be defined. Let $(e, \mathbb{G}_1, \mathbb{G}_2)$ be of public knowledge. We present the scheme for signing messages with (maximum) bit length ℓ . Although, the message space can be expanded to contain arbitrary messages using a collision resistant hash function to map strings to the set of binary strings with (maximum) length ℓ . Waters function $W(\cdot)$ is defined as $W(m) = h \prod h_i^{m_i}$, where m_i denotes the i -th bit of m .

Setup(1^k): Pick y randomly from \mathbb{Z}_p and set $g_1 = g^y$. Pick random elements $g_2, h, t_1, t_2, \dots, t_{n+1}, h_1, h_2, \dots, h_\ell$ from \mathbb{G}_1 . Define and output the following:

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} \quad \text{and} \quad msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, t_1, t_2, \dots, t_{n+1}, h)$$

KeyGen(msk, A): Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0) = y$, choose random elements r_i in \mathbb{Z}_p for $i \in A$, and output

$$ssk = \left\langle \left\{ g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \right\}_{i \in A} \right\rangle$$

Sign(ssk, m): Parse the signing key as $ssk = \langle \{ssk_{1i}, skk_{2i}\}_{i \in A} \rangle$, pick random elements s_i in \mathbb{Z}_p for all $i \in A$, and output

$$\sigma = \left\langle A, \left\{ skk_{1i} W(m)^{s_i}, \quad skk_{2i}, \quad g^{s_i} \right\}_{i \in A} \right\rangle$$

Verify(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(W(m), \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1)$$

The concrete conversion and converted-signature verification algorithms are as follows:

Convert(mpk, m, σ, B): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{aligned} \text{for all } i \in S: \quad & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B \setminus S}(0)} \\ \text{for all } i \in B \setminus S: \quad & \tilde{\sigma}_{1i} \leftarrow (T(i)W(m))^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} \end{aligned}$$

and output $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$.

CvtVerify($mpk, m, \tilde{\sigma}, B$): Parse the converted signature as $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$. Check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(W(m), \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1)$$

Furthermore, the iVerify protocol flow is as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s'_i and r'_i for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.

$$\check{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r'_i} W(m)^{s'_i} \quad \text{and} \quad \check{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r'_i} \quad \text{and} \quad \check{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s'_i}$$

2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \check{\sigma}_{1i}^{1/\tau_i}$ and sends $\langle \{\hat{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$ to the verifier.
3. Both the signature holder and the verifier calculate the following for all $i \in B$:

$$\begin{aligned} u_0 &\leftarrow e(g_2, g_1) & u_{1i} &\leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} \\ u_2 &\leftarrow \prod_{i \in B} e(T(i), \check{\sigma}_{2i})^{\Delta_{i,B}(0)} & u_3 &\leftarrow \prod_{i \in B} e(W(m), \check{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{aligned}$$

4. The signature holder performs the following ZK-PoK for the verifier:

$$\text{ZK-PoK} \left\{ (\{\tau_i\}_{i \in B}) : \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_3 \right\}$$

5.4 Threshold Attribute-Based C-Signatures

We define threshold attribute-based C-signatures (t-ABCS) to accommodate for construction of an anonymous credential scheme where users' pseudonyms are in the form of committed values. Hence, we extend t-ABS⁺ schemes to schemes supporting efficient protocols for signing and verifying signatures on committed values (i.e., pseudonyms) in the following. Since the signatures are on committed values,

we define our schemes on messages of the form (m, r) such that r is a random value that is used to compute a commitment on m . The signer will only see the commitment on m during the signing and (m, r) remains a private input to the user. This technique enables the user to prove their credential on a different commitment on m using a new random value r' and hence be able to perform unlinkable transactions. Details of the scheme appears in the following.

5.4.1 Definition

A t-ABCS scheme consists of a t-ABS⁺ scheme defined on messages in the form $\tilde{m} = (m, r)$, a commitment scheme, and the following additional protocols:

iCSign: An *interactive signing protocol* for signing a committed value on the *signer* side and obtaining a signature on a committed value on the *user* side. The public inputs are the t-ABS master public key mpk , the commitment public key cpk , and a commitment M . User's private inputs are the message to be signed (m, r) containing a random value r such that $M = \text{Commit}(cpk, m, r)$. Signer's private input is its signing key ssk . At the end of the protocol execution the user will output signer's signature σ on m .

iHVerify: An *interactive verification protocol* for proving knowledge of a converted signature on the *prover* side and verifying a converted signature on the *verifier* side. The public inputs are the t-ABS master public key mpk and verifier's verification attribute set B . Prover's private input is the tuple $(m, r, \tilde{\sigma})$ such that $\tilde{\sigma}$ is a converted signature on (m, r) . The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B .

iCVerify: An *interactive verification protocol* for proving knowledge of a converted signature on a committed value on the *prover* side and verifying a converted signature on a committed value on the *verifier* side. The public inputs are the t-ABS master public key mpk , the commitment public key cpk , a commitment M' , and verifier's verification attribute set B . Prover's private input is the tuple $(m, r, r', \tilde{\sigma})$ such that $M' = \text{Commit}(cpk, m, r')$ and $\tilde{\sigma}$ is a converted signature on (m, r) . The verifier has no private input. At the end of the protocol execution the verifier will output a binary value reflecting prover's converted signature validity against B .

Correctness. Besides correctness of the underlying t-ABS⁺ scheme, we require that the above protocols should satisfy the correctness property. For iCSign it means that if the user and the signer follow the protocol, then the user should output a valid signature. In other words, if σ is the output the user gets from running the protocol with a signer with attribute set A , then $\text{Convert}(mpk, (m, r), \sigma, B)$ should pass the verification test for any B if $|A \cap B| \geq d$. Correctness for iHVerify and iCVerify means that if the prover and the verifier follow the protocol, then the verifier finds out the validity or invalidity of the prover's converted signature correctly. This means that the verifier's output is the same as $\text{CvtVerify}(mpk, (m, r), \tilde{\sigma}, B)$.

Security. Besides security of the underlying t-ABS⁺ scheme, we require the following security properties from the additional protocols. We require that the iCSign protocol is secure in the following

senses:

Security for the user: Users with different private inputs m should remain indistinguishable for the signer, even if the signer acts maliciously. In particular, we require that for any mpk and cpk , there is no malicious signer that can distinguish if it is interacting with a user with private input containing m_0 or with a user with private input containing m_1 .

Security for the signer: The protocol should reveal (almost) no information other than a single signature on a known committed value to a user, even though the user acts maliciously. In particular, we require that for any mpk and cpk , and for any (possibly malicious) user, there exists a *simulator*, that with only a *one-time* access to the signing oracle, can simulate a signer's interaction with the user.

We require that the $iHVerify$ is (i) a zero knowledge protocol (*security for the prover*) and (ii) a proof of knowledge of a triplet $(m, r, \tilde{\sigma})$ such that $CvtVerify(mpk, (m, r), \tilde{\sigma}, B) = 1$ (*security for the verifier*). We require that the $iCVerify$ is (i) a zero knowledge protocol (*security for the prover*) and (ii) a proof of knowledge of a quadruple $(m, r, r', \tilde{\sigma})$ such that $M' = Commit(cpk, m, r')$ and $CvtVerify(mpk, (m, r), \tilde{\sigma}, B) = 1$ (*security for the verifier*).

A t-ABCS scheme is said to be correct and secure if the underlying t-ABS⁺ scheme defined on messages in the form $\tilde{m} = (m, r)$, is correct and C-EUF-CMAA-secure, the commitment scheme is correct and secure, and the associated $iCSign$, $iHVerify$ and $iCVerify$ protocols are correct and secure for the user, signer, prover and verifier in the above senses.

5.4.2 Construction

Our t-ABS⁺ scheme can be modified to be defined on messages in the form of $\tilde{m} = (m, r)$ as follows: (i) in the **Setup** algorithm, add a random element g_3 from \mathbb{G}_1 to the master public key and set $mpk = (g, g_1, g_2, g_3, t_1, t_2, \dots, t_{n+1}, h)$, and (ii) in the **Sign**, **Verify**, **Convert**, and **CvtVerify** algorithms, replace g_1^m with $g_1^m g_3^r$ everywhere. We will not use the $iVerify$ protocol. The scheme is as follows.

Setup(1^k): Pick y randomly from \mathbb{Z}_p and set $g_1 = g^y$. Pick random elements $g_2, g_3, h, t_1, t_2, \dots, t_{n+1}$ from \mathbb{G}_1 . Define

$$T(x) \triangleq g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)}$$

and output the following:

$$msk = y \quad \text{and} \quad mpk = (g, g_1, g_2, g_3, t_1, t_2, \dots, t_{n+1}, h)$$

KeyGen(msk, A): Choose a random $d-1$ degree polynomial $q(x)$ such that $q(0) = y$, choose random elements r_i in \mathbb{Z}_p for $i \in A$, and output

$$ssk = \left\langle \left\{ g_2^{q(i)} T(i)^{r_i}, \quad g^{r_i} \right\}_{i \in A} \right\rangle$$

Sign($ssk, (m, r)$): Parse the signing key as $ssk = \langle \{ssk_{1i}, ssk_{2i}\}_{i \in A} \rangle$, pick random elements s_i in \mathbb{Z}_p for all $i \in A$, and output

$$\sigma = \left\langle A, \quad \left\{ ssk_{1i} (g_1^m g_3^r \cdot h)^{s_i}, \quad ssk_{2i}, \quad g^{s_i} \right\}_{i \in A} \right\rangle$$

Verify($mpk, (m, r), \tilde{\sigma}, B$): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$ and check if the following equation holds:

$$\prod_{i \in S} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^m g_3^r \cdot h, \sigma_{3i})} \right)^{\Delta_{i,S}(0)} = e(g_2, g_1)$$

Convert($mpk, (m, r), \sigma, B$): Parse the signature as $\sigma = \langle A, \{\sigma_{1i}, \sigma_{2i}, \sigma_{3i}\}_{i \in A} \rangle$. Select an $S \subseteq A \cap B$ such that $|S| = d$. Calculate the converted signature components as follows:

$$\begin{aligned} \text{for all } i \in S: \quad & \tilde{\sigma}_{1i} \leftarrow \sigma_{1i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow \sigma_{2i}^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow \sigma_{3i}^{1/\Delta_{i,B \setminus S}(0)} \\ \text{for all } i \in B \setminus S: \quad & \tilde{\sigma}_{1i} \leftarrow (T(i)g_1^m g_3^r h)^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{2i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} & \tilde{\sigma}_{3i} \leftarrow g^{1/\Delta_{i,B \setminus S}(0)} \end{aligned}$$

and output $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$.

CvtVerify($mpk, (m, r), \tilde{\sigma}, B$): Parse the converted signature as $\tilde{\sigma} = \langle \{\tilde{\sigma}_{1i}, \tilde{\sigma}_{2i}, \tilde{\sigma}_{3i}\}_{i \in B} \rangle$ and check if the following equation holds:

$$\prod_{i \in B} \left(\frac{e(\tilde{\sigma}_{1i}, g)}{e(T(i), \tilde{\sigma}_{2i}) \cdot e(g_1^m g_3^r \cdot h, \tilde{\sigma}_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1) \quad (5.3)$$

For the commitment scheme, consider the scheme with $cpk = (g_1, g_3)$ and $\text{Commit}(cpk, m, r) = g_1^m g_3^r$. This scheme is (unconditionally) hiding and (computationally) binding if the discrete logarithm problem is hard. For this commitment scheme and the above t-ABS scheme, we introduce the following additional protocols to construct a t-ABCS scheme altogether. The iCSign protocol is as follows:

1. the user gives a ZK-PoK of (m, r) such that $M = g_1^m \cdot g_3^r$.
2. the signer picks random elements s_i in \mathbb{Z}_p for all $i \in A$, calculates the signature as below, and sends it to the user.

$$\sigma = \langle A, \{ \text{ssk}_{1i}(Mh)^{s_i}, \text{ssk}_{2i}, g^{s_i} \}_{i \in A} \rangle$$

The iHVerify and iCVerify protocols are as follows:

1. The signature holder randomizes the converted signature by first choosing random elements s'_i and r'_i for $i \in B$ and then calculating the following. Note that the resulting randomized converted signature is a valid converted signature itself.

$$\check{\sigma}_{1i} = \tilde{\sigma}_{1i} \cdot T(i)^{r'_i} (g_1^m g_3^r \cdot h)^{s'_i} \quad \text{and} \quad \check{\sigma}_{2i} = \tilde{\sigma}_{2i} \cdot g^{r'_i} \quad \text{and} \quad \check{\sigma}_{3i} = \tilde{\sigma}_{3i} \cdot g^{s'_i}$$

2. The signature holder chooses random values τ_i for all $i \in B$ and sets $\hat{\sigma}_{1i} \leftarrow \check{\sigma}_{1i}^{1/\tau_i}$ and sends $\langle \{\hat{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$ to the verifier.
3. Both the signature holder and the verifier calculate the following for all $i \in B$:

$$\begin{aligned} u_0 &\leftarrow e(g_2, g_1) & u_{1i} &\leftarrow e(\hat{\sigma}_{1i}, g)^{\Delta_{i,B}(0)} & u_2 &\leftarrow \prod_{i \in B} e(T(i), \check{\sigma}_{2i})^{\Delta_{i,B}(0)} \\ u_{31} &\leftarrow \prod_{i \in B} e(g_1, \check{\sigma}_{3i})^{\Delta_{i,B}(0)} & u_{32} &\leftarrow \prod_{i \in B} e(g_3, \check{\sigma}_{3i})^{\Delta_{i,B}(0)} & u_{33} &\leftarrow \prod_{i \in B} e(h, \check{\sigma}_{3i})^{\Delta_{i,B}(0)} \end{aligned}$$

4. The signature holder performs the following ZK-PoK for the verifier based on the protocol:

$$\begin{aligned} \text{for iHVerify: } & \text{ZK-PoK} \left\{ (m, r, \{\tau_i\}_{i \in B}) : \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33} \right\} \\ \text{for iCVerify: } & \text{ZK-PoK} \left\{ (m, r, r', \{\tau_i\}_{i \in B}) : M' = g_1^m g_3^{r'} \wedge \prod_{i \in B} u_{1i}^{\tau_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33} \right\} \end{aligned}$$

Correctness. The protocol iCSign is correct for obvious reasons. Correctness of the iHVerify and iCVerify protocols is also easy to see, considering the fact that the equation $\prod_{i \in B} u_{1i}^{r_i} = u_0 u_2 u_{31}^m u_{32}^r u_{33}$ is just a rearrangement of Equation 5.3.

Security. On unforgeability of our scheme, we prove Theorem 5.4.1 that comes in the following. As discussed under unforgeability of our t-ABS scheme, the theorem implies that the scheme is also EUF-CMAA-secure with a loose reduction or is EUF-CMAA-secure with an efficient reduction in the random oracle model.

Theorem 5.4.1 *The above t-ABCS scheme, i.e. our t-ABS⁺ scheme defined on messages in the form $\tilde{m} = (m, r)$, is C-SUF-CMAA-secure if the CDH problem is hard.*

Proof. Suppose that there exists an adversary A who forges a converted signature for the t-ABS scheme on a selected message $\tilde{\mu} = (\mu, \rho)$ and verification attribute set β in a chosen message and attribute set attack with probability ϵ . Then, one of the following is true:

- either* with probability at least $\epsilon/2$, A forges on a message (μ, ρ) such that for all queried messages (m, r) during the attack $g_1^m g_3^r \neq g_1^\mu g_3^\rho$,
- or* with probability at least $\epsilon/2$, A forges on a message (μ, ρ) such that for some queried message (m, r) during the attack $g_1^m g_3^r = g_1^\mu g_3^\rho$.

Let's call the above types of forgery as type 1 and type 2 forgery, respectively. We show that both types of forgery enables us to solve the CDH problem.

Dealing with A Type 1 Forger: Suppose we are given a type 1 forger A. We show how construct an algorithm B to solve the CDH problem. B is given (g, g^a, g^b) as input and calculates g^{ab} by running A as a subroutine and simulating the attack environment for it. When B initiates A, A outputs its target verification attribute set β and target message $\tilde{\mu} = (\mu, \rho)$. B sets $g_1 = g^a$ and $g_2 = g^b$. Then it chooses a random n degree polynomial $f(x)$ and another random n degree polynomial $u(x)$ such that $u(x) = -x^n$ if and only if $x \in \beta$. B then sets $t_i = g_2^{u(i)} g^{f(i)}$ for $i = 1, \dots, n+1$. Note that we implicitly have $T(x) = g_2^{x^n + u(x)} g^{f(x)}$ since

$$\begin{aligned} T(x) &= g_2^{x^n} \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(x)} = g_2^{x^n} \prod_{i=1}^{n+1} (g_2^{u(i)} g^{f(i)})^{\Delta_{i,N}(x)} \\ &= g_2^{x^n} g_2^{\sum_{i=1}^{n+1} u(i) \Delta_{i,N}(x)} g^{\sum_{i=1}^{n+1} f(i) \Delta_{i,N}(x)} \\ &= g_2^{x^n + u(x)} g^{f(x)} \end{aligned}$$

B then chooses z and γ randomly in \mathbb{Z}_p and sets $g_3 = g_1^z$ and $h = g_1^{-\mu} g_3^{-\rho} \cdot g^\gamma$. It then gives $pk = (g, g_1, g_2, g_3, t_1, t_2, \dots, t_{n+1}, h)$ to A who will start its secret key and signature queries. B responds to these queries as follows.

On a secret key query for a user with attribute set α , where $|\beta \cap \alpha| < d$, B defines $\Gamma = \beta \cap \alpha$ and lets Γ' be any set such that $\Gamma \subseteq \Gamma' \subseteq \alpha$ and $|\Gamma'| = d - 1$. B then sets $S = \Gamma' \cup \{0\}$ and calculates the following:

- for $i \in \Gamma'$ chooses r_i and λ_i randomly in \mathbb{Z}_p and sets

$$ssk_i = \left\langle g_2^{\lambda_i} T(i)^{r_i}, g^{r_i} \right\rangle.$$

- for $i \in \alpha \setminus \Gamma'$ chooses r'_i randomly in \mathbb{Z}_p and sets

$$ssk_{1i} = \left(g_1^{\frac{-f(i)}{i^n+u(i)}} \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{r'_i} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)}$$

$$ssk_{2i} = \left(g_1^{\frac{-1}{i^n+u(i)}} g^{r'_i} \right)^{\Delta_{0,S}(i)}.$$

The simulation is obviously correct for $i \in \Gamma'$. To see why the simulation for $i \in \alpha \setminus \Gamma'$ is also correct, let $q(x)$ be an $n-1$ degree polynomial such that $q(i) = \lambda_i$ for $i \in \Gamma'$ and $q(0) = a$. This polynomial is implicitly chosen randomly by random choices of elements λ_i by \mathbf{B} . Also note that $i^n + u(i) \neq 0$ for all $i \notin \beta$, which covers all elements $i \in \alpha \setminus \Gamma'$. Furthermore, let's implicitly set $r_i = \left(r'_i - \frac{a}{i^n+u(i)} \right) \Delta_{0,S}(i)$ and we will have:

$$\begin{aligned} ssk_{1i} &= \left(g_1^{\frac{-f(i)}{i^n+u(i)}} \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{r'_i} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)} \\ &= \left(g^{\frac{-af(i)}{i^n+u(i)}} \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{r'_i} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)} \\ &= \left(g_2^a \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{\frac{-a}{i^n+u(i)}} \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{r'_i} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)} \\ &= \left(g_2^a \left(g_2^{i^n+u(i)} g^{f(i)} \right)^{r'_i - \frac{a}{i^n+u(i)}} \right)^{\Delta_{0,S}(i)} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)} \\ &= g_2^{a\Delta_{0,S}(i)} T(i)^{r_i} \prod_{j \in \Gamma'} g_2^{\lambda_j \Delta_{j,S}(i)} \\ &= g_2^{q(0)\Delta_{0,S}(i) + \sum_{j \in \Gamma'} q(j)\Delta_{j,S}(i)} T(i)^{r_i} \\ &= g_2^{q(i)} T(i)^{r_i} \end{aligned}$$

and

$$ssk_{2i} = \left(g_1^{\frac{-1}{i^n+u(i)}} g^{r'_i} \right)^{\Delta_{0,S}(i)} = \left(g^{r'_i - \frac{a}{i^n+u(i)}} \right)^{\Delta_{0,S}(i)} = g^{r_i}.$$

Also note that random choices of elements r'_i implies random elements r_i . Thus, ssk is simulated correctly by \mathbf{B} .

On a signature query by a user with attribute set α on a message (m, r) , where $(m, r) \neq (\mu, \rho)$ or $|\beta \cap \alpha| < d$, \mathbf{B} simulates the signature as follows:

- If $|\beta \cap \alpha| < d$ then \mathbf{B} calculates ssk as above and then calculates the signature as follows for random choices of s_i

$$\sigma = \left\langle \{ssk_{1i} (g_1^m g_3^r \cdot h)^{s_i}, ssk_{2i}, g^{s_i}\}_{i \in \alpha} \right\rangle.$$

- Otherwise \mathbf{B} chooses a random $d-1$ degree polynomial $q'(x)$ such that $q'(0) = \frac{\gamma}{m-\mu+z(r-\rho)}$ and random s'_i and r_i in \mathbb{Z}_p for all $i \in \alpha$ and sets

$$\sigma = \left\langle \left\{ g_2^{q'(i)} T(i)^{r_i} (g_1^m g_3^r \cdot h)^{s'_i}, g^{r_i}, g_2^{\frac{-1}{m-\mu+z(r-\rho)}} g^{s'_i} \right\}_{i \in \alpha} \right\rangle.$$

In the former case, B's simulation is correct for obvious reasons. We show that the simulation is correct for the latter case as well. Note that since A is a type 1 forger, we have $g_1^m g_3^r \neq g_1^\mu g_3^\rho$, which implies that $m + zr \neq \mu + z\rho$ (since $g_3 = g_1^z$). This ensures that $m - \mu + z(r - \rho) \neq 0$ and the above assignments are well defined. Now, let's (implicitly) define $s_i = s'_i - \frac{b}{m - \mu + z(r - \rho)}$ and $q(i) = q'(i) + a + \frac{\gamma}{m - \mu + z(r - \rho)}$. We have:

$$\begin{aligned}
\sigma_{1i} &= g_2^{q'(i)} T(i)^{r_i} (g_1^m g_3^r \cdot h)^{s'_i} \\
&= g_2^{q'(i)} T(i)^{r_i} (g_1^m g_3^r \cdot h)^{\frac{b}{m - \mu + z(r - \rho)}} (g_1^m g_3^r \cdot h)^{s'_i - \frac{b}{m - \mu + z(r - \rho)}} \\
&= g_2^{q'(i)} T(i)^{r_i} (g_1^m g_3^r \cdot g_1^{-\mu} g_3^{-\rho} g^\gamma)^{\frac{b}{m - \mu + z(r - \rho)}} (g_1^m g_3^r \cdot h)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} \left(g_1^{m - \mu + z(r - \rho)} \cdot g^\gamma \right)^{\frac{b}{m - \mu + z(r - \rho)}} (g_1^m g_3^r \cdot h)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} g_1^b \cdot g^{\frac{b\gamma}{m - \mu + z(r - \rho)}} (g_1^m g_3^r \cdot h)^{s_i} \\
&= g_2^{q'(i)} T(i)^{r_i} g_2^a \cdot g_2^{\frac{\gamma}{m - \mu + z(r - \rho)}} (g_1^m g_3^r \cdot h)^{s_i} \\
&= g_2^{q(i)} T(i)^{r_i} (g_1^m g_3^r \cdot h)^{s_i}
\end{aligned}$$

and

$$\sigma_3 = g_2^{\frac{-1}{m - \mu + z(r - \rho)}} g^{s'_i} = g^{s'_i - \frac{b}{m - \mu + z(r - \rho)}} = g^{s_i}$$

Note that random choices of s'_i and $q'(x)$ imply randomness of s_i and $q(x)$. We also have $q(0) = q'(0) + a + \frac{\gamma}{m - \mu + z(r - \rho)} = a$. Thus B's simulations in the latter case are correct as well.

Finally, the adversary A outputs a forgery $\tilde{\sigma}$ for the verification attribute set β and message (μ, ρ) . Since $\tilde{\sigma}$ is a valid converted signature, we have:

$$\prod_{i \in B} \left(\frac{e(\sigma_{1i}, g)}{e(T(i), \sigma_{2i}) \cdot e(g_1^\mu g_3^\rho \cdot h, \sigma_{3i})} \right)^{\Delta_{i,B}(0)} = e(g_2, g_1).$$

Now, note that $g_1^\mu g_3^\rho \cdot h = g^\gamma$ and also for any $i \in \beta : T(i) = g^{f(i)}$. Thus the above equation can be rewritten as follows:

$$\prod_{i \in B} \left(\frac{e(\sigma_{1i}, g)}{e(\sigma_{2i}^{f(i)}, g) \cdot e(\sigma_{3i}^\gamma, g)} \right)^{\Delta_{i,B}(0)} = e(g^{ab}, g).$$

Hence B is able to calculate g^{ab} and solve the CDH problem instance as follows:

$$\prod_{i \in B} \left(\frac{\sigma_{1i}}{\sigma_{2i}^{f(i)} \cdot \sigma_{3i}^\gamma} \right)^{\Delta_{i,B}(0)} = g^{ab}.$$

Dealing with A Type 2 Forger: Suppose we are given a type 2 forger A. We show how construct an algorithm C to solve the discrete logarithm and hence the CDH problem. C is given (g, g^z) as input and calculates z by running A as a subroutine and simulating the attack environment for it. When C initiates A, A outputs its target attribute set β and target message $\tilde{\mu} = (\mu, \rho)$. C follows the Setup algorithm of the t-ABS scheme to generate msk and mpk except for g_3 which it sets as $g_3 = g^z$. Thus, C picks y randomly from \mathbb{Z}_p and sets $g_1 = g^y$ and also chooses random elements $g_2, h, t_1, t_2, \dots, t_{n+1}$ from \mathbb{G}_1 . $T(x)$ is defined in the same way. C then keeps $msk = y$ private and gives $mpk = (g, g_1, g_2, g_3, t_1, t_2, \dots, t_{n+1}, h)$ to the forger. Since C knows msk , it is able to answer A's secret key and signature queries by simply running the KeyGen and Sign algorithms of the t-ABS scheme.

Finally, the adversary A outputs a forgery $\tilde{\sigma}$ for the verification attribute set β and message (μ, ρ) . Since A is a type 2 forger, for some queried message (m, r) during the attack we have $g_1^m g_3^r = g_1^\mu g_3^\rho$, hence $ym + zr = y\mu + z\rho$. Thus C can calculate the discrete logarithm as $z = y \frac{\mu - m}{r - \rho}$. Note that $r - \rho \neq 0$, since otherwise $r = \rho$ together with $ym + zr = y\mu + zr$ would imply that $m = \mu$, hence $(\mu, \rho) = (m, r)$ and (μ, ρ) would not count as a forgery. ■

On security of our additional protocols, we prove the following theorem.

Theorem 5.4.2 *The above protocols iCSign, iHVerify, and iCVerify are secure for the user, signer, prover and verifier.*

Proof. For the iCSign protocol, the only information the signer receives about m during the protocol is the commitment M , since the first step of the protocol is zero knowledge. Now since M perfectly hides m , the protocol satisfies the “security for the user” property. Technically, a distinguisher for the protocol implies either a distinguisher for the zero knowledge protocol in the first step or a distinguisher for the commitment scheme, which in turn contradicts witness indistinguishability of the zero knowledge protocol or the hiding property of the commitment scheme, respectively.

To prove the second property for iCSign protocol, we introduce the following simulator. The simulator first runs the knowledge extractor for the proof of knowledge in the first step of the protocol, which gives a pair (m, r) such that $M = g_1^m \cdot g_3^r$. Then the simulator queries a signature on (m, r) from the signing oracle and sends the acquired signature to the user. This shows that the “security for the signer” property is also satisfied.

To prove the zero knowledge property for the iHVerify and iCVerify protocols, note that the tuple that the prover sends to the verifier before performing the ZK-PoK, i.e., $\langle \{\hat{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$, consists of three random values independent of the actual converted signature. The reason is that these components are randomized by independent random elements τ_i , r'_i , and s'_i , respectively. Thus the following simulator is able to simulate the view of the verifier. First, the simulator picks random elements $\hat{\sigma}_{1i}$, $\check{\sigma}_{2i}$, and $\check{\sigma}_{3i}$ and sends them to the verifier. Then, it runs the simulator for the zero knowledge protocol which simulates the rest of the verifier’s view. Thus both the iHVerify and iCVerify protocols satisfy the “security for the prover” property.

To prove the proof of knowledge property for the iHVerify and iCVerify protocols, we must introduce an extractor for each that extracts a message converted-signature pair. Consider the following algorithm. First, the extractor receives the values $\langle \{\hat{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$ from the prover. Then it runs the extractor for the proof of knowledge in the last step of the protocol to extract a tuple $(m, r, \{\tau_i\}_{i \in B})$ if the protocol is iHVerify or $(m, r, r', \{\tau_i\}_{i \in B})$ if the protocol is iCVerify. Now, by letting $\check{\sigma}_{1i} = \hat{\sigma}_{1i}$ the extractor gets a randomized converted signature $\check{\sigma} = \langle \{\check{\sigma}_{1i}, \check{\sigma}_{2i}, \check{\sigma}_{3i}\}_{i \in B} \rangle$, which is itself a valid converted signature. Thus, the “security for the verifier” property is satisfied for both protocols. ■

Furthermore, hardness of the CDH problem implies hardness of the discrete logarithm problem, which, in turn, is sufficient for the commitment scheme to be secure. Thus our t-ABCS scheme is secure as per our definition if CDH is hard.

5.5 Threshold Attribute-Based Anonymous Credential Systems

A *credential system* involves users and organizations. Organizations issue *credentials* to users based on their policy, which might require users to prove possession of credentials from other organizations. Users are considered holders of the credential they are issued with and might wish to prove possession of their credentials to organizations to get a service or be granted new credentials. An *anonymous credential system* is a credential system in which users are known to organizations only by their *pseudonyms*. Such pseudonyms should be well-defined, i.e., each pseudonyms must belong to only one user. This is guaranteed by requiring all users to reveal their identity and pseudonyms to a trusted authority and get a *zeroth credential* for each of their pseudonyms. Possession of this zeroth credential is proved to an organization at the time of forming the pseudonym with the organization.

A basic anonymous credential system must support a minimal of three basic protocols, respectively for *forming* pseudonyms, *granting* credentials, and *verifying* credentials. All these protocols are between a user and an organization. Users first form the pseudonyms with any organization that they might wish to issue a credential to them. Then they might be granted a credential on their formed pseudonym. Possession of such a credential can be verified by an organization at a later time. The user proves possession of her credential to obtain a service from the verifier. If this service includes issuing a credential to the user, then the user must have formed a pseudonym with the organization previously and prove possession of her credential on the formed pseudonym. Otherwise, there is no need to involve a pseudonym. The user simply proves possession of her credential in this case. Thus, an optional property is that the system supports two types of verification protocols: one for verifying possession of a credential on a formed pseudonym, and another for simply verifying possession of a credential, which implements more efficiently than the former.

An attribute-based anonymous credential system is one with attribute-based organizations. Each organization is given a signing key based on its attributes by a trusted authority. A threshold attribute-based anonymous credential system is an attribute-based anonymous credential system that supports threshold attribute-based verification. We assume that the “credential types” in the system are fixed and each organization is given extra attributes for the credential types they can issue. Hence, a credential only consists of the identity of the credential holder and the signature of the organization on it, using the appropriate signing attribute for the credential type. Hence, a driving license would be in the form of a signature on the identity of a user with the attribute “authorized driving license issuer” among the issuer organization attribute set.

5.5.1 Security Framework

To formalize a security definition for a threshold attribute-based anonymous credential system, we use the simulation paradigm. Two models of a credential system, an ideal model and a real model, are presented. In both models, the system contains users and organizations as entities. Each model contains an *active* adversary in the sense that not only the adversary sees all the messages sent to the *corrupted* entities (users and organizations), but also upon corruption, the adversary gets hold of all

the secret information and future actions of the corrupted entity. However, only *static* adversaries are considered. That is, the adversary only gets to pick the entities it wishes to corrupt in the beginning of the system execution and is not allowed to dynamically corrupt new entities during the system run. Of course, this is a restricted model for adversaries, since in the general sense the adversary can corrupt entities in the system dynamically.

In the real model, the entities communicate with each other directly. However, in the ideal model, they communicate via a *trusted party*, which locally performs all calculations needed to carry out a transaction and just reports the corresponding outputs to the entities involved. The adversary is assumed to be unable to eavesdrop the communications. To be able to compare the two systems, a scheduler entity is introduced to the system, called the *environment*. The environment schedules transactions in the system and gathers output information. In particular, it tells each entity which protocol to carry out and with whom. At the end of the protocol, the entities involved report back the outcome of the protocol. The environment proceeds in periods and in each period it schedules only one transaction. Concurrent scheduling is not considered.

We do not address a *composable* notion of security, but rather a *stand-alone* notion. Both notions are simulation-based notions of security and hence are used for definition of security for multi-party protocols. They both have the ability to capture security against adversaries that act passively or actively, and statically or dynamically. A stand-alone notion of security guarantees security of the system as long as the system is used on its own and not as a subsystem. A composable notions of security, on the other hand, is able to guarantee security even if the system is used as a subsystem in a larger setting. In the absence of the supersystem, a composable notion of security is reduced to a stand-alone notion. A notable composable framework of security is Canetti's framework of *Universal Composability* [Can01].

A depiction of a typical system with the above setting is shown in Figure 5.1. This framework of security is the same as what Camenisch and Lysyanskaya used for defining security of anonymous credential systems [CL01, Lys02].

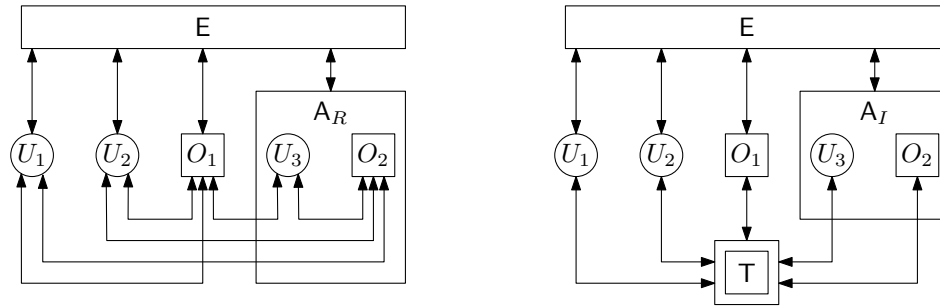


Figure 5.1: The real model (left) vs. the ideal model (right) in a system consisting of the environment E , the users U_1 , U_2 , and U_3 , and the organizations O_1 and O_2 , where the adversary A controls the user U_3 and the organization O_2 . T is the ideal-world trusted party.

A real system is said to be secure if it *emulates* a secure ideal model. Emulation means that for any arbitrary scheduling and any arbitrary real-model adversary, an ideal-model adversary can be found such that the two systems are indistinguishable in the eyes of the environment. An implication of

this property is that whatever the real-model adversary can extract from the real-model system, the ideal-model adversary can extract from the ideal-model system. Now, since the ideal system is defined in a way that its security against ideal-model adversaries is guaranteed, the security of the real-model system follows.

5.5.2 Ideal Model for t-ABACS

The Model. In the ideal, model users and organizations interact through a trusted party T who makes sure the system remains anonymous and secure. There is a public universal set of attributes \mathbb{U} . Each organization O has a set of attributes $A_O \in \mathbb{U}$, which is assumed to be public and fixed during the life of the system. Besides, each organization O , based on its policy at a certain time, might be interested to verify users' credentials against a set of attributes. We denote this set by $B_O \in \mathbb{U}$. B_O is assumed to be made public by the organization and fixed during each period. T maintains three lists: a list of user passwords L_P , a list of user pseudonyms L_N , and a list of issued credentials L_C . L_P contains pairs of user names U and their passwords K_U and is used to authenticate (and hence identify) users. We assume that initially this list contains all user names and passwords in the system. L_N contains triplets of user names U , organization names O , and their pseudonyms N_{UO} , and is initially empty and is filled by T as users form pseudonyms with organizations. L_C contains pairs of user pseudonyms N_{UI} and issuer organization attributes A_I , and is initially empty and is filled by T as organizations grant credentials to users. User pseudonyms are assumed to be chosen randomly and independent of user identities. There are four basic operations as follows.

FormNym $[U \leftrightarrow \mathsf{T} \leftrightarrow O](N_{UO})$: This is a protocol between a user U and an organization O . U wants to form a pseudonym N_{UO} with O . The protocol flow is as follows.

$U \rightarrow (\text{FormNym}, (U, K_U), N_{UO}, O) \rightarrow \mathsf{T}$: U sends her login info (U, K_U) to T along with a request for forming a pseudonym N_{UO} with O .

$\mathsf{T} \rightarrow (\text{FormNym}, N_{UO}) \rightarrow O \mid (U, K_U) \in L_P$: T checks U 's login info and if K_U is U 's password then tells O that a user wants to form a pseudonym N_{UO} with it.

$O \rightarrow (d) \rightarrow \mathsf{T}$: O either accepts or rejects the request and informs T of its decision d .

$\mathsf{T} \rightarrow (d) \rightarrow U$: T informs U of O 's decision d . If O accepts, T also stores the triple (U, O, N_{UO}) in L_N .

GrantCred $[U \leftrightarrow \mathsf{T} \leftrightarrow I](N_{UI})$: This is a protocol between a user U and an organization I . U , who has already formed a pseudonym N_{UI} with I , wants to be granted a credential by I on N_{UI} . I has set of attributes A_I . The protocol flow is as follows.

$U \rightarrow (\text{GrantCred}, (U, K_U), N_{UI}, I) \rightarrow \mathsf{T}$: U sends her login info (U, K_U) to T along with a request for being granted a credential by I on a pseudonym N_{UI} .

$\mathsf{T} \rightarrow (\text{GrantCred}, N_{UI}) \rightarrow I \mid (U, K_U) \in L_P \wedge (U, I, N_{UI}) \in L_N$: T checks U 's login and pseudonym info. If K_U is U 's password and N_{UI} is U 's pseudonym with I then T tells I that the user with pseudonym N_{UI} wishes to be granted a credential.

$I \rightarrow (d) \rightarrow \mathsf{T}$: I either accepts or rejects the request and informs T of its decision d .

$\mathsf{T} \rightarrow (d) \rightarrow U$: T informs U of I 's decision d . If I accepts, T also stores the pair (N_{UI}, A_I) in L_C .

$\mathsf{VerifyCred}[U \leftrightarrow \mathsf{T} \leftrightarrow V](N_{UI})$: This is a protocol between a user U and an organization V . U has a credential issued by I on N_{UI} . A_I is the attribute set of I and B_V is the verification attribute set of V . U wants V to verify that she has been issued a credential by an organization whose attributes A_I has at least d elements in common with V 's verification attribute set B_V . The protocol flow is as follows.

$U \rightarrow (\mathsf{VerifyCred}, (U, K_U), N_{UI}, V) \rightarrow \mathsf{T}$: U sends her login info (U, K_U) to T along with a request for her credential on N_{UI} to be verified by V .

$\mathsf{T} \rightarrow (\mathsf{VerifyCred}) \rightarrow V \mid (U, K_U) \in L_P \wedge [\exists I : (U, I, N_{UI}) \in L_N \wedge (N_{UI}, A_I) \in L_C \wedge |A_I \cap B_V| \geq d]$:
 T checks U 's login and credential info. If K_U is U 's password and there exists an organization I such that N_{UI} is U 's pseudonym with I and a credential has been issued on N_{UI} by an organization with attributes A_I which has at least d elements in common with V 's verification attribute set B_V , then T tells I that the user has a credential from an organization whose attributes has at least d elements in common with V 's verification attribute set.

$V \rightarrow (\mathsf{Ack}) \rightarrow \mathsf{T}$: V acknowledges verification of the credential.

$\mathsf{T} \rightarrow (\mathsf{Ack}) \rightarrow U$: T passes V 's acknowledgment to U .

$\mathsf{VerifyCredOnNym}[U \leftrightarrow \mathsf{T} \leftrightarrow V](N_{UI}, N_{UV})$: This is a protocol between a user U and an organization V . U has a credential issued by I on N_{UI} and has already formed a pseudonym N_{UV} with V . A_I is the attribute set of I and B_V is the verification attribute set of V . U wants V to verify that she has been issued a credential by an organization whose attributes A_I has at least d elements in common with V 's verification attribute set B_V . The protocol flow is as follows.

$U \rightarrow (\mathsf{VerifyCredOnNym}, (U, K_U), N_{UI}, V, N_{UV}) \rightarrow \mathsf{T}$: U sends her login info (U, K_U) to T along with a request for her credential on N_{UI} to be verified by V who knows her by N_{UV} .

$\mathsf{T} \rightarrow (\mathsf{VerifyCredOnNym}, N_{UV}) \rightarrow V \mid (U, K_U) \in L_P \wedge (U, V, N_{UV}) \in L_N \wedge [\exists I : (U, I, N_{UI}) \in L_N \wedge (N_{UI}, A_I) \in L_C \wedge |A_I \cap B_V| \geq d]$: T checks U 's login, pseudonym, and credential info. If K_U is U 's password, N_{UV} is U 's pseudonym with V , and there exists an organization I such that N_{UI} is U 's pseudonym with I and a credential has been issued on N_{UI} by an organization with attributes A_I which has at least d elements in common with V 's verification attribute set B_V , then T tells I that the user with pseudonym N_{UV} has a credential from an organization whose attributes has at least d elements in common with V 's verification attribute set.

$V \rightarrow (\mathsf{Ack}) \rightarrow \mathsf{T}$: V acknowledges verification of the credential.

$\mathsf{T} \rightarrow (\mathsf{Ack}) \rightarrow U$: T passes V 's acknowledgment to U .

Usage Scenario. Let's show how the above ideal model can be used to provide an attribute-based and anonymous solution for a credential system. Consider a user U in the system who already has credentials from organizations O_1 and O_2 on her pseudonyms with them, N_{UO_1} and N_{UO_2} , respectively. There is a third organization O_3 that issues credentials to users that possess two credentials: (i) a credential from an organization with at least d attributes in B_{31} , and (ii) a credential from an organization with at least d attributes in B_{32} . Suppose that O_1 and O_2 satisfy these conditions,

respectively. Now, to get a credential from O_3 , U first runs $\text{FormNym}(N_{UO_3})$ with O_3 to form the pseudonym N_{UO_3} . Then, she runs $\text{VerifyCredOnNym}(N_{UO_1}, N_{UO_3})$ with verification attribute set B_{31} and $\text{VerifyCredOnNym}(N_{UO_2}, N_{UO_3})$ with verification attribute set B_{32} with O_3 . As a result, O_3 is convinced that the owner of pseudonym N_{UO_3} satisfies conditions (i) and (ii), and hence issues a credential on the same pseudonym by running $\text{GrantCred}(N_{UO_3})$ with her. Note that, O_3 does not learn anything more than conditions (i) and (ii) about O_1 and O_2 . Suppose there is a fourth organization O_4 that provides services (that does not include issuing credentials) to users that possess a credential from an organization with at least d attributes in B_4 . U can simply run $\text{VerifyCred}(N_{UO_3})$ with verification attribute set B_4 with O_4 . This convinces O_4 that the user they are dealing with satisfies their policy. Hence, the service is provided to the user.

Properties of the Model: The above model ensures the following security and privacy properties:

Attribute-Based Unforgeability of Credentials: The fact that records can be added to L_C only when a credential is issued together with checking L_C at verification time make it impossible to forge credentials. That is, one cannot prove a credential via VerifyCred or VerifyCredOnNym if it was not issued by an issuer with at least d attributes in common with B .

Privacy of Users: Organizations do not find out anything about a user other than her ownership of some credentials, even if they collude. Thus, the users' privacy is preserved except for the inherent and unavoidable leakage of information as a result of obtaining and/or verification of credentials under the same pseudonym. This property includes the following properties:

Anonymity: The only pieces of information about users the organizations see during the protocols are their pseudonyms. Hence, users are known to organizations only by their pseudonyms and to link a pseudonym to a user, even if organizations and other users collude, they cannot do better than random guessing.

Unlinkability: It is infeasible to link two transactions involving the same user as long as the user uses different pseudonyms in the transactions. Hence, it is also infeasible to link two pseudonyms belonging to the same user.

Issuer Attribute Privacy: Verifiers do not find out anything about the attributes of the credential issuers other than the fact that their attributes satisfy the verification policy.

Non-Transferability of Credentials: Since L_N is checked during both issuing and verifying the credentials, it is infeasible to get credentials on behalf of some other user or transfer one's credential to others.

5.5.3 A Concrete t-ABACS System

Consider a t-ABCS scheme as defined above. We propose the following t-ABACS system based on this scheme. We assume that there is a trusted signing key generator authority outside the system that issues signing keys for organizations based on their attributes. Organizations' attribute sets are all subsets of a universal public attribute set \mathbb{U} . There is also a trusted pseudonym consistency authority that issues zeroth credentials to users and makes sure that pseudonyms remain well-defined.

Init(1^k): During the initiation phase, each user U in the system picks a secret SK_U and each organization O with attributes A_O contacts the system signing key generator to get a signing secret key ssk_O .

FormNym [$U \leftrightarrow O$] (N_{UO}): User U picks a random r_O and forms a commitment to her secret $N_{UO} = \text{Commit}(cpk, SK_U, r_O)$. She then sends N_{UO} to O and proves that she knows the pair (SK_U, r_O) using a ZK-PoK protocol. U and O save N_{UO} as the pseudonym of U with O .

GrantCred [$U \leftrightarrow I$] (N_{UI}): User U and credential issuer I who have already formed a pseudonym $N_{UI} = \text{Commit}(cpk, SK_U, r_I)$, carry out the iCSign protocol on public inputs (mpk, cpk, N_{UI}) , user's private input (SK_U, r_I) , and issuer's private input ssk_I . User stores her output σ .

VerifyCred [$U \leftrightarrow V$] (N_{UI}): User U first calculates $\tilde{\sigma} = \text{Convert}(mpk, (SK_U, r_I), \sigma, B_V)$. Then, user U and verifier V carry out the iHVerify protocol on public inputs (mpk, B_V) and user's private inputs $(SK_U, r_I, \tilde{\sigma})$.

VerifyCredOnNym [$U \leftrightarrow V$] (N_{UI}, N_{UV}): User U first calculates $\tilde{\sigma} = \text{Convert}(mpk, (SK_U, r_I), \sigma, B_V)$. Then, user U and verifier V who have already formed a pseudonym $N_{UV} = \text{Commit}(cpk, SK_U, r_V)$, carry out the iCVerify protocol on public inputs (mpk, cpk, N_{UV}, B_V) and user's private inputs $(SK_U, r_I, r_V, \tilde{\sigma})$.

We assume that pseudonyms are well-defined. That is, each pseudonym belongs to only one user. In practice, this can be guaranteed as follows. All users are required to reveal their identity and pseudonyms to a trusted authority and get a *zeroth credential* for each of their pseudonyms. Possession of this zeroth credential is proved to an organization at the time of forming the pseudonym with the organization.

We prove that the above system is a secure implementation of the t-ABACS system. In particular, we prove the following theorem. In our proof, we provide an ideal-model adversary for any real-model adversary and briefly show how they remain indistinguishable in the eyes of the environment as long as the underlying t-ABCS scheme is secure.

Theorem 5.5.1 *The above concrete t-ABACS system emulates the ideal model for a t-ABACS if the underlying t-ABCS scheme is EUF-CMAA-secure.*

Proof. Let E be the environment and A_R be the adversary in the real model. To prove that our system emulates the ideal model, we must present a ideal-model adversary A_I such that E cannot distinguish if it is talking to the ideal system (of ideal parties and A_I) or to the real system (of real parties and A_R). To do this, we run a copy of A_R and *translate* its communications into the ideal model. As for A_R 's communication with E , no translation is needed. However, we must be able to do the two following translations: (i) translate any message from T , intended for an ideal corrupted party, to a message that A_R understands, i.e., a message intended for a real corrupted party, and (ii) translate any message from A_R (i.e., from a real corrupted party), intended for a real honest party, to a message that T understands. We introduce a translator S that takes care of the above in the following. The combination of S and A_R can be seen as the ideal-model adversary A_I that we propose. A depiction of our simulation of the ideal model using the real adversary A_R and the translator S comes is Figure 5.2.

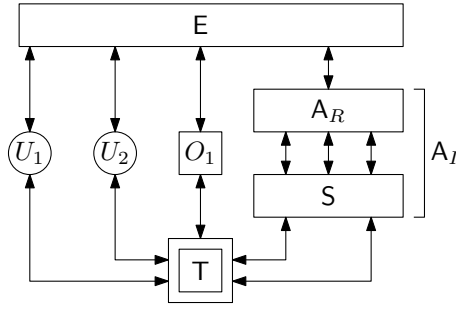


Figure 5.2: Simulation of the ideal model in Figure 5.1 (left) in the proof using the real adversary A_R from the same figure (right) as a black box with its three incoming/outgoing communication channels. S translates these communications into communications with T .

Since all the protocols in our systems are between a user and an organization, S needs to translate messages in two cases: either a protocol between a corrupted user and an honest organization or a protocol between an honest user and a corrupted organization. The translator S first generates secret signing keys for the ideal organizations and then does the following based on the relevant case. In the following we assume that each pseudonym N corresponds with only one pair (SK, r) , since otherwise, the binding property of the commitment scheme is broken. Technically speaking, we are able to use any adversary that breaks the binding property to solve the discrete logarithm problem. Thus, we exclude such adversaries from the following simulation.

Case 1. On the event that a corrupted user initiates a protocol with an honest organization, S does as follows based on the initiated protocol:

- When a corrupted user initiates a real-model **FormNym** protocol with an honest organization O on pseudonym N , S runs the extractor for the proof of knowledge of a committed value protocol and extracts the user's secret key SK and the corresponding r . If SK is a new key, S sets up login information (U, K_U) with T for the corrupted user and stores (U, K_U) under the new entry SK in its records. Otherwise, S looks SK up in its records and retrieves the login information (U, K_U) . Then S runs the ideal-model **FormNym** protocol with T on N . Upon completion of the protocol, S forwards the output to the corrupted user and finishes the real-model **FormNym** protocol execution. Finally, S stores (O, N, r) under SK in its records. Note that this record corresponds to $(U, O, N) \in L_N$ in T 's records.
- When a corrupted user initiates a real-model **GrantCred** protocol with an honest organization I on pseudonym N , S runs the simulator for the **iCSign** protocol. This simulates the interaction for the user. At some point, the simulator queries for a signature on some message (SK, r) . S stops the simulator execution at this time and looks SK up in its records. Then it retrieves the corresponding login information (U, K_U) . Using this login information, S starts an ideal-model **GrantCred** protocol with T on N . When T informs S that the credential is granted, S computes the queried signature σ on (SK, r) using the secret signing key it has generated before for I . Then it answers the signature query of the simulator and resumes execution of the simulator code. Finally, S updates its record of (I, N, r) to (I, N, r, σ) . Note that this record corresponds to $(N, A_I) \in L_C$ in T 's records. Also note that the successful completion of the ideal-model

GrantCred protocol means that $(U, I, N) \in L_N$ and hence (I, N, r) exists in records under SK .

- When a corrupted user initiates a real-model **VerifyCred** protocol with an honest organization V , S runs the extractor for the **iHVerify** protocol and extracts the corrupted user's secret key SK , the corresponding r , and a converted signature $\tilde{\sigma}$ on (SK, r) . S proceeds only if $\tilde{\sigma}$ is a valid converted signature on (SK, r) with respect to B_V . S then looks SK up in its records and retrieves the corresponding (U, K_U) . It also searches for r in its records on SK and checks if it finds a matching record (I, N', r, σ) . If no such matching record is found S checks if any of the corrupted organizations have at least d attributes in common with B_V . If such an organization I is found, S first runs the ideal-model **FormNym** and **GrantCred** protocols on behalf of both U and I with T to form a pseudonym N' and get a credential on it. If neither a matching record nor a suitable corrupted organization is found, S outputs $((SK, r), \tilde{\sigma})$ and halts. Then S runs an ideal-model **VerifyCred** protocol with T on issuer pseudonym N' and relays the outcome of the protocol back to the corrupted user.
- When a corrupted user initiates a real-model **VerifyCredOnNym** protocol with an honest organization V on pseudonym N , S runs the extractor for the **iCVerify** protocol and extracts the corrupted user's secret key SK , the corresponding r and r_V , and a converted signature $\tilde{\sigma}$ on (SK, r) . S proceeds only if $\tilde{\sigma}$ is a valid converted signature on (SK, r) with respect to B_V . S then looks SK up in its records and retrieves the corresponding (U, K_U) . It also searches for r in its records on SK and checks if it finds a matching record (I, N', r, σ) . If no such matching record is found or the converted signature is not equal to $\tilde{\sigma}$, S checks if any of the corrupted organizations have at least d attributes in common with B_V . If such an organization I is found, S first runs the ideal-model **FormNym** and **GrantCred** protocols on behalf of both U and I with T to form a pseudonym N' and get a credential on it. If neither a matching record nor a suitable corrupted organization is found, S outputs $((SK, r), \tilde{\sigma})$ and halts. Then S runs an ideal-model **VerifyCredOnNym** protocol with T on issuer pseudonym N' and verifier pseudonym N and relays the outcome of the protocol back to the corrupted user.

Case 2. On the event that an honest user initiates a protocol with a corrupted organization, T contacts S . S does as follows based on the initiated protocol:

- When an honest user initiates an ideal-model **FormNym** protocol with a corrupted organization O on pseudonym N , T contacts S with a pseudonym establishment request on N . S picks a random secret key SK' for this user. Then S calculates a commitment N' on SK' with a random r' and runs the real-model **FormNym** protocol as a user with O . Upon successful completion of the protocol, S responds to T to complete the ideal **FormNym** protocol and establish the pseudonym N with the honest user. Finally, S stores (O, N, r', N') under secret key SK' in its records. Note that this record corresponds to a record $(U, O, N) \in L_N$ in T 's records for some U that we do not know.
- When an honest user initiates an ideal-model **GrantCred** protocol with a corrupted organization I on pseudonym N , T contacts S with a credential issuance request on N if N is the honest user's pseudonym with I (i.e., $(U, I, N) \in L_N$ for U being the honest user). S first searches its records for a pseudonym N and retrieves the corresponding SK' and (I, N, r', N') . Such a

record exists since $(U, I, N) \in L_N$. Then it runs the **iCSign** protocol as a user with I on input (SK', r') and pseudonym N' . Upon completion of the protocol, if S gets a valid signature σ' on (SK', r') , it notifies T that a credential is granted and completes the ideal-model **GrantCred** protocol with T . Finally, S updates the record (I, N, r', N') to (I, N, r', N', σ') . Note that this record corresponds to $(N, A_I) \in L_C$ in T 's records.

- When an honest user initiates an ideal-model **VerifyCred** protocol with a corrupted organization V on some issuer pseudonym, T contacts S with a credential verification request if the user has a credential on its issuer pseudonym, from an issuer with at least d common attributes with B_V . S runs the simulator for the **iCVerify** protocol. This simulates the interaction with V . Upon completion of the simulation, S acknowledges credential verification to T and finishes the ideal-model **VerifyCred** protocol execution.
- When an honest user initiates an ideal-model **VerifyCredOnNym** protocol with a corrupted organization V on some issuer pseudonym and a verifier pseudonym N , T contacts S with a credential verification request on N if the user has a credential on its issuer pseudonym, from an issuer with at least d common attributes with B_V . S searches its records for a pseudonym N and retrieves the corresponding SK' and (V, N, r', N') . Then S runs the simulator for the **iCVerify** protocol on N' . This simulates the interaction with V . Upon completion of the simulation, S acknowledges credential verification to T and finishes the ideal-model **VerifyCredOnNym** protocol execution.

The simulation only halts when a corrupted user manages to produce a pair $((SK, r), \tilde{\sigma})$ such that $\tilde{\sigma}$ is a valid converted signature on (SK, r) and is not issued through a **GrantCred** protocol by an honest organization. This means that the signature basically is a forgery! Thus if a combination of an environment E and an adversary A_I (itself made of S and A_R) causes the simulation to halt, we can use them to construct an algorithm that forges signatures for the t-ABCS scheme. In the following we show that if the simulation does not halt, then it will provide an indistinguishable simulation.

Let's call each of the above protocols a *simulated* protocol and the above system containing the translator S the *simulated* system. Assume that the environment E schedules a total of ν protocols. We define an i -th *hybrid* system for $i \in \{0, 1, \dots, \nu\}$ as a system in which the first i protocols are simulated protocols and the rest of the protocols are ideal protocols. With such a definition, the ideal system can be seen as the zeroth hybrid and the simulated system as the ν -th hybrid.

Now, if a polynomial time environment E distinguishes between the ideal system and the simulated system, then it should be able to distinguish between $(j-1)$ -th and j -th hybrids for some $j \in \{1, \dots, \nu\}$. Since the only difference between the two hybrids is the j -th protocol, E should be able to distinguish between a simulated j -th protocol and an ideal j -th protocol. Depending on which protocol the j -th protocol is, we have the following cases:

- If the j -th protocol is a **FormNym** protocol between a corrupted user and an honest organization, then E should be able to distinguish between a knowledge extractor and a verifier in the proof of knowledge of a committed value protocol, which is a contradiction.
- If the j -th protocol is a **GrantCred** protocol between a corrupted user and an honest organization, then we use E to construct an algorithm that distinguishes between a simulator and a signer in

the iCSign protocol.

- If the j -th protocol is a VerifyCred or a VerifyCredOnNym protocol between a corrupted user and an honest organization, then E should be able to distinguish between a knowledge extractor and a verifier in the iHVerify or iCVerify protocol, respectively, which is a contradiction.
- The j -th protocol cannot be a FormNym protocol between an honest user and a corrupted organization, since the simulation exactly follows the ideal and real-model protocol procedure.
- If the j -th protocol is a GrantCred protocol between an honest user and a corrupted organization, then we use E to construct an algorithm that distinguishes between interactions with users with different private inputs in the iCSign protocol.
- If the j -th protocol is a VerifyCred or a VerifyCredOnNym protocol between an honest user and a corrupted organization, then we use E to construct an algorithm that distinguishes between a prover and a simulator in the iHVerify or iCVerify protocol, respectively.

Hence we have shown that unless either the t-ABS scheme is forgeable, the commitment scheme is not secure, or either of the iCSign, iHVerify, or iCVerify protocols is not secure in the sense of our definitions, then we succeed in simulating an indistinguishable system for E and this completes the proof. ■

5.6 Concluding Remarks

In this chapter we proposed schemes, protocols, and systems that can be viewed as steps toward designing fully anonymous attribute-based credential systems. We defined and realized threshold attribute-based signature and extended them to threshold attribute-based c-signature schemes that can be used as a building block for realizing threshold attribute-based anonymous credential systems. In these schemes, a credential holder has the option to reveal selective parts of the attributes of the issuer of their credential. Since the attributes of the issuer usually is correlated with the attributes of the credential holder, e.g., in terms of locality, the selective revelation of the issuer attributes provides more privacy for credential holders. Devising solutions for such a selective revelation property using previous techniques, namely proofs of partial knowledge, gives protocols with a cost in the order of the *product* of the sizes of the attribute universe partitions. However, our schemes provide solutions that cost in the order of the *sum* of the sizes of the attribute universe partitions.

A foreseen application for our schemes is in the field of biometric public key cryptography, in which biometric attributes of the users in the system serve as their public key. As one can imagine, since no two reading of the same biometric attribute are exactly equal, in such systems fuzzy matching of attributes needs to be embedded in the cryptographic primitives in use. One such primitive is cryptographic signatures, in which fuzzy verification of the signer attributes needs to be possible. Our proposed threshold attribute-based signatures provide such functionality.

We proved our threshold attribute-based signature and threshold attribute-based c-signature schemes to be selectively unforgeable based on the hardness of the CDH problem and we shown that tight existential unforgeability can be achieved for our threshold attribute-based signature construction

either through assuming random oracles or using Waters’s technique. However, although we expect random oracles to also work for our threshold attribute-based c-signature construction, we do not know of any random oracle free threshold attribute-based c-signature construction that provides existential unforgeability. We leave this as an open problem for further research.

We proposed a construction of threshold attribute-based anonymous credential system based on any secure threshold attribute-based c-signature. We provided a security proof for our threshold attribute-based anonymous credential system in a stand-alone model of security that considers active but static adversaries. A stand-alone security model only guarantees security if the system is used by itself and not as a part of a larger setting. A composable security model, such as the universal composability framework [Can01], on the other hand is able to provide a security guarantee even if the system is used within a larger setting. Hence, our security model can be strengthened in two possible ways: to a model that considers dynamic adversaries and/or to a composable model. A future research direction therefore is to design anonymous credential systems for which security can be proved against dynamic adversaries and/or in a composable model.

Our schemes and hence our threshold attribute-based anonymous credential system only supports threshold verification. In other words, the verifier’s policy of attribute verification is limited to requiring that the issuer of a credential possesses at least a threshold of attributes out of a total number of attributes. More generic attribute verifications may be required in different application scenarios. Among these are support for multi-level access structures and more generally monotone access structures. Designing efficient attribute-based anonymous credential systems supporting general attribute verification policies remains an open problem.

Chapter 6

Conclusions and Open Problems

In this thesis we made contributions to secure and privacy-preserving use of electronic credentials. Our contributions are in three different levels. In the basic scheme level and in the case where the credential statement does not specify the identity of the credential holder, we identified a gap in the literature and made contributions toward filling the gap. In particular, we observed that the literature lacks efficient credential schemes that provide the credential holder with total control over the transfer of their credential. We formalized credential ownership proofs as a solution to this gap and provided several constructions for such schemes that we showed are provably secure. Our proposed schemes include two generic constructions and an efficient scheme that we showed compares favourably with regards to the previous schemes that may be used to provide the same functionality.

In a higher level and in the case where the credential statement does specify the identity of the credential holder, we have identified universal designated-verifier signature schemes as one of the proposals in the literature that has attracted plenty of attention. We proposed a generic construction of such schemes and proved their security. We showed that schemes constructed through our generic constructions compare favourably to the previous schemes, specially since our schemes provide provable non-delegatability and setting independence. Non-delegatability guarantees that the ability to generate signatures cannot be delegated by the possibly malicious entities in the system. Setting independence enables entities with different cryptographic settings to be able to act as credential issuers and verifiers in the system without the need to conform with the setting of the others.

In the system level, we proposed a threshold attribute-based anonymous credential system that ensures signer attribute privacy for the users in the system besides the other privacy guarantees of anonymity and unlinkability. This property ensures that attributes of the issuers of a user's credentials does not leak unnecessary information about the user. To construct such systems, we first formalize and develop a new primitive that we call a threshold attribute-based c-signature and we use it as a building block to realize our threshold attribute-based anonymous credential system. We provide formalization and provably secure schemes for the proposed building block. We prove our system secure given that our building block satisfies certain security properties.

We discuss our contributions in detail in the following.

In Chapter 3, we introduced the concept of secure credential ownership proofs and proposed several constructions. We have shown general constructions based on identity-based encryption and identification schemes. Plenty of secure schemes for each of those primitives have been proposed in the

literature, offering a wide range of options to implement secure credential ownership proofs. Furthermore, the equivalence we have shown between credential ownership proofs and identity-based identifications introduces new scheme designs for the latter (and hence for identity-based signatures through the Fiat-Shamir transform). Our result on the security of the GQ protocol for proving ownership of RSA-FDH credentials enables current credential systems which use such signatures to integrate GQ easily with guaranteed security, while all the previously issued credentials can be still used in the new system.

Table 6.1: Credential ownership proofs in comparison with other credential system solutions

	Sig	COP	Sig+ZK	ACS
Authenticity	✓	✓	✓	✓
Integrity	✓	✓	✓	✓
Non-Repudiability	✓	✓	✓	✓
Non-Impersonability	✗	✓	✓	✓
Zero Knowledge	✗	✗	✓	✓
Non-Transferability	✗	✗	✗	✓
Unlinkability	✗	✗	✗	✓
Anonymity	✗	✗	✗	✓
Issuing Cost	(1,1)	(1,1)	(1,1)	(10,5)
Verification Cost	(1,1)	(5,5)	(10,5)	(200,100)

This chapter can be seen as an attempt to fill a part of the gap between the two ends of the credential systems spectrum, namely standard signatures and pseudonym systems, with the former offering merely the basic security properties and the latter offering ultimate security and privacy protection. For many purposes, the full protection guaranteed in pseudonym systems is not needed. Hence, properly defining and securely designing new schemes which give up parts of such full protection for better efficiency still remains a challenging open problem.

Table 6.1 compares credential ownership proofs with some other credential system solutions in terms of properties they provide and costs. In this table, ‘Sig’, ‘COP’, ‘Sig+ZK’, and ‘ACS’ respectively stand for standard signature, credential ownership proof, standard signature with zero knowledge proof of credential ownership, and anonymous credential system. Costs are in the form (computational,communicational) and are rounded to the nearest “currency” value, i.e. $\{1, 2, 5\} \times 10^i$. As the table shows, both credential ownership proofs and signature schemes with zero knowledge proofs of credential ownership fall in between the two ends of the spectrum, namely standard signatures and anonymous credential systems, both in terms of protection guarantees and cost. Credential ownership proofs are able to provide protection against impersonation with half of the verification computational cost of zero knowledge proofs of credential ownership.

Non-interactive schemes are usually preferred to interactive protocols in cryptography since they reduce the complexity of communication between entities in the system. Our credential ownership proof protocols provide an interactive solution to the corresponding motivating problem. Since our credential ownership proof protocols are public coin protocols they can be converted to non-interactive protocols using the Fiat-Shamir transform. However, the security of the resulting schemes are only

known to be provable in the Random Oracle Model. Designing non-interactive credential ownership proofs with provable security in the standard model remains an open problem.

Table 6.2: A summary of constructions in Chapter 4

Scheme and its signing	U(M)DV Extensions and their designation	(M)DV Variants and their signing
SS: Sign	UDVS: $\text{SoK}\{\tilde{\sigma} \vee sk_v\}$ UMDVS: $\text{SoK}\{\tilde{\sigma} \vee (\bigwedge_j sk_{v_j})\}$	
(H)IBS: $\text{SoK}\{\widetilde{usk_s}\}(m)$	(H)IBUDVS: $\text{SoK}\{\tilde{\sigma}_{(H)IBS} \vee \widetilde{usk_v}\}$ (H)IBUMDVS: $\text{SoK}\{\tilde{\sigma}_{(H)IBS} \vee (\bigwedge_j \widetilde{usk_{v_j}})\}$	(H)IBDVS: $\text{SoK}\{\widetilde{usk_s} \vee \widetilde{usk_v}\}(m)$ (H)IBMDVS: $\text{SoK}\{\widetilde{usk_s} \vee (\bigwedge_j \widetilde{usk_{v_j}})\}(m)$
(H)IBRS: $\text{SoK}\{\bigvee_i \widetilde{usk_{s_i}}\}(m)$	(H)IBUDVRS: $\text{SoK}\{\tilde{\sigma}_{(H)IBRS} \vee \widetilde{usk_v}\}$ (H)IBUMDVRS: $\text{SoK}\{\tilde{\sigma}_{(H)IBRS} \vee (\bigwedge_j \widetilde{usk_{v_j}})\}$	(H)IBDVRS: $\text{SoK}\{\bigvee_i \widetilde{usk_{s_i}} \vee \widetilde{usk_v}\}(m)$ (H)IBMDVRS: $\text{SoK}\{\bigvee_i \widetilde{usk_{s_i}} \vee (\bigwedge_j \widetilde{usk_{v_j}})\}(m)$

In Chapter 4, we have proposed generic constructions of UDVS and IBS schemes for a large class of signatures. Our constructions result in schemes with comparable cost and size to those of their counterparts. Our generic UDVS constructions are provably non-delegatable. Many IBS schemes can be seen as instances of our generic IBS construction. It is possible to use our techniques to construct generic hierarchical identity-based signatures, identity-based universal designated verifier signatures, and identity-based ring signatures. Table 6.2 shows a summary of the generic constructions proposed in this chapter. SS with signing algorithm Sign is the original signature scheme. Each table cell includes the schemes we construct and their corresponding plain and multi-designated-verifier signing or designation procedure. A ‘ \sim ’ denotes a converted quantity using our proposed methods. Note that in (hierarchical) identity-based variants, user secret keys are constructed as signatures on the user identities. Namely, we gave generic constructions of universal (multi) designated-verifier signatures, (hierarchical) identity-based signatures, (hierarchical) identity-based universal (multi) designated-verifier signatures, (hierarchical) identity-based (multi) designated-verifier signatures, (hierarchical) identity-based ring signatures, (hierarchical) identity-based universal (multi) designated-verifier ring signatures, and (hierarchical) identity-based (multi) designated-verifier ring signatures from almost any signature scheme.

Our generic construction of universal designated-verifier signature schemes offer provable non-delegatability and signer—verifier setting independence. Furthermore, security proofs for our schemes are based on either weaker or similar assumptions than those of their counterparts. Compared to previous schemes, our construction almost in all cases provide schemes with faster or equal on-line designation and designated signature verification times. However, our schemes are only provable in the Random Oracle Model. Table 6.3 summarizes how our universal designated-verifier signature

scheme constructions compare on average to the previous schemes. In this table, ‘★’, ‘+’, ‘=’, or ‘−’ respectively denote that our scheme compares ‘always favourably’, ‘favourably’, ‘almost equally’, or ‘unfavourably’ over the previous schemes.

Table 6.3: Summary of how our UDVS constructions compare with previous schemes

Property	Rate
non-delegatability	★
setting independence	★
security assumptions	+
on-line designation cost	+
designated verification cost	=
random oracle freeness	−

Our generic construction of identity-based signature schemes is one that captures all the previously proposed random oracle based schemes in the literature. It provides schemes with fast on-line signature calculation times in many cases. Generic construction of identity-based signature schemes without random oracles remain an open problem.

In Chapter 5, we introduced a new scheme called a threshold attribute-based signature, which allows verification of signatures as originating from a fuzzy signer. We proposed a basic concrete t-ABS scheme and provided a tight security reduction for selective unforgeability based on hardness of the CDH problem and discussed how to achieve existential unforgeability both in the standard and the random oracle models. We have shown that our basic t-ABS scheme admits to an efficient threshold attribute-based C-signature that can be used as a building block to realize privacy-enhanced anonymous credential systems. However, existential unforgeability of our t-ABCS scheme is based on a loose generic reduction. Designing t-ABCS schemes with tight existential unforgeability remains as an open problem.

Our attribute-based signature scheme and hence our attribute-based anonymous credential system only support simple threshold verification policies. A possible future direction is to design schemes that support more complex verification policies. That is, to generalize our schemes to ones in which the verification algorithm gets as input a (complex) policy, rather than a verification attribute set, and the verification goes through if the signer attributes satisfy the verification policy. Systems based on such schemes can accommodate for a larger application spectrum.

Our threshold attribute-based anonymous credential system is proved secure against active but static adversaries in a stand-alone model of security. This guarantees security of our system whenever used independently against adversaries that are able to get hold of users in the setup phase but cannot corrupt users during the life of the system. This leaves the problem of designing systems that are provable against dynamic adversaries and/or in a composable security framework as an open problem.

The threshold attribute-based anonymous credential system we proposed can be used in applications in which fuzzy verification of issuer’s attributes are required. An example of such applications is biometric-based public key systems in which entity public keys are derived from their biometric features. Threshold verification policies appear to be applicable in such cases where a threshold number of biometric features should match for positive identity verification. Our system can be used

as an anonymous credential system in which issuers possess biometric features and are able to issue credentials using their biometric features. Such credentials can be verified by entities possessing a different read of the issuer's biometric features since the fuzzy verification property of our system allows credentials to be verified as valid if sufficient feature overlap exists between the two reads.

In the big picture, we see our contributions as steps toward realizing a comprehensive secure and privacy-preserving suite of schemes, protocols, and system designs for the digital age as individuals in the community become evermore concerned about their privacy.

Bibliography

- [AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In Lars R. Knudsen, editor, *EUROCRYPT '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer, 2002. (Cited on page 24.)
- [ASW00] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. *Selected Areas in Communications, IEEE Journal on*, 18(4):593–610, 2000. (Cited on page 63.)
- [BB04a] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [CC04], pages 223–238. (Cited on pages 99 and 101.)
- [BB04b] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In Cachin and Camenisch [CC04], pages 56–73. (Cited on page 65.)
- [BDS⁺03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003. (Cited on page 51.)
- [BDZ04] Feng Bao, Robert H. Deng, and Jianying Zhou, editors. *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, volume 2947 of *Lecture Notes in Computer Science*. Springer, 2004. (Cited on pages 136, 139 and 140.)
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *CRYPTO '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. (Cited on pages 31, 37, 38, 39 and 85.)
- [BF03] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. (Cited on pages 37 and 38.)
- [BG92] Mihir Bellare and Oded Goldreich. On Defining Proofs of Knowledge. In Brickell [Bri93], pages 390–420. (Cited on page 20.)

- [BLMQ05] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In Roy [Roy05], pages 515–532. (Cited on page 88.)
- [BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In Boyd [Boy01], pages 514–532. (Cited on page 65.)
- [BM07] Feng Bao and Steven Miller, editors. *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2007, Singapore, March 20-22, 2007*. ACM, 2007. (Cited on page 139.)
- [BN06] Mihir Bellare and Gregory Neven. Multi-Signatures in the Plain Public-Key Model and a General Forking Lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security (ACMCCS '06)*, pages 390–399. ACM, 2006. (Cited on page 60.)
- [BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-Based Identification and Signature Schemes. In Cachin and Camenisch [CC04], pages 268–286. (Cited on pages 25, 32, 42, 43, 56, 57, 85 and 88.)
- [BNPS01] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The Power of RSA Inversion Oracles and the Security of Chaum’s RSA-Based Blind Signature Scheme. In Paul F. Syverson, editor, *Financial Cryptography (FC '01)*, volume 2339 of *Lecture Notes in Computer Science*, pages 319–338. Springer, 2001. (Cited on page 16.)
- [Boy01] Colin Boyd, editor. *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*. Springer, 2001. (Cited on pages 131 and 138.)
- [BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In Moti Yung, editor, *CRYPTO '02*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002. (Cited on pages 22, 30, 32, 44 and 61.)
- [BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security (ACMCCS '93)*, pages 62–73. ACM, 1993. (Cited on pages 15, 54, 65 and 99.)
- [BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In *EUROCRYPT '96*, pages 399–416, 1996. (Cited on pages 32, 43, 45 and 66.)
- [Bra00] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000. (Cited on page 30.)
- [Bri93] Ernest F. Brickell, editor. *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992*,

- Proceedings*, volume 740 of *Lecture Notes in Computer Science*. Springer, 1993. (Cited on pages 130 and 137.)
- [BSS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Universal Designated Verifier Signature Proof (or How to Efficiently Prove Knowledge of a Signature). In Roy [Roy05], pages 644–661. (Cited on pages 30, 34 and 36.)
- [BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-Free Secret-Ballot Elections (extended abstract). In *STOC '94*, pages 544–553, 1994. (Cited on page 51.)
- [Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS '01*, pages 136–145, 2001. (Cited on pages 115 and 124.)
- [CC03] Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In Yvo Desmedt, editor, *Public Key Cryptography (PKC '03)*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003. (Cited on pages 28, 57, 88 and 89.)
- [CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004. (Cited on pages 130 and 131.)
- [CC08] Feng Cao and Zhenfu Cao. An Identity Based Universal Designated Verifier Signature Scheme Secure in the Standard Model. *Journal of Systems and Software*, To Appear, 2008. (Cited on page 90.)
- [CD00] Jan Camenisch and Ivan Damgård. Verifiable Encryption, Group Encryption, and Their Applications to Separable Group Signatures and Signature Sharing Schemes. In Tatsuki Okamoto, editor, *ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000. (Cited on page 63.)
- [CDM00] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient Zero-Knowledge Proofs of Knowledge Without Intractability Assumptions. In Imai and Zheng [IZ00], pages 354–372. (Cited on pages 49, 71 and 97.)
- [CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In Yvo Desmedt, editor, *CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994. (Cited on pages 56, 57 and 93.)
- [CDV06] Dario Catalano, Yevgeniy Dodis, and Ivan Visconti. Mercurial Commitments: Minimal Assumptions and Efficient Constructions. In Shai Halevi and Tal Rabin, editors, *TCC '06*, volume 3876 of *Lecture Notes in Computer Science*, pages 120–144. Springer, 2006. (Cited on page 63.)
- [CE87] David Chaum and Jan-Hendrik Evertse. A Secure and Privacy-protecting Protocol for Transmitting Personal Information Between Organizations. In Odlyzko [Odl87], pages 118–167. (Cited on pages 30 and 96.)

- [Cha85] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Commun. ACM*, 28(10):1030–1044, 1985. (Cited on pages 30 and 96.)
- [CL01] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In Birgit Pfitzmann, editor, *EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer, 2001. (Cited on pages 97 and 115.)
- [CL02] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN '02*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer, 2002. (Cited on page 65.)
- [CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In Matthew K. Franklin, editor, *CRYPTO '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004. (Cited on page 65.)
- [CM07] Jean-Sébastien Coron and Alexander May. Deterministic Polynomial-Time Equivalence of Computing the RSA Secret Key and Factoring. *Journal of Cryptology*, 20(1):39–50, 2007. (Cited on page 60.)
- [Cor00] Jean-Sébastien Coron. On the Exact Security of Full Domain Hash. In Mihir Bellare, editor, *CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, 2000. (Cited on page 46.)
- [Cra05] Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005. (Cited on page 139.)
- [CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In Burton S. Kaliski Jr., editor, *CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997. (Cited on pages 56 and 96.)
- [CS00] Ronald Cramer and Victor Shoup. Signature Schemes Based on the Strong RSA Assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000. (Cited on pages 55 and 65.)
- [DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable Cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. (Cited on pages 4 and 31.)
- [Des88] Yvo Desmedt. Subliminal-Free Authentication and Signature (Extended Abstract). In *EUROCRYPT '88*, pages 23–33, 1988. (Cited on pages 4 and 31.)
- [DH76] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on pages 14, 17, 22 and 51.)
- [DK99] Yvo Desmedt and Kaoru Kurosawa. Practical and Proven Zero-Knowledge Constant Round Variants of GQ and Schnorr. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82(1):69–76, 1999. (Cited on page 49.)

- [ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985. (Cited on page 65.)
- [FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-Knowledge Proofs of Identity. *J. Cryptology*, 1(2):77–94, 1988. (Cited on pages 21, 30 and 57.)
- [FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Odlyzko [Odl87], pages 186–194. (Cited on pages 15, 21, 24, 57 and 88.)
- [FS90] Uriel Feige and Adi Shamir. Witness Indistinguishable and Witness Hiding Protocols. In *STOC '90*, pages 416–426. ACM, 1990. (Cited on page 21.)
- [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 '06*, volume 4284 of *Lecture Notes in Computer Science*, pages 178–193. Springer, 2006. (Cited on page 56.)
- [GHR99] Rosario Gennaro, Shai Halevi, and Tal Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In *EUROCRYPT '99*, pages 123–139, 1999. (Cited on pages 55 and 66.)
- [GJM99] Juan A. Garay, Markus Jakobsson, and Philip D. MacKenzie. Abuse-Free Optimistic Contract Signing. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer, 1999. (Cited on page 51.)
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *STOC '82*, pages 365–377. ACM, 1982. (Cited on page 15.)
- [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. (Cited on pages 23, 51, 55, 63 and 66.)
- [GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18(1):186–208, 1989. (Cited on pages 4 and 19.)
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229. ACM, 1987. (Cited on page 15.)
- [GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that Yield Nothing But Their Validity for All Languages in NP Have Zero-Knowledge Proof Systems. *J. ACM*, 38(3):691–729, 1991. (Cited on pages 20 and 97.)
- [GMY06] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening Zero-Knowledge Protocols Using Signatures. *J. Cryptology*, 19(2):169–209, 2006. (Cited on page 63.)

- [Gol01] Oded Goldreich. *Foundations of Cryptography: Volume I – Basic Tools*. Cambridge University Press, 2001. (Cited on page 19.)
- [Gol04] Oded Goldreich. *Foundations of Cryptography: Volume II – Basic Applications*. Cambridge University Press, 2004. (Cited on page 15.)
- [GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A “Paradoxical” Indentity-Based Signature Scheme Resulting from Zero-Knowledge. In Shafi Goldwasser, editor, *CRYPTO ’88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 1988. (Cited on pages 29, 32, 43, 44, 57 and 88.)
- [GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Zheng [Zhe02], pages 548–566. (Cited on pages 56 and 96.)
- [GW04] Shafi Goldwasser and Erez Waisbard. Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In Moni Naor, editor, *TCC ’04*, volume 2951 of *Lecture Notes in Computer Science*, pages 77–100. Springer, 2004. (Cited on pages 55 and 66.)
- [GZ08] Shanjing Guo and Yingpei Zeng. Attribute-based Signature Scheme. *Information Security and Assurance, 2008. ISA 2008. International Conference on*, pages 509–511, April 2008. (Cited on page 96.)
- [HBSO03] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie K. Orman. Hidden Credentials. In Sushil Jajodia, Pierangela Samarati, and Paul F. Syverson, editors, *WPES ’03*, pages 1–8. ACM, 2003. (Cited on page 56.)
- [HCW05] Zhenjie Huang, Kefei Chen, and Yumin Wang. Efficient Identity-Based Signatures and Blind Signatures. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS ’05*, volume 3810 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2005. (Cited on page 88.)
- [Hes02] Florian Hess. Efficient Identity Based Signature Schemes Based on Pairings. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography (SAC ’02)*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002. (Cited on pages 88 and 90.)
- [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 ’06*, volume 4307 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2006. (Cited on pages 53, 55 and 81.)
- [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. (Cited on page 81.)
- [IZ00] Hideki Imai and Yuliang Zheng, editors. *Public Key Cryptography, Third International Workshop on Practice and Theory in Public Key Cryptography, PKC 2000, Melbourne,*

- Victoria, Australia, January 18-20, 2000, *Proceedings*, volume 1751 of *Lecture Notes in Computer Science*. Springer, 2000. (Cited on pages 132 and 137.)
- [JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In *EUROCRYPT '96*, pages 143–154, 1996. (Cited on pages 53, 55 and 72.)
- [KH04] Kaoru Kurosawa and Swee-Huay Heng. From Digital Signature to ID-based Identification/Signature. In Bao et al. [BDZ04], pages 248–261. (Cited on pages 32, 42, 56 and 85.)
- [Kha07a] Dalia Khader. Attribute Based Group Signature with Revocation. Cryptology ePrint Archive, Report 2007/241, 2007. <http://eprint.iacr.org/2007/241>. (Cited on page 96.)
- [Kha07b] Dalia Khader. Attribute Based Group Signatures. Cryptology ePrint Archive, Report 2007/159, 2007. <http://eprint.iacr.org/2007/159>. (Cited on page 96.)
- [Kha08] Dalia Khader. Authenticating with Attributes. Cryptology ePrint Archive, Report 2008/031, 2008. <http://eprint.iacr.org/2008/031>. (Cited on page 96.)
- [LK08] Jin Li and Kwangjo Kim. Attribute-Based Ring Signatures. Cryptology ePrint Archive, Report 2008/394, 2008. <http://eprint.iacr.org/2008/394>. (Cited on page 96.)
- [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 '05*, volume 3783 of *Lecture Notes in Computer Science*, pages 61–71. Springer, 2005. (Cited on page 55.)
- [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 '06*, volume 4116 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2006. (Cited on page 81.)
- [LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym Systems. In Howard M. Heys and Carlisle M. Adams, editors, *Selected Areas in Cryptography (SAC '99)*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1999. (Cited on pages 30 and 96.)
- [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 '04*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2004. (Cited on pages 73 and 81.)
- [LV04b] Fabien Laguillaumie and Damien Vergnaud. Multi-designated Verifiers Signatures. In Javier Lopez, Sihan Qing, and Eiji Okamoto, editors, *ICICS '04*, volume 3269 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2004. (Cited on page 55.)
- [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 '05*, volume

- 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005. (Cited on pages 7, 52, 55 and 68.)
- [Lys02] Anna Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design*. PhD thesis, Massachusetts Institute of Technology, 2002. (Cited on pages 30, 96, 97 and 115.)
- [Mil75] Gary L. Miller. Riemann’s Hypothesis and Tests for Primality. In *STOC ’75*, pages 234–239. ACM, 1975. (Cited on page 60.)
- [MPR08] Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-Based Signatures: Achieving Attribute-Privacy and Collusion-Resistance. Cryptology ePrint Archive, Report 2008/328, 2008. <http://eprint.iacr.org/2008/328>. (Cited on page 96.)
- [MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1997. (Cited on page 81.)
- [NSM05] Ching Yu Ng, Willy Susilo, and Yi Mu. Universal Designated Multi Verifier Signature Schemes. In *ICPADS ’05 (2)*, pages 305–309. IEEE Computer Society, 2005. (Cited on pages 55 and 83.)
- [Odl87] Andrew M. Odlyzko, editor. *Advances in Cryptology - CRYPTO ’86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*. Springer, 1987. (Cited on pages 132 and 134.)
- [Oka92] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Brickell [Bri93], pages 31–53. (Cited on pages 57 and 88.)
- [OO90] Tatsuaki Okamoto and Kazuo Ohta. How to Utilize the Randomness of Zero-Knowledge Proofs. In Alfred Menezes and Scott A. Vanstone, editors, *CRYPTO ’90*, volume 537 of *Lecture Notes in Computer Science*, pages 456–475. Springer, 1990. (Cited on pages 4 and 31.)
- [OO98] Kazuo Ohta and Tatsuaki Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In Hugo Krawczyk, editor, *CRYPTO ’98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer, 1998. (Cited on page 58.)
- [Pat02] Kenneth G Paterson. ID-Based Signatures from Pairings on Elliptic Curves. *Electronics Letters*, 38(18):1025–1026, 2002. (Cited on page 88.)
- [PS00a] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology*, 13(3):361–396, 2000. (Cited on pages 58, 60 and 65.)
- [PS00b] Guillaume Poupard and Jacques Stern. Short Proofs of Knowledge for Factoring. In Imai and Zheng [IZ00], pages 147–166. (Cited on page 60.)
- [PS06] Kenneth G. Paterson and Jacob C. N. Schuldt. Efficient Identity-Based Signatures Secure in the Standard Model. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP ’06*, volume 4058 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2006. (Cited on page 90.)

- [Rab79] Michael O. Rabin. Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979. (Cited on page 65.)
- [Roy05] Bimal K. Roy, editor. *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*. Springer, 2005. (Cited on pages 131 and 132.)
- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978. (Cited on pages 6, 16, 45, 51, 60, 65 and 142.)
- [RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In Boyd [Boy01], pages 552–565. (Cited on page 55.)
- [SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal Designated-Verifier Signatures. In Chi-Sung Lai, editor, *ASIACRYPT '03*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2003. (Cited on pages 6, 30, 55, 67, 68, 71 and 81.)
- [Sch91] Claus-Peter Schnorr. Efficient Signature Generation by Smart Cards. *J. Cryptology*, 4(3):161–174, 1991. (Cited on pages 57 and 65.)
- [Sha84] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO '84*, pages 47–53, 1984. (Cited on pages 9, 25, 55, 88 and 95.)
- [SKS06] G. Shailaja, K. Phani Kumar, and Ashutosh Saxena. Universal Designated Multi Verifier Signature without Random Oracles. In Saraju P. Mohanty and Anirudha Sahoo, editors, *ICIT '06*, pages 168–171. IEEE Computer Society, 2006. (Cited on page 83.)
- [SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. *Symposium on Cryptography and Information Security (SCIS), Okinawa, Japan*, pages 26–28, January 2000. (Cited on page 88.)
- [SS07] Siamak F. Shahandashti and Reihaneh Safavi-Naini. Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. Cryptology ePrint Archive, Report 2007/462, 2007. <http://eprint.iacr.org/2007/462>. Full version of [SS08]. (Cited on page IX.)
- [SS08] Siamak F. 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 (PKC '08)*, volume 4939 of *Lecture Notes in Computer Science*, pages 121–140. Springer, 2008. (Cited on pages IX and 138.)
- [SS09a] Siamak F. Shahandashti and Reihaneh Safavi-Naini. Generic Constructions for Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. *IET Information Security*, (to appear), 2009. (Cited on page IX.)

- [SS09b] Siamak F. Shahandashti and Reihaneh Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. In Bart Preneel, editor, *AfricaCrypt '09*, volume 5580 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2009. (Cited on pages IX and 139.)
- [SS09c] Siamak F. Shahandashti and Reihaneh Safavi-Naini. Threshold Attribute-Based Signatures and Their Application to Anonymous Credential Systems. Cryptology ePrint Archive, Report 2009/126, 2009. <http://eprint.iacr.org/2009/126>. Full version of [SS09b]. (Cited on page IX.)
- [SSB06] Siamak F. Shahandashti, Reihaneh Safavi-Naini, and Joonsang Baek. Concurrently-secure credential ownership proofs, 2006. Available through corresponding author’s home page: <http://sites.google.com/site/siamax>. Full version of [SSB07]. (Cited on page IX.)
- [SSB07] Siamak F. Shahandashti, Reihaneh Safavi-Naini, and Joonsang Baek. Concurrently-secure credential ownership proofs. In Bao and Miller [BM07], pages 161–172. (Cited on pages IX and 139.)
- [SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In Cramer [Cra05], pages 457–473. (Cited on pages 95 and 101.)
- [SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In Bao et al. [BDZ04], pages 86–100. (Cited on pages 68 and 81.)
- [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 '04*, volume 3108 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2004. (Cited on page 55.)
- [TGO⁺07] Raylin Tso, Juan Manuel González Nieto, Takeshi Okamoto, Colin Boyd, and Eiji Okamoto. Verifier-Key-Flexible Universal Designated-Verifier Signatures. In Steven D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 403–421. Springer, 2007. (Cited on pages 55, 81 and 82.)
- [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 '06 (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 58–69. Springer, 2006. (Cited on page 81.)
- [Ver08] Damien Vergnaud. New Extensions of Pairing-based Signatures into Universal (Multi) Designated Verifier Signatures. *The Computing Research Repository (CoRR)*, abs/0802.1076, 2008. <http://arxiv.org/abs/0802.1076v1>. (Cited on page 83.)
- [Wat05] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [Cra05], pages 114–127. (Cited on pages 41, 65 and 104.)

- [YCD08] Piya Yang, Zhenfu Cao, and Xiaolei Dong. Fuzzy Identity Based Signature. Cryptology ePrint Archive, Report 2008/002, 2008. <http://eprint.iacr.org/2008/002>. (Cited on page 96.)
- [Yi03] Xun Yi. An Identity-Based Signature Scheme from the Weil Pairing. *Communications Letters, IEEE*, 7(2):76–78, 2003. (Cited on page 88.)
- [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 '05*, volume 3531 of *Lecture Notes in Computer Science*, pages 483–498, 2005. (Cited on page 81.)
- [Zhe02] Yuliang Zheng, editor. *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*. Springer, 2002. (Cited on pages 135 and 140.)
- [ZK02] Fangguo Zhang and Kwangjo Kim. ID-Based Blind Signature and Ring Signature from Pairings. In Zheng [Zhe02], pages 533–547. (Cited on page 55.)
- [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, Yuanshun Dai, and Laurence Tianruo Yang, editors, *EUC'05 Workshops*, volume 3823 of *Lecture Notes in Computer Science*, pages 825–834. Springer, 2005. (Cited on pages 55 and 89.)
- [ZSS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In Bao et al. [BDZ04], pages 277–290. (Cited on page 65.)

Index

- Σ protocol, 57
- ABE, *see* attribute-based encryption
- ACS, *see* anonymous credential system
- active adversary, 114
- active attack, 21
- adversary, 14
- anonymous credential system, 114
- asymmetric cryptography, 14
- attribute-based encryption, 95
- binding, 97
- C-EUF-CMAA, 100
- C-signature, 97
- CA, *see* certificate authority
- canonical, 19
- CDH assumption, *see* Computational Diffie-Hellman assumption
- certificate
 - public key —, 14
- certificate authority, 14
- challenge, 19
- challenger, 15
- cheating prover, 22
- cheating verifier, 22
- chosen message and attribute set attack, 98
- chosen message attack, 23, 28
- clone, 22
- CMAA, *see* chosen message and attribute set attack
- commitment, 19
- commitment scheme, 97
- Computational Diffie-Hellman assumption, 17
- concurrent attack, 21
- converted signature, 99
- COP, *see* credential ownership proof
- COP-IMP-AA, 33
- COP-IMP-CA, 33
- COP-IMP-PA, 33
- credential ownership proof, 32
- credential system, 114
- cryptography
 - asymmetric —, 14
 - public key —, 14
- Discrete Logarithm assumption, 17
- DL assumption, *see* Discrete Logarithm assumption
- dummy attribute, 101
- emulation, 15, 97, 115
- environment, 115
- EUF-CMA, 23
- EUF-CMAA, 98
- existential unforgeability, 23
 - identity-based —, 28
- extractor, 20
- FDH, *see* full-domain hash
- Fiat-Shamir transform, 24, 57
- Forking Lemma, 60
- full-domain hash, 45
- game-based security definition, 15
- GQ protocol, 44
- hiding, 97
- honest-verifier zero-knowledge, 20
- HVZK, *see* honest-verifier zero-knowledge
- IBI, *see* identity-based identification scheme
- IBS, *see* identity-based signature scheme
- ID-EUF-CMA, 28
- ID-IMP-AA, 25
- ID-IMP-CA, 25
- ID-IMP-PA, 25

- ideal model, 97
- ideal vs. real paradigm, 97
- identification scheme, 21
- identity-based identification scheme, 25
- identity-based signature scheme, 26
- IMP-AA, 22
- IMP-CA, 22
- IMP-PA, 22
- impersonation, 21
- input
 - private —, 18
 - public —, 18
- interactive proof, 19
- interactive signing protocol, 107
- interactive verification protocol, 99, 107
- key
 - private —, 14
 - public —, 14
 - secret —, 14
- key-generation center, 25
- KGC, *see* key-generation center
- language, 18
- Lemma
 - Forking —, 60
 - Reset —, 61
- move, 18
- non-interactive proof, 19
- NP, 18
- NP relation, 56
- off-line designation, 82
- on-line designation, 82
- one more RSA inversion assumption, 16
- passive attack, 21
- PKI, *see* public key infrastructure
- plain model, *see* standard model
- PoK, *see* proof of knowledge
- private input, 18
- private key, 14
- proof of knowledge, 20
- pseudonym, 114
- public input, 18
- public key, 14
- public key certificate, 14
- public key cryptography, 14
- public key infrastructure, 14
- public-coin, 19
- random oracle model, 15
- Reset Lemma, 61
- response, 19
- round, 18
- RSA, *see* [RSA78]
- RSA assumption, 16
 - one more —, 16
- secret key, 14
- security for the prover, 108
- security for the signer, 108
- security for the user, 108
- security for the verifier, 108
- selective unforgeability, 98
- Σ protocol, 57
- signature scheme, 22
- signer-attribute privacy, 101
- simulation paradigm, 15, 97
- special soundness, 56
- SpS, *see* special soundness
- standard model, 15
- static adversary, 115
- SUF-CMAA, 98
- t-ABACS, *see* threshold attribute-based anonymous credential systems
- t-ABCS, *see* threshold attribute-based C-signature
- t-ABS, *see* threshold attribute-based signature
- threshold attribute-based anonymous credential systems, 114
- threshold attribute-based C-signature, 106
- threshold attribute-based signature, 97
- transcript, 20
- UDVS, *see* universal designated-verifier signature
- UDVSP, *see* universal designated-verifier signature proof

unforgeability, 23, 28

universal designated-verifier signature, 66

universal designated-verifier signature proof, 36

view, 20

WI, *see* witness indistinguishability

witness, 18

witness indistinguishability, 21

zero knowledge, 19

 honest verifier —, 20

zeroth credential, 114

ZK, *see* zero knowledge