

Secure Offline Legitimation Systems

**Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik**

**von
Gerrit Bleumer**

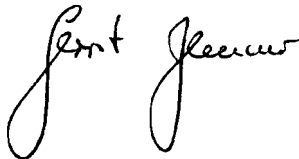
**Dortmund
2001**

Tag der mündlichen Prüfung: 19. Oktober 2001

Dekan: Prof. Dr. Bernd Reusch

Gutachter: Prof. Dr. Joachim Biskup (Erstgutachter),
Prof. Dr. Birgit Pfitzmann (Zweitgutachterin),
Prof. Dr. Horst Wedde,
Prof. Dr. Heiko Krumm (Vorsitzender des Prüfungsausschusses),
Dr. Peter Kemper.

Schildow, den 07. September 2004

A handwritten signature in black ink, appearing to read 'Gerrit Bleumer'. The signature is written in a cursive style with a large initial 'G' and 'B'.

(Gerrit Bleumer)

Foreword

*“Engineers must study not only what technologies can do FOR people
but also what they do TO people,
and they must learn to steer technology
more sensitively and skillfully through the political process.”*

— Edward Wenk [225]

(Science Advisor to three US presidents)

While our modern societies rapidly turn into information societies, powerful players like governments, secret services and public prosecutors ever more strongly demand for complete surveillance of fax, phone, e-mail, payment transactions and more. However, this thesis advocates to design the information infrastructures so that they can respect and enforce the legitimate security interests of ALL participants, be they active users or passive users¹ of the computer systems. So the pleading at the heart of this thesis is:

*As little observability of citizens in every day’s transactions and
as much security against misuse and fraud as possible.*

This is not an impossible dream, this is a vision that can become reality if citizens are getting more conscious and demanding rather than getting distracted by the exciting features of today’s applications. The customers, employers, tax payers, voters, i.e., all individuals of an information society, perform numerous transactions with each other and with provider organizations every day. Almost any of these transactions leaves digital traces about individual clients at the provider’s organization. Since digital traces are easily accumulated into digital profiles on individual behavior, the privacy of any individual is at risk. The problem should neither be disregarded as a hobbyhorse of some frustrated researchers or as the paranoia of some radical dropouts; it is a serious threat to democratic societies. Privacy threats are insidious for two reasons. (i) They are not easy to quantify because its impacts are

1) The term “usee” describes those whose data is processed by computer systems but who usually do not use these computer systems themselves.

diffuse: What happens if patients face the fact that their medical records including genetic fingerprints are released or sold to third parties, e.g., employers, insurance companies, pharmaceutical suppliers? How will citizens (and also small enterprises) react if information about their consuming behavior, social and political activities is available as digital goods on the Web? It is unrealistic to assume that conformity to majority opinions would not increase under such conditions. (ii) Privacy threats are easily overlooked because individuals do not *feel* to be observed, therefore do not feel to be threatened individually, and therefore feel little need to defend themselves appropriately. Unlike countries or large enterprises engaged in *information warfare* [97], individuals seldom know their observers, can hardly measure their loss of privacy, are usually not organized and only few are skilled to take appropriate countermeasures. (iii) Privacy threats are ubiquitous through the installation of video cameras and other biometric scanning technology. Places of public interest, roads, tunnels, bridges, gas stations, train stations, airports, and ATM are only a few examples of non-stop video monitoring. The next generation of public key infrastructures will replace passwords by biometric recognition facilities. A profound overview of surveillance technologies has been compiled by DuncanCampbell [55] for the STOA unit of the European Parliament and was presented to the European Parliament. Simson Garfinkel has published a thorough analysis of privacy threats in the US [115].

The threat to privacy is illustrated by an episode in 1996 that has been partially documented at the Cambridge Workshop on “Personal Medical Information — Security, Engineering, and Ethics” [2]. The British National Health Service (NHS) had proposed to build a UK-wide medical network in order to increase efficiency of all transactions between General Practitioners (GPs), clinics, hospitals, pharmacies and other points of care. After serious and persistent requests, it was promised that patient data should not be sent in clear but only in encrypted form and the NHS as a national organization had therefore looked for a method that was acceptable to the intelligence community. The NHS was advised (if not oppressed) to employ an unpublished encryption mechanism called “redpike”. The service’s obvious intention was to keep easy access to all data sent along the medical network. Had only the patients been affected, this proposal supposedly would have been accepted without much noise. In this case however, many medical doctors have felt not only their patients’ privacy threatened but also their own, and so the British Medical Association (BMA) started a campaign against the NHS proposal. (The controversy died out later because the NHS did not pursue the original plan further.)

Evidently, anonymity is not an end in itself. In small communities, anonymity can be counterproductive, and even in large societies it has to be balanced against other legitimate interests like for example fighting organized crime, terrorism and money laundry. It shall be shown that for not so small classes of applications fair privacy protecting solutions exist. It must be left to the democratic process to decide about which technical infrastructures we want to face in the next millennium.

Abstract

Organizing the interdependencies within and between communities is one of the ongoing challenges of mankind. Once organizations are formed, companies run their businesses, and a legal system is in place, there is an urgent need for procedures to perform legally binding transactions. This in turn brings up the need for unforgeable documents or tokens of legitimation. Traditional examples are letters and cheques with handwritten signatures or seals, hard-to-counterfeit bills, drivers licences and passports with hardly removable pictures imprinted, etc. The implementations of legitimations change as the technological paradigms change, but the need for legitimations persists. In information societies, many of the traditional implementations are obsolete because they are no longer efficient and often too costly. In addition, information technology often provides better approximations to the ideal properties of legitimations, e.g. unforgeability. Electronic commerce is one if not the pioneering area where the new implementations of legitimations are developed, tested and put into everyday's practice. Examples are electronic wallets, phone cards, e-cash, e-tickets, etc.

While an amount of money can be regarded as a legitimation to consume a corresponding portion of the national gross product, there are also other kinds of legitimations. This work starts by categorizing them and identifying important examples in real life. Namely, we distinguish *personal* and *coin legitimations*. The former cannot be transferred between holders and the latter cannot be used more often than a pre-specified limit. Orthogonal to these categories then are privacy requirements. This is where electronic implementations are really superior to traditional implementations: Not only are they more efficient, but they can achieve more privacy for holders of legitimations than the conventional paper based implementations can. Such electronic implementations have been introduced in 1985 by Chaum [60] as *credentials*. Holders can get a credential from an issuer and later show it to a verifier without letting the issuer and verifiers recognize that they have issued and verified a credential of the same holder (*unlinkability*). Although several cryptographic mechanisms for credentials have been suggested since, formal definitions have been given only for the special case of electronic cash. We propose a formal modular framework to define the different categories of credentials sketched above (including electronic cash). Furthermore, we suggest the first mechanism for personal credentials that can be shown many times in an unlinkable way. In order to achieve non-transferability, we suggest the use of

biometric verification of holders without releasing any biometric data to more or less centralized databases where they could be aggregated, analyzed and re-used in unintended ways.

In addition to privacy of holders, we also consider privacy of issuers of credentials (against verifiers). This turns out to be useful in more complex applications. The final section presents the detailed design of how compulsory health insurances can be billed without the health insurers even learning which physician is treating which patient, let alone which patient gets which therapy or medicament. This way, the trust relationship between patients and physicians can be protected optimally, and it is nevertheless possible to identify falsely claiming physicians after the fact.

Kurzfassung

Eine ständige Herausforderung jeder Gesellschaft ist es, einen verlässlichen Rahmen für die Beziehungen ihrer Mitglieder (Individuen und Gruppen) herzustellen und aufrechtzuerhalten. Sobald Organisationen gegründet sind, Firmen ihre Geschäfte betreiben und ein Rechtssystem installiert ist, werden Verfahren zur rechtsverbindlichen Interaktion benötigt. Dies wiederum erfordert fälschungssichere Dokumente oder Ausweise. Traditionelle Beispiele sind gesiegelte oder unterschriebene Briefe, unterschriebene Schecks, schwer fälschbare Geldscheine, Führerscheine oder Reisepässe mit aufgedruckten Passbildern. Die Form der Legitimationen mag sich entsprechend der technologischen Paradigmen einer Gesellschaft verändern und weiterentwickeln, aber die grundsätzliche Notwendigkeit von Legitimationen bleibt bestehen. In einer Informationsgesellschaft sind viele der herkömmlichen, d.h. papiergestützten Formen überholt, weil sie zu ineffizient und oft auch zu teuer sind. Überdies erlaubt Informationstechnologie häufig bessere Annäherungen an die idealen Eigenschaften von Legitimationen, z.B. Unfälschbarkeit. E-Commerce, ist eines wenn nicht sogar das führende Gebiet, auf dem die zukünftigen Formen von Legitimationen entwickelt, erprobt, und wahrscheinlich auch flächendeckend eingesetzt werden. Beispiele sind elektronische Brieftaschen, Geldkarten, e-cash, e-tickets, etc.

Es gibt neben Geld weitere Sorten von Legitimationen², und die vorliegende Arbeit beginnt mit ihrer Klassifizierung illustriert durch praktische Beispiele. Im wesentlichen unterscheiden wir zwischen persönlichen und Münz-Legitimationen. Erstere können unter Besitzern nicht weitergegeben werden, während letztere nur begrenzt oft benutzt werden können. Orthogonal zu dieser Unterscheidung betrachten wir Anonymitätsanforderungen—oder stärker Unverkettbarkeitsanforderungen der Besitzer von Legitimationen. Auf diesem Gebiet bieten digitale Implementierungen von Legitimationen qualitative Vorteile gegenüber herkömmlichen: Sie sind nicht nur effizienter herzustellen, zu speichern und zu prüfen, sondern können tatsächlich Unverkettbarkeit der Transaktionen desselben Besitzers erzielen. Kryptographische Implementierungen sind erstmals 1985 von Chaum [60] untersucht und unter dem

2) Volkswirtschaftlich kann Geld als Legitimation oder Anspruch auf einen entsprechenden Teil des Bruttosozialprodukts angesehen werden.

Begriff *Credentials* eingeführt worden. Hier kann ein Besitzer sein Credential von einem Anbieter bekommen und es später einem Dritten (Prüfer) zeigen, ohne daß Anbieter und Prüfer hinterher erkennen könnten, daß sie mit demselben Besitzer zu tun hatten. Obwohl seitdem mehrere Verfahren für Credentials vorgeschlagen worden sind, gibt es formale Definitionen bisher nur für den Spezialfall Münz-Credentials. Wir geben eine formale modulare Definition für alle oben beschriebenen Arten von Credentials. Weiterhin geben wir die erste Konstruktion für persönliche Credentials, die mehrfach unverkettbar gezeigt werden können. Um Weitergabe der Credentials zu verhindern, untersuchen wir den Einsatz biometrischer Erkennungsverfahren, wobei die biometrischen Daten der Credentialbesitzer nicht in zentrale Datenbanken gelangen können, in denen sie gesammelt, analysiert und in unerwünschter Weise weiterverwendet werden könnten.

Über die Unverkettbarkeitsforderungen von Besitzern hinaus betrachten wir hier erstmals auch Anonymitätsforderungen der Anbieter gegen Prüfer von Credentials. Dies kann in komplexeren Anwendungsgebieten wünschenswert oder nötig sein. Im letzten Kapitel entwerfen wir detailliert, wie mit gesetzlichen Krankenversicherungen so abgerechnet werden kann, daß die Versicherer nicht einmal erfahren, welcher Arzt welchen Versicherten behandelt, geschweige denn welcher Patient welche Behandlung und welches Medikament bekommt. Auf diese Weise wird das Vertrauensverhältnis zwischen Arzt und Patient optimal geschützt, und dennoch können Ärzte nachträglich identifiziert und zur Verantwortung gezogen werden, wenn sie nicht erbrachte Leistungen abrechnen oder überhöhte Gebühren in Rechnung stellen.

Acknowledgments

My research period at the University of Hildesheim started with Joachim Biskup as senior supervisor and Andreas Pfitzmann and Birgit Pfitzmann as junior supervisors. I have much appreciated the liberal and productive atmosphere that Joachim Biskup has created and has let grow. He did not get nervous when I took several weeks looking after a new kind of protocol but showed interest in everything I did. Without this period of unfettered research this thesis would not have been finished. Back in 1990, Joachim Biskup proposed to explore security issues in medical payment systems [18] under the AIM program (Advance Informatics in Medicine) [78], which was sponsored by the European Commission from 1992 through 1995 by about 120 Mio. ECU. Under that program, I have made a lot of experience about security and other aspects of computerizing the medical sector.

About five years ago, Andreas Pfitzmann suggested to me the high level idea of credentials that do not only protect the privacy of holders but also that of issuers. At that time, none of us had the faintest idea how to arrive at practical solutions. The cryptographic primitives that finally turned out to help solving the problem had only just been published and in a sense, the idea was ahead of its time.

At about that time, Birgit Pfitzmann pointed out to me that the concept of personal credentials, i.e., credentials that refer to a particular human being, need to include some biometric mechanism in order to link the digital credentials to the biological beings in a non-repudiable way. This problem had been considered earlier by Chaum in his patent [62], and was later reconsidered by Damgård in the concluding paragraph of [83]. Birgit has supported my work more than anyone else over the years; she has suggested designs of definitions, motivated me to lay as formal foundations as possible in this area, relentlessly cryptanalyzed my protocols and did thorough proof reading.

Together with Matthias Schunter I have looked for applications of the new cryptographic mechanisms. His background in electronic commerce (EU-CAFE) and my own background in security of health information systems (EU-SEISMED, EU-ISHTAR) made us write a paper on charging and reimbursing medical treatment in case of compulsory health insurances. Matthias was a creative partner in playing the “yes, it must work — no, it can’t” game, that has been reported to make engineering of new cryptographic schemes both productive and entertaining and sometimes causing headaches. (See Ron Rivest when he found the RSA algorithm in 1977 [114,p74].) It is even more exciting if you

change the roles from time to time. If I am asked why I am researching, I say “curiosity”, if I am asked why in the field of computer security, I say “social responsibility to a large extent”, but if I am asked, what makes me go on with research, I must admit it is the stimulation and inspiration I get from experiences like the above.

Michael Waidner has become a “background partner” to me. Although extremely busy, he has continued motivating me, has pointed me to important related work and always showed interest in this work.

I thank all my colleagues in the SEISMED and ISHTAR consortia (both sponsored by the Commission of the European Union) for patiently introducing a non-medical fellow like me into the field of security in medical informatics and the corresponding working group of the international medical informatics association (IMIA). I owe important insights to Ab Bakker, Barry Barber, Bernd Blobel, John Davey, Sokratis Katsikas, Eike-Henner Kluge, Erik Flikkenschild, Kees Louwerse and many others.

During two years of this work, I have been supported by AT&T Labs and I have much enjoyed the hospitality and encouragement of David Maher and his group. It was a pleasure to discuss parts of this work with Matt Franklin, Andrew Odlyzko, Mike Reiter and Serge Vaudenay.

The layout of this book is influenced by “Concrete Mathematics” [116]. In their book, Graham, Knuth and Patashnik not only provide a steady source of mathematical and humorous creativity, but also a paragon of excellent typesetting and layout.

Finally, I am deeply indebted to my wife Anne and my children Thilo, Tobias and Viola for their support and patience with me.

Table of Contents

Foreword	iii
Abstract	v
Kurzfassung	vii
Acknowledgments	ix
Table of Contents	xi
Figures and Tables	xv
1	Introduction ----- 1
1.1	Contributions to Cryptography 2
1.2	Contributions to Medical Informatics 3
1.3	Organization of the Work 4
1.4	System Complexity and the Rigidity of Security Arguments 4
2	A Guided Tour to Legitimizations ----- 5
2.1	Basic Legitimation Systems 6
2.2	Effectiveness and Security Requirements 6
2.2.1	Effectiveness 7
2.2.2	Issuer Legitimation 7
2.2.3	Holder Authorization 7
2.2.4	Certification 7
2.2.5	Holder Privacy 7
2.2.6	Issuer Privacy 7
2.3	Real-Life Examples 7
2.4	Categorization of Basic Legitimation Systems 9
2.5	More Complex Legitimation Systems 11

3	Notations and Assumptions	13
3.1	General Notation	13
3.1.1	Probabilities	13
3.1.2	Algebraics	14
3.1.3	Asymptotics	14
3.2	Protocols, Semantics, Interfaces and Operations	15
3.2.1	Execution Functions	16
3.2.2	Internal and External Interfaces	19
3.2.3	Protocol Composition and Decomposition	21
3.3	Cryptographic Schemes and Mechanisms	22
3.3.1	Notational Conventions	23
3.3.2	Security Requirements and Attackers	24
3.3.3	Generating Prekeys	25
3.3.4	Encoding Domain Elements	26
3.3.5	Participant Identifiers	26
3.3.6	Typed Parameters	26
3.3.7	Notation of Protocols	26
3.4	Discrete Log Frameworks and Settings	27
3.5	Double Discrete Log Frameworks and Settings	29
3.6	Complexity Theoretic Assumptions	29
3.6.1	Subgroup Discrete Log Assumption	29
3.6.2	Subgroup Representation Assumption	30
3.6.3	Subgroup Diffie-Hellman Assumption	30
3.7	Random Oracle Model	31
4	Cryptographic Tools	33
4.1	Proof-of-Knowledge Schemes	34
4.1.1	Definition of Proof-of-Knowledge Schemes	34
4.1.2	Proving Linear Relations of Witness Components	37
4.1.3	Zero-Knowledge, Witness Indistinguishability, and Witness Hiding	37
4.1.4	Extended Proof-of-Knowledge Schemes	39
4.1.5	Related Work on Proofs-of-Knowledge	42
4.1.6	Chaum-Pedersen-Van De Graaf Proof-of-Knowledge Mechanism	43
4.1.7	Brands Proof-of-Knowledge Mechanism	46
4.1.8	Extended Chaum-Pedersen-Van De Graaf Proof-of-Knowledge Mechanism	47
4.2	Blind Signature Schemes	51
4.2.1	Definition of Blind Signature Schemes	52
4.2.2	Chaum-Pedersen Signature Mechanism	55
5	Credential Schemes	59
5.1	Overview	60
5.1.1	Core Credential Schemes	62

5.1.2	Issuer Groups and Issuer Anonymity	66
5.1.3	Mobile User Devices	67
5.1.4	Embedded Security Modules	68
5.1.5	Biometric Sensors	69
5.1.6	Summary and New Achievements	70
5.2	Definition of Credential Schemes	71
5.2.1	Preparation	72
5.2.2	General Definition	74
5.3	Security Definitions of Credential Schemes	81
5.3.1	Overview of Security Requirements	81
5.3.2	Overview of Attacker Models	82
5.3.3	Definitions of Integrity Requirements	83
5.3.4	Definitions of Holder Privacy	84
5.3.5	Interrelations between Integrity and Privacy Definitions	89
5.4	Personal Credential Schemes	90
5.4.1	Definition	90
5.4.2	The Cryptographic Mechanism in a Nutshell	90
5.4.3	Use of Pseudonyms, Signatures, Witnesses and Co-Witnesses	91
5.4.4	Personal Credential Mechanism	92
5.5	Coin Credential Schemes	110
5.5.1	Definition	110
5.5.2	Coin Credential Mechanism	110
5.6	Bond Credential Schemes	121
5.6.1	Definition	121
5.6.2	Cryptographic Mechanism	121
5.7	Conjunctions of Personal and Coin Credentials	121
6	Group Signature Schemes	123
6.1	Overview of Existing Literature	123
6.1.1	Options and Features	123
6.1.2	Existing Work	124
6.2	New Achievements	125
6.3	Blind Static Group Signature Schemes	126
6.4	Cryptographic Mechanism	129
6.4.1	Efficiency Comparison	138
7	Group Credential Schemes	141
7.1	Definition of Group Credential Schemes	141
7.2	New Cryptographic Mechanisms	143
7.2.1	Coin Group Credential Scheme	143
7.3	Conjunctions of Personal and Coin Credentials with Static Group Credentials	143
8	Mobile Patient Assistants	145

8.1	Introduction	145
8.2	Contractual Framework	146
8.3	Paper-based Charging and Clearing	148
8.3.1	Analysis	148
8.3.2	Participants and Their Specific Security Requirements	149
8.3.3	Exceptions	151
8.4	Implementation Options	151
8.4.1	Enrollments, Referrals and Prescriptions	151
8.4.2	Anonymity Options	153
8.4.3	Cryptographic Extensions	153
8.4.4	Implementing the Legitimations	153
8.4.5	Charging for Medical Treatment	154
8.4.6	Charging for Paramedical or Specialist Treatment and Medications	155
8.4.7	Capping the Total Annual Reimbursement	158
8.4.8	Discussion	158
8.5	Implementation of the Clearing Process	159
8.5.1	Initialization	159
8.5.2	Policy Holders' Views	161
8.5.3	Healthcare Providers' Views	163
8.5.4	Limiting the Total Cost	164
8.6	Security	164
8.6.1	Availability and Integrity Requirements	164
8.6.2	Privacy Requirements	165
9	Conclusions and Open Questions	167
A	Index of Symbols	A-1
B	Index of Authors	B-1
C	References	C-1
C.1	Bibliography	C-1
C.2	Web Links	C-13
D	Index of Keywords	D-1

Figures and Tables

Figure 2-1	School Reports	8
Figure 2-3	Bonds, Stocks	9
Figure 2-2	Electronic Coins	9
Figure 3-1	Diffie-Hellman Key Agreement	27
Figure 4-1	Communication Topology of a 2-Prover Proof of Knowledge	40
Figure 4-2	Chaum-Evertse-Van De Graaf Proof of Knowledge: <i>prove</i>	43
Figure 4-3	Brands Proof of Knowledge: <i>proveWitRel</i>	47
Figure 4-4	Extended Proof of Knowledge: <i>proveExt</i>	48
Figure 4-5	Producing a Chaum-Pedersen Signature	56
Figure 5-1	Relations of the Core Credential Schemes	63
Figure 5-2	Configuring an SPM, a MUD and a Security Module S	68
Figure 5-3	Embedding a Biometer	69
Table 5-1	Summary of Security Requirements	71
Table 5-2	Overview of References about Offline Credential Schemes	71
Figure 5-4	Overview of Pseudonyms	73
Figure 5-5	Setup of Issuer and Observer Keys	74
Table 5-3	Overview of Attacker Models for given Security Requirements	82
Table 5-4	Types of Unlinkability and underlying attacker coalitions	85
Figure 5-6	Tree of credentials evolving from an explicitly introduced root pseudonym	85
Table 5-5	Views on MUD D resulting from executions of	87
Table 5-6	Views on MUD D resulting from executions of	88
Table 5-8	Use of Pseudonyms and Witnesses During <i>Show</i>	92
Table 5-7	Use of Pseudonyms and Witnesses During <i>Intro</i> and <i>Issue</i>	92
Figure 5-7	Generating an Issuer Key Pair	93
Figure 5-9	Introducing a Pseudonym	94

Figure 5-8	Personalizing an Observer	94
Figure 5-10	Issuing a Personal Credential	95
Figure 5-11	Showing a Personal Credential for a new Target Pseudonym	96
Figure 5-12	Showing a Personal Credential for a re-used Target Pseudonym	97
Table 5-9	Attackers and their views on the MUD in <i>Lifen</i>	105
Table 5-10	Attackers and their views on the MUD in <i>lifen</i>	108
Figure 5-13	Introducing a Pseudonym	112
Figure 5-14	Issuing a Coin Credential	113
Figure 5-15	Showing a Coin Credential	114
Figure 5-16	Extracting a Witness	114
Table 5-11	Attackers and their views on the MUD in <i>Life1</i>	119
Figure 6-1	Proving Knowledge of one-out-of- n Witnesses	130
Figure 6-2	Producing a Chen-Pedersen Group Signature	132
Figure 6-3	Identifying the Signer	133
Table 6-1	Computational and Communication Effort	138
Table 6-2	Bit Lengths of Public Keys and Signatures	138
Figure 8-1	Contractual Framework	147
Figure 8-2	Flow of Billing Related Information	149
Figure 8-3	Use of Patient Legitimations.	152
Figure 8-4	Showing an Enrollment and a Referral for one Target Pseudonym	155
Table 8-1	Combinations of Implementations and their Characteristics	156
Figure 8-5	Initialization Phase	160
Figure 8-6	Charging and Clearing of Referrals and Prescriptions	161
Figure 8-7	Charging and Clearing of Medical Treatment	164

1

Introduction

More and more interactions within and between the sectors of our information societies rely upon electronic services and media. The work flows in business, industry, healthcare, administration, insurance, press and TV are getting more mediated by computer networks and they provide more and more digital services to each other and to consumers; for example, electronic commerce, electronic cash, credit cards, Pay-TV, insurance cards, etc. In general, *service consumers* request access to certain services that are provided by *service providers*. Usually, access to a service shall or must be conditioned by a legitimization of the provider (to offer the service) and by a legitimization of the consumer (to obtain the service). Furthermore, providers and consumers may have an interest in protecting their privacy—against each other, or against third parties. In particular, legitimization systems need to ensure:

INTEGRITY REQUIREMENTS

- (i) *Provider Legitimation*: Only¹ licensed providers shall provide a service.
- (ii) *Consumer Legitimation*: Only authorized consumers shall access a service.
- (iii) *Unforgeability*: Legitimizations of providers and consumers cannot be forged.

CONFIDENTIALITY REQUIREMENTS

- (iv) *Privacy Protection*: Service consumers and service providers shall not be unnecessarily observable. The most interesting options are to protect the consumers' privacy against the providers and third parties and the providers' privacy against third parties who might verify legitimizations later on.

Legitimizations to access a service can be much more sophisticated than just by a provider and a consumer legitimization. In general, legitimizations can additionally depend on the time or the order of access

1) Alternatively, we could have required the stronger “iff” instead of an “if only”. This would imply an availability requirement, namely that every provider who is licenced also provides the respective service. As usual, such availability requirements cannot be enforced by digital means, and not even by legal proceedings. Putting a service to action relies completely on the provider.

1 INTRODUCTION

requests, the number of consumers participating in one access request (*dual control*), etc. All these extensions are not considered explicitly in the following, but the solutions presented can be extended to reflect some of them as well.

Digital legitimation systems are to create, control, manage or delete the legitimations to use services in a legally effective way. From now on, we consider the legitimations themselves as services, which can be provided and obtained. The following work outlines a modular framework for legitimation systems that satisfy the above security requirements (i)-(iv). Two optional kinds of consumer legitimation are distinguished: Physical identity of the consumer and consumability of legitimations. Combinations of these properties result in the following four categories of legitimations.

Free Legitimations are not personalized and are not consumed when being used. They can be freely copied and transferred among holders.

Personal Legitimations are personalized to human holders but are not consumed when being used. Holders can make copies of their personal legitimations but cannot transfer them among each other (e.g., driver's licences).

Coin Legitimations are not personalized but are consumed when being used. Holders cannot make copies of their coin legitimations but can transfer them among each other (e.g., electronic coins, e-cash).

Bond Legitimations are personalized and are consumed when being used. They can neither be copied nor transferred among holders.

More complex legitimations can be constructed by combining legitimations of the basic categories (e.g., driver's licence AND e-cash, etc.). Furthermore, hierarchies of legitimations are common, e.g. regional authorities provide licences to schools to provide reports. The reports in turn serve to the graduates as licences to attend a college, etc. Surely more work is needed to cover other features, like effective service availability, i.e., fault tolerance, delegation, etc. All these extensions are not covered in the sequel.

We consider legitimation systems of each of the four basic categories, give semi-formal security definitions with respect to requirements (i)-(iv) above and propose efficient cryptographic implementations, i.e. without using cut-and-choose techniques. Since there is considerable work on coin legitimations (e.g., electronic coin systems, payment systems, etc.), we focus upon the other categories and, orthogonally, upon service provider privacy. Finally, we present a complete solution to reimburse expenses for medical treatment and medicaments by compulsory health insurances, which is applied to the German health care system as an example.

We follow the pioneering work of David Chaum [58,60,65] who has introduced the term *credentials* for legitimations that achieve strong privacy for holders by allowing them to use different pseudonyms in their various relationships with issuers and verifiers.

1.1 Contributions to Cryptography

This work introduces notations both for defining cryptographic schemes and for constructing particular instances (Section 3 on p.13). The definitory part uses ideas of the theory on abstract data types.

The constructive part uses a kind of declarative programming language that includes calls of local or multi-party sub-protocols. The notation captures security parameters, pre-computed constants of a scheme, generation of keys and other local or multi-party operations as well as integrity and confidentiality properties, be they formulated probabilistically, algebraically or asymptotically.

Extended proof-of-knowledge schemes are introduced in Section 4.1 on p.34. They are a generalization of diverted proofs of knowledge first defined by Okamoto, Ohta [173].

A personal credential scheme is defined and an efficient construction is presented in Section 5.4 on p.90. Personal cryptographic modules of each holder bind their credentials to their respective biometric identities. Such credentials cannot be transferred among holders unless they break their personal cryptographic modules. However, the cryptographic modules cannot accumulate information of past interactions nor can they release (bits of) the biometric identities of their holders. Moreover the issuing and showing of such credentials is unlinkable even against collaborating issuers and verifiers.

A new restrictive blind group signature is presented in Section 6.4 on p.129. It can replace the restrictive blind signature scheme in the construction of coin credentials in Section 5.5 on p.110 in order to obtain a coin group credential scheme. This is described in Section 7.2 on p.143.

1.2 Contributions to Medical Informatics

Computerizing the health care sector is a necessity and a substantial threat at the same time [19]. It is necessary in order to increase efficiency and to keep cost under control, and it introduces the threat of exposing highly sensitive and personal data to unintended persons. The threat is substantial for two reasons:

- Computerized medical and billing records will, once they are established, almost certainly be accessible also by unintended insiders, i.e. personnel of hospitals or general practices, or outsiders, i.e., adversary attorneys or corporations like health insurers or employers of a patient.
- An increasing number of health care providers is involved in the modern caring processes (*shared care*). The more practises, specialists, pharmacists, etc. are involved in each case, the more difficult it becomes to keep the door closed against unintended outflow of patient data.

It is a major challenge to design medical and billing records that reflect the diversity of health care processes; and it is an even bigger challenge to properly respect the legitimate privacy interests of the patients and the physicians. It is by far not enough to simply plug in a secure database or a secure payment system. This can be seen also from the various standardizing activities CEN TC 251 [231], OMG-CORBAMED [240] and others. A recent overview by Grimson, Grimson and Hasselbring is found in [137].

A particular problem arises in the case where cost for medical treatment and medicaments are reimbursed by a compulsory health insurer. Compulsory health insurers charge their policy holders according to the income, not the state of health of the policy holders. In Germany, for example, compulsory health insurers turn over about half of all cost in health care. In contrast to private health insurers who want to know which policy holder incurs which cost, compulsory health insurers do not generally need to know this correspondence. In this case, credential schemes and group credential

schemes can be applied in order to achieve sufficient privacy for the patient-doctor relationships against health insurers (Section 8 on p.145).

1.3 Organization of the Work

An informal introduction to legitimations and their typical integrity and confidentiality requirements is given in Section 2 on p.5. Different integrity requirements naturally induce the categorization into free, personal, coin and bond legitimation systems suggested earlier. The notation and cryptographic assumptions for sections 4 through 7 are summarized in Section 3 on p.13. The fundamental cryptographic primitives (proofs of knowledge and restrictive blind signatures) are introduced in Section 4 on p.33. Section 5 on p.59 presents the definitions of the four categories of credential primitives. In Section 6 on p.123 we give a definition of blind group signatures and suggest efficient implementations for one-time blind group signatures. In Section 7 on p.141, we define group credential primitives. Finally, we show in Section 8 on p.145 how compulsory health insurances can be charged in a privacy protecting but yet unforgeable way in order to reimburse expenses for medical treatment and medications. All the primitives introduced have proved to be useful for this challenging task in one way or another.

1.4 System Complexity and the Rigidity of Security Arguments

The cryptographic mechanisms presented in this work span from relatively simple and well known mechanisms to highly complex interactive protocols using the simpler mechanisms as building blocks and involving several participants. We are going to present proofs of security for simpler cryptographic mechanisms and gradually decrease the rigidity of security arguments when we proceed to more complex cryptographic systems. We make this approach explicit by stating the security claims of simpler mechanisms as propositions, while stating the security claims of more complex cryptographic mechanisms as “security suggestions”. The security arguments following “security suggestions” are termed “security considerations”, which indicates less rigidity than a mathematical proof and allows some degree of heuristic reasoning. We recognize that some of the security considerations have a potential of being formalized more deeply and using less heuristic reasoning. Quite possibly the cryptographic mechanisms themselves may have a potential to be modified so as to allow more rigid security considerations. This may be subject to further research.

2

A Guided Tour to Legitimizations

“The issues that divide or unite people in society are settled not only in the institutions and practices of politics proper, but also, and less obviously, in tangible arrangements of steel and concrete, wires and semiconductors, nuts and bolts.”
— Langdon Winner [226]

A *legitimation* is any means to authorize and possibly also to control access to resources. Those who issue and/or verify legitimizations are called *issuers* and *verifiers*, respectively; or simply *providers* if we do not distinguish them. Note that “provider” is a rather broad term here that includes any kind of verifier—even, for example, courts. Those who receive legitimizations are called *owners*. In general, each party, i.e., natural person or legal entity, can take one or more of the above roles.

In the following, we will regard legitimizations as permissions or privileges of their owners. In principle, legitimizations can as well represent obligations, prohibitions, or compulsions of their owner. We shortly come back to these in Section 2.5 on p.11. Legitimizations are useful for providers to co-ordinate many parties’ demands for certain resources, which in turn might be legitimizations themselves. There are countless examples for legitimizations for more or less specified services in all societal, business or healthcare affairs: identity cards, membership cards, drivers’ licenses, insurance policies, school reports, cash, medical prescriptions, voting permits, etc. Sometimes, owners keep their legitimizations by means of personal physical tokens such as plastic-, magnetic-, inductive-, or smart cards, or certified paper document. Alternatively, owners may delegate to manage their legitimizations to some agent.

We are interested in designing *legitimation systems*, i.e., computer systems to manage legitimizations. Since many kinds of legitimizations are considered personal privileges, their owners will seek for *personal control* over these legitimizations. As the growing popularity of organizers and other *mobile user devices (MUD)* [185] shows, owners are willing to put significant trust into equipment under their own control and we therefore consider mobile user devices a promising approach for owners to keep and manage their legitimizations. A mobile user device is a piece of autonomous hardware with its own power supply, computing, storing, input/output facilities, and equipped with proper software. For instance think of a personal digital assistant, organizer, palmtop computer or programmable cell

phone. Providers often use more powerful stationary provider machines like POS-terminals or ATMs. We are particularly interested in *off-line legitimation systems*, which allow holders to connect their mobile user devices to each other or to stationary provider machines and which do not require (remote) third parties being on-line in order to process transactions. Connections can be made either by point-to-point connection (serial, infrared, direct plugging) or via some network (wired or wireless).

2.1 Basic Legitimation Systems

Basic legitimations have so simple access structures that *ownership* is a meaningful concept, i.e., we assume by default that basic legitimations are issued to an individual, the owner. (We consider more complex legitimations like medical records in Section 2.5 on p.11.) Basic legitimations are characterized by four quantities

<i>Issuer</i>	party who has issued or provided the basic legitimation.
<i>Content</i>	specifies a right or privilege, e.g., in the real world,
<i>Owner</i>	party to whom the basic legitimation has been issued,
<i> Holders</i>	one or more parties who actually hold a digital instance (copy) of a basic legitimation. Such an instance is called a <i>certificate</i> ¹ .

The issuer creates a basic legitimation by assigning some privilege (content) to an individual (owner). By issuing a respective certificate, the issuer creates an instance of the basic legitimation. Typically, the issuer provides the certificate to its owner, but may as well provide it to a third party who receives it in behalf of the owner. Anyone who holds an instance (certificate) of a basic legitimation is called a holder of it. In general, different holders can hold certificates, of the same basic legitimation. Two certificates of the same basic legitimation can be different or exact copies of one another.

2.2 Effectiveness and Security Requirements

It is common to distinguish effectiveness (in case of no attacks) and availability, integrity, confidentiality (in case of attacks). Effectiveness (Section 2.2.1 on p.7) is a general requirement on each system. Availability is not considered here, because it cannot be enforced by digital means alone. On basic legitimation systems there are three integrity requirements, namely legitimation of issuers (Section 2.2.2 on p.7) and authorization of holders (Section 2.2.3 on p.7) and certification of certificates (Section 2.2.4 on p.7). The *integrity requirements* serve the issuers' interests in the first place. The specific *confidentiality requirements* of basic legitimation systems are consumer privacy and issuer privacy. The interesting part of consumer privacy is to keep holders anonymous against verifiers and, as far as possible, also against issuers. (The owner of a basic legitimation can be anonymized by using a pseudonym identifier in the respective certificates.) We consider holder privacy in Section 2.2.5 on p.7. The interesting part of issuer privacy is keeping issuers anonymous against verifiers. We consider

1) Note that the terminology about "certificates" is far from consistent throughout the literature and the evolving variety of computer products, particularly authentication systems [176].

issuer privacy in Section 2.2.6 on p.7. Clearly, holder and issuer privacy serve the interests of holders and issuers, respectively.

2.2.1 Effectiveness

If a licensed issuer provides a basic legitimation, each of its instances (certificates) is acceptable to any verifier.

2.2.2 Issuer Legitimation

Based on a directory of issuers and any instance (certificate) of a given basic legitimation, a verifier can check if the issuer was legitimated to issue the basic legitimation (*issuer legitimation*).

2.2.3 Holder Authorization

Based on any instance (certificate) of a given basic legitimation, a verifier can check if the actual holder uses the certificate in a legitimate way (*holder authorization*). Some types of basic legitimations can be used by any holder. Other types of basic legitimations can be used by their owner only. Still other types can be used according to how often they have been used before or when or where they are used.

A basic legitimation may change in certain ways when it is used, and thereby its holder authorization may change accordingly. For some types of basic legitimations once they are issued their content and owner cannot be changed anymore. Other types of basic legitimations automatically change their owner when they are used. For example, the previous owner loses them like a physical token that is given away, and the one to whom they are presented becomes the new owner.

2.2.4 Certification

Based on any instance (certificate) of a given basic legitimation, a verifier can check if the content and owner are properly certified.

2.2.5 Holder Privacy

Certificates of a basic legitimation shall not reveal the identity of their holders to any verifier. Even stronger, if an issuer issues two certificates to holders who later present them to a verifier, then the verifier shall be unable to tell, even if he collaborates with the issuer, which issuing and which showing refers to the same holder.

2.2.6 Issuer Privacy

Certificates of a basic legitimation shall not reveal the identity of their issuers to any verifier. Since issuers shall be kept responsible for the basic legitimations they issue, there needs to be a mechanism to later deanonymize issuers. Therefore, we will only consider the case that issuers are anonymized within a group of equally legitimated issuers. Deanonymizing an issuer shall be possible only by the help of her managing group.

2.3 Real-Life Examples

Before we are going to categorize basic legitimations, consider the following real life examples:

2 A GUIDED TOUR TO LEGITIMATIONS

Example (i) (CERTIFIED DOCUMENT) An organization, company, or institute can certify documents such that recipients of those documents can prove to third parties the validity of the content of that document (for example contracts). Holders of a certified document can show it as often as they like, regardless who they are or who the owner of that document is. By showing a certified document to a third party, the actual holder may or may not reveal her identity.

Example (ii) (SCHOOL REPORTS) Schools² can *issue* school reports to students who have passed their exams. Graduates can *show* their reports, e.g., to companies or universities in order to apply for jobs or university courses. Verifiers accept school reports if they are valid and the issuing school is properly accredited. School reports are not consumed after being shown, but their content and owner cannot be changed. The typical stages of using a school report are depicted in Figure 2–1 on p.8.

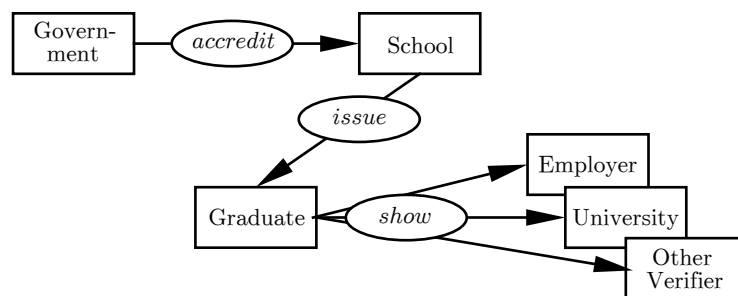


FIGURE 2–1 School Reports

Example (iii) (DRIVERS’S LICENSES) After passing some theoretical and a driving test, one is issued a driver’s license. For example, on the occasion of a vehicle spot check, these licenses have to be shown to a police patrol. A driver’s licence allows the police patrol to check whether the individual actually driving, i.e. the holder, is in fact the individual shown on the portrait of the driver’s licence, i.e., the owner. Like school reports, drivers’ licenses can be shown over and over again during their validity period, and neither their content nor owner can be changed.

Example (iv) (ELECTRONIC COINS) Customers can *withdraw* electronic coins from their bank accounts on certain conditions (account not overdrawn, good credit rating, etc.) and they can freely *pay* their electronic coins to other individuals. A payee accepts electronic coins if the coins are valid and the respective bank of issue holds a proper licence. By a successful payment, the owner of the electronic coin, if we regard it as a basic legitimation, changes from the payer to the payee. Payees can later *deposit* the received coin at the bank of issue (or any of its agents). After a successful deposit, the owner changes again from the intermediate payee to the bank finally receiving the coin. The typical stages of using electronic coins are depicted in Figure 2–2 on p.9. In some electronic payment schemes, coins can be paid several times from individual payer to individual payee before they are deposited back into a bank. This is indicated by the dotted part of Figure 2–2 on p.9.

2) This example is most realistic if one thinks of “virtual” schools or colleges providing correspondence courses.

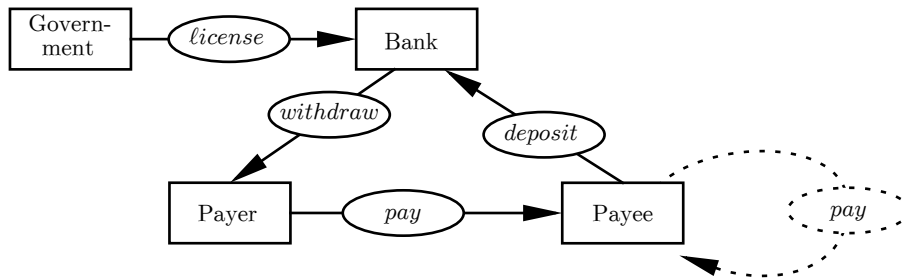


FIGURE 2-2 Electronic Coins

Example (v) (PHONECARDS) Prepaid phonecards can be regarded as multi-show electronic coins, each show representing a partial payment. They are consumed after a fixed number of payments. If the bound is $k \in \mathbb{N}$, we call these coins *k-show*.

Example (vi) (BONDS, STOCKS) Customers can *buy* or *sell securities, mortgage bonds, bonds* or *stocks* by their banks or brokers. Regarded as basic legitimations, these examples are similar to electronic coins because their owner changes upon successful buying or selling, respectively. Unlike electronic coins however, they can be sold by their owner only (Figure 2-3 on p.9).

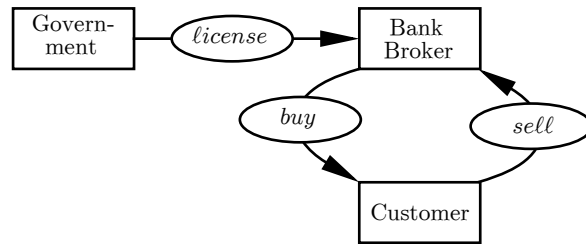


FIGURE 2-3 Bonds, Stocks

EXAMPLE (VII) (AIRFARE TICKETS) Customers can get *airfare tickets* issued, but only the owner of an airfare ticket can use it and after it has been used, it is cancelled. In terms of a basic legitimation, the owner of an airfare ticket is de-assigned at the moment he passes the gate at the departure terminal.

2.4 Categorization of Basic Legitimation Systems

Motivated by the examples of the previous section, we now derive four categories of basic legitimation systems according to their sort of holder authorization. We first give a short description of each category and then characterize them in terms of the simple model introduced in Section 2.1 on p.6.

Free Certificates represent basic legitimations that can be shown by their holders as often as they like, and their content never changes. See Section 2.3 on p.7, Example (i) ((CERTIFIED DOCUMENT)).

Personal Certificates represent basic legitimations whose content and owner cannot be changed. Personal certificates can be shown only by their owner. See Section 2.3 on p.7, Example (ii) ((SCHOOL REPORTS)) and Example (iii) ((DRIVERS'S LICENSES)).

Coin Certificates represent basic legitimations whose owner changes when any of its instances is transferred. The previous owner (payer) is then replaced by the recipient (payee). See Section 2.3 on p.7, Example (iv) ((ELECTRONIC COINS)) and Example (v) ((PHONECARDS)).

Bond Certificates represent basic legitimations that can be transferred by their owner only and where the owner changes when any of its instances is transferred. The previous owner is then replaced by the recipient. See Section 2.3 on p.7, Example (vi) ((BONDS, STOCKS)) and Example (vii) ((AIR-FARE TICKETS)).

Next we ask for each basic legitimation system how its holder authorizations can be enforced. A straightforward implementation is to keep and manage the legitimations of all participants in a central database system similar to an account management system. This is in fact what governments, corporations, insurers, and other businesses do today. One of the major problems with centralized implementations is that customers and clients can never be certain who has or can obtain what personal information about them. Probably the only practical way to alleviate this problem is to make sure that the numerous institutions that keep and manage legitimations have a sound privacy policy in place and are monitored by independent organizations to make sure they adhere to these policies. In steadily globalized businesses around the world, however, the above privacy problem tends to overwhelm any sound privacy policies and privacy monitoring organizations. The driving forces are mergers and acquisitions of companies across business sectors (banks and insurers, telecoms and media industries), deregulation and privatization of former government institutions (telecoms and postal authorities) and ever more refined tracking methods such as genetic fingerprints and biometric data. Personal information about customers is becoming an ever more valuable resource in itself.

In the following, we are interested in decentralized implementations of legitimation systems, which have the potential of really solving the privacy problem because they give customers back the control over their personal legitimations and in particular to use them in a privacy oriented way. We will shortly investigate for each type of basic legitimation, which problems must be solved by any decentralized implementation managing this type of basic legitimation.

Free legitimation systems require no holder authorization at all and can be implemented by digital certificates that can be shown by any holder.

For personal legitimation systems there need to be measures in place enforcing that they are indeed shown by their owner. In case of human owners and holders, the holder authorization problem can be solved only by biometric recognition means.

For coin legitimation systems there need to be measures in place that enforce the change of ownership after being shown a specified number of times. There is a pro-active and a retro-active approach to solve this problem. The pro-active approach is to prevent holders from successfully showing coin certificates that they have already shown before. The retro-active approach is to automatically reveal the identity of holders who show coin certificates that they have already shown before. The difficulty of enforcing change of ownership is that either holders must be prevented from proliferating copies of

their coin certificates in an uncontrolled fashion, or there must be a powerful verification infrastructure that can monitor the showing of each single copy of coin certificates in a timely manner.

For bond legitimation systems the holder authorization problems of both personal and coin legitimation systems have to be solved.

2.5 More Complex Legitimation Systems

This section gives a short outlook on more complex legitimation systems. Since they are not analyzed further we only present them as illustrating examples.

Holders of personal, coin or bond certificates sometimes need to make copies of their certificates in order to enable delegates to show these certificates to third parties in behalf of their holders. For example, a graduate student Alice can make a copy of her degree certificate and give it to a friend Bob so he may prove to someone else that Alice owns the degree, while Alice herself stays home sick. This works only if the verifier does not require Alice to show up and verify her biometric identity because he knows her. Likewise, Alice may wish to make a copy that only Bob can use to prove her degree. Such requirements raise the issue of derived certificates. For example, from a personal certificate one can derive a free certificate whose content is the complete original personal certificate including its owner and holder.

In general, any action of a legitimation system can be guarded by legitimations of another content type. For instance, in Example (ii) on p.8 only accredited schools may issue school reports. In Example (iv) ((ELECTRONIC COINS)), only banks of issue (and their agents) are legitimated to mint cash or cheques. The guarding legitimations are usually regarded as superior legitimations and act as *licenses*. In general, this leads to a *hierarchy of legitimation types*, where legitimations of a parent type guard actions on legitimations of the child types. In practice, these type hierarchies have only a few levels, e.g., school reports, school accreditations, federal or state law.

If a content type represents disadvantages for the respective owners of certificates, and if therefore owners have a natural interest in holding them back, the certificates are called *negative* [65]. In paper-based implementations, negative certificates are usually not stored and managed by their owners alone. For instance, the prosecution authorities create and run criminal records of individuals. For negative certificates, correctness, unforgeability and privacy requirements carry over from above, but integrity requirements are imposed by the “other side”. For example, it is the providers who require certificates being consumed after payment if the certificates represent credit. But if they represent debit it is the owners who require them being consumed after being “paid”. This switch of interest has severe implications on how to enforce the integrity requirements. See for example [65].

2 A GUIDED TOUR TO LEGITIMATIONS

3

Notations and Assumptions

The general notation is summarized in Section 3.1 on p.13, protocols and operations are introduced in Section 3.2 on p.15 and Section 3.3 on p.22. In Section 3.4 on p.27 and 3.7 the complexity theoretic and other assumptions relevant to this work are stated. Commonly used symbols are listed and explained in Index A.

3.1 General Notation

3.1.1 Probabilities

Let S be a *sample space* of *simple, alternative events*, and call any subset $E \subseteq S$ an *event* of S . We will only consider discrete sample spaces¹. A *probability distribution* $Prob[]$ on a sample space S is a mapping from events of S to real numbers such that the following probability axioms are satisfied:

- 1) For each event $E \subseteq S$: $0 \leq Prob[E]$.
- 2) $Prob[S] = 1$.
- 3) For two disjunct events $E, F \subseteq S$: $Prob[E \cup F] = Prob[E] + Prob[F]$.

The set of samples with positive probability is denoted $[P]$. If S is finite and every simple event $e \in S$ is assigned the same probability $Prob[e] = 1/|S|$, then we have the uniform probability distribution on S . The probability of an event $E \subseteq S$ is $|E|/|S|$, where $|E|$ and $|S|$ denote the number of simple events contained in E and S .

Taking a simple, alternative event from S and assigning it to a variable x is denoted by $x \leftarrow S$. If an event is taken from S according to the uniform distribution and assigned to x , then we say that it is chosen from S “uniformly at random” or simply “at random” and we denote this by $x \in_R S$. A (discrete) *random variable* X is any function from a sample space S to the set $\{0, 1\}^*$ of binary strings².

1) Sample spaces are usually subsets of the reals. However, as usual in complexity theory, the following sample spaces are all subsets of the binary strings $\{0, 1\}^*$.

2) Only discrete random variables are considered that range over subsets of binary strings.

3 NOTATIONS AND ASSUMPTIONS

For a random variable X and a string $x \in \{0, 1\}^*$, we define the event $X = x$ to be the set $\{e \in S \mid X(e) = x\}$; thus

$$\text{Prob}[X = x] = \sum_{\{e \in S \mid X(e) = x\}} \text{Prob}[e] . \quad (3.1)$$

In the following work, important examples of random variables are the outputs of probabilistic algorithms and protocols. Therefore, the elementary notation (3.1) will be suitably extended in Section 3.2.1 on p.16.

3.1.2 Algebraics

A *family* of objects is a function from an *index set* I to some object set Z . It is denoted $\{z_i\}_{i \in I}$ indicating that index $i \in I$ is mapped to element $z_i \in Z$. A family of random variables is called an *ensemble*.

If $f()$ defines a function from set A to set B , it can be naturally extended to a set function $F()$ from any subset $\alpha \subseteq A$ to subsets of B as follows

$$F(\alpha) = \bigcup_{a \in \alpha} \{f(a)\} .$$

For convenience, we will also write $f(\alpha)$ instead of $F(\alpha)$ because the two different uses of f are sufficiently distinguished by its argument.

If $f()$ is an injective function from A to B , then $f^{-1}(b)$ denotes the function that assigns to each element $b \in \text{Im}(f)$ its pre-image. If $f()$ is not injective, then $f^{-1}(b)$ denotes the set-valued function that assigns to each element $b \in \text{Im}(f)$ its set of pre-images.

The length of a bitstring is denoted $|\bullet|_2$.³ The binary length of a number $k \in \mathbb{N}$ is the length of the binary representation of that number, denoted as $|k|_2$.

An element of a vectorspace V of finite dimension $m \in \mathbb{N}$ over a field F is usually written \vec{v} , and the i -th component of \vec{v} ($i \in [1, m]$) is denoted v_i .

If we rewrite expressions or equations in several steps, we usually display each step in a separate line. To help the reader understand the rewriting from line (a) to the next line (b), we give hints in a smaller font on the right hand side of line (b).

3.1.3 Asymptotics

The phrase “for all k sufficiently large” means $\exists k_0 \forall k > k_0$. Consider three functions $\nu(k)$, $\mu(k)$ and $\phi(k)$ whose image is included in the real interval $[0, 1]$.

- $\nu(k)$ is said to vanish faster than every polynomial iff for all $c > 0$ and sufficiently large inputs k : $\nu(k) < k^{-c}$. Sequences behaving like $\nu(k)$ are said to be *negligible*.
- $\mu(k)$ is said to approach 1 faster than every polynomial iff for all $c > 0$ and sufficiently large inputs k : $\mu(k) > 1 - k^{-c}$. Sequences behaving like $\mu(k)$ are called *overwhelming*.
- $\phi(k)$ is said to vanish slower than some polynomial iff there is a $c > 0$ such that for sufficiently large inputs k : $\phi(k) > k^{-c}$. Sequences behaving like $\phi(k)$ are called *non-negligible*.

3) The index distinguishes the symbol $|\bullet|_2$ from the symbol $|\bullet|$, which returns the absolute value of its argument.

Note, that non-negligible is NOT the negation of negligible, but rather a very strong negation of it. There are sequences in between, which are neither negligible nor non-negligible. Here is an example:

$$\varphi(k) = \begin{cases} k^{-2} & \text{for } k \text{ odd} \\ 2^{-k} & \text{for } k \text{ even} \end{cases}$$

A function $f(n)$ is said to be big- O of $g(n)$, denoted $f(n) = O(g(n))$, if there is a constant $c > 0$ such that for all sufficiently large $n \in \mathbb{N}$: $f(n) \leq c \cdot g(n)$.

3.2 Protocols, Semantics, Interfaces and Operations

For protocols, we use the computational model of interactive Turing machines (ITM) introduced by Goldwasser, Micali, and Rackoff [123].

Definition 3.1 Interactive Turing Machine (ITM)

An *interactive Turing machine* (ITM) is a *probabilistic* Turing machine T equipped with a read-only *input tape* I , a *work tape* W , a *random tape*, the same number $\nu \in \mathbb{N}_0$ of read-only *communication tapes* C_j^{\leftarrow} ($j \in [1, \nu]$) and *write-only communication tapes* C_j^{\rightarrow} , and a write-only *output tape* O . In order to model additional access to external information, ITMs can optionally be equipped with a read-write *composition tape* Q . This tape serves to provide input to an ITM that is inconvenient to feed through its input tape. It may model a-priori knowledge that an ITM has prior to a computation. Or, if ITMs are sequentially composed such that one takes some or all of the output of the other as input, the composition tape is used to model persistent memory accessible by both machines.

Before an interactive Turing machine is executed, the input tape contains the strings to be taken as input. The communication tapes and output tapes are empty. The random tape contains an infinite sequence of random bits, and can be scanned only from left to right. We say that an interactive Turing machine *flips a coin* if it reads the next bit from its random tape. For specific ITMs, it is more convenient to talk about choosing elements uniformly at random from a specified set R than just about flipping coins. We say that an interactive Turing machine *chooses* an element ρ from R if it repeatedly flips k coins until it finds some $\rho \in R$. We call this procedure the *pick-and-test algorithm* (PAT). The elements ρ are called *internal choices*. Throughout this work, the bits on the random tape of an interactive Turing machine are presumed uniformly distributed. The way how ITMs choose elements from a domain R ensures that the chosen elements are uniformly distributed, too.

For all but the random tape, we re-use the tape identifiers also for the variables taking the respective strings as values, and we call the values of I , C_i^{\leftarrow} , C_i^{\rightarrow} and O , and Q : *inputs*, *read-messages*, *write-messages*, *outputs*, *compositional inputs* and *compositional outputs* of T , respectively.

The *computation time* of an ITM is the number of its computation steps. An ITM is called *polynomial-time* if its computation time is bounded above by a polynomial in the length of its input. An ITM is called *efficient* if its computation time is bounded above by a polynomial of degree ≤ 3 in the length of the input to the ITM. Unless otherwise mentioned, ITMs are probabilistic polynomial-time (PPT). If its computation time is not limited, the ITM is called *computationally unlimited*. \blacklozenge

Remark: Interactive Turing machines exchange data in untyped form, which is clear from the fact that the tapes of an interactive Turing machine do not provide any information about how messages are to be interpreted. The interpretation of messages is totally up to the ITMs reading the messages.

Definition 3.2 *n*-Party Protocol

An *n*-party protocol $pr = (T_1, \dots, T_n)$ is an *n*-tuple of ITMs T_1, \dots, T_n such that:

- each pair of ITMs either share a pair of communication tapes (one in each direction) or none. If T_i and T_j ($i, j \in [1, n]$) share a pair of communication tapes, they are said to be *directly connected* and in this case the read messages of T_i are the write-messages of T_j and vice versa: $C_i^{\leftarrow} = C_j^{\rightarrow}$ and $C_i^{\rightarrow} = C_j^{\leftarrow}$. In this case we use only the symbols C_j^{\rightarrow} and C_i^{\leftarrow} for write-only communication tapes.
- the graph $TP_{pr} = (\{T_1, \dots, T_n\}, V)$ is connected, where $(T_i, T_j) \in V$ is an edge iff T_i and T_j are directly connected. It is called the *communication topology* of pr .

Each ITM T_i ($i \in [1, n]$) of a protocol is called a *party* or *participant*. A protocol is called an *algorithm* if it consists of only one participant, i.e., $n = 1$, otherwise it is called an *interactive protocol*. A protocol (T_0, T_1, \dots, T_n) is *starlike* with center T_0 if its communication topology is a star with center T_0 : $V = \{(0, i) \mid i \in [1, n]\}$.

The time model underlying interactive protocols in this work is synchronous linear time. Interactive protocols can therefore be regarded as a sequence of consecutive *rounds*. Before the first round, each machine reads its respective input tape. In each round $i = 1, 2, \dots$ of the protocol, each machine (i) reads a read-message from each of its read-only communication tapes and its optional composition tape, (ii) performs some *steps* (i.e., local computations) using its work tape and random tape and (iii) writes a write-message or special abort symbol to each of its write-only communication tapes and optional strings to its output tape and composition tape. In the following round, each participant's read-only communication tape contains the write-messages provided by other ITMs in the preceding round. Whenever a participant reads an abort symbol on any of its read-only communication tapes, it writes an abort symbol to all of its write-only communication tapes and then terminates.

The *computation time* of a protocol pr on input I_1, \dots, I_n of respective participants T_1, \dots, T_n is the sum of the computation times of all its participants. (A notation for the computation time of a protocol will be introduced in Definition 3.10 on p.20.) A protocol is *polynomial-time* (sometimes called *computational*) if its computation time is bounded above by a polynomial in the length of the protocol's input, i.e., the sum of the lengths of the inputs of all participants. A protocol is called *efficient* if its computation time is bounded above by a polynomial of degree ≤ 3 in the length of the protocol's input. A protocol is called *computationally unlimited* if there is no limit on its computation time. ◆

3.2.1 Execution Functions

The behavior of an interactive Turing machine, i.e., its write-messages and outputs, is completely determined by its inputs, random coins and read-messages. This leads to the definition of execution functions as a probabilistic description of ITMs (see Beaver [9,10,11]).

Definition 3.3 ITM Execution Function

Consider an ITM T and the sample space of all internal choices it can make. We call the simple events of this sample space the (possible) executions of T .

Formally, the *probabilistic execution function* of the ITM T is an ensemble, namely the family $T_{I, C^{\leftarrow}, \{Q\}}$ of random variables⁴ indexed by the tuple $(I, C^{\leftarrow}, \{Q\})$ of an input I , a sequence of read-messages C^{\leftarrow} and an optional message Q from the composition tape of T . Each random variable takes an input from the sample space of internal choices of T and returns a tuple $(O, C^{\rightarrow}, \{Q'\})$ of outputs, write-messages and an optional message written back to the composition tape of T . For each particular input tuple $(I, C^{\leftarrow}, \{Q\})$, the probability assigned to each returned tuple $(O, C^{\rightarrow}, \{Q'\})$ is determined by the ITM T . Note that the probability distribution of the random choices made by T , i.e. the bits written on the random tape, is presumed uniform. Taking a sample according to a random variable $T_{I, C^{\leftarrow}, \{Q\}}$ and assigning it to the output parameter tuple $(O, C^{\rightarrow}, \{Q'\})$ is denoted:

$$(O, C^{\rightarrow}, \{Q'\}) \leftarrow T(I, C^{\leftarrow}, \{Q\}) . \quad (3.2)$$

To keep the notation intuitive, we name the probabilistic execution function after the ITM (T) that defines the probability distribution of the return values. And we write the index tuple $(I, C^{\leftarrow}, \{Q\})$ of the family of random variables like an input to the ensemble.

The *deterministic execution function* of T is defined as follows: On input I , some read-messages C^{\leftarrow} and internal choices ρ , the output O , write-messages C^{\rightarrow} and optional messages to the composition tape are returned. This function is denoted

$$(O, C^{\rightarrow}, \{Q'\}) \leftarrow_{\rho} T(I, C^{\leftarrow}, \{Q\}) . \quad \blacklozenge \quad (3.3)$$

Since all read-messages of participants of a protocol are the write-messages of other participants, the behavior of a protocol, i.e., all write-messages and output messages are determined by the input messages and random coins of all participating ITMs. A particular behavior of a protocol is sometimes called an *execution*, *run* or *instance* of the protocol. An execution is said to be *aborted* if any machine writes an abort symbol during the execution, otherwise it is said to be *successful*. This leads to the definition of the execution function of a protocol.

Definition 3.4 Protocol Execution Function

The *probabilistic execution function* of a protocol $pr = (T_1, \dots, T_n)$, $n \in \mathbb{N}$ is an ensemble, namely the family of the following random variables indexed by the tuple (I_1, \dots, I_n) of inputs I_j and optional composition messages Q_j of all participants T_j ($j \in 1, \dots, n$) of pr . Each random variable takes an input from the sample space of n -tuples of internal choices of each participant and returns the n -tuples (O_1, \dots, O_n) , (ρ_1, \dots, ρ_n) , $(C_1^{\leftarrow}, \dots, C_n^{\leftarrow})$, $(C_1^{\rightarrow}, \dots, C_n^{\rightarrow})$, (Q_1', \dots, Q_n') , of outputs, internal choices, read-messages and write-messages and optional composition messages of all participants. The probability distribution is over the sample space of n -tuples of internal choices of all participants. Taking a sample according to this distribution, is denoted

4) Curly brackets indicate optional parameters.

$$\binom{T_1}{(O_1, \rho_1, C_1^{\leftarrow}, C_1^{\rightarrow}, \{Q_1'\})}, \dots, \binom{T_n}{(O_n, \rho_n, C_n^{\leftarrow}, C_n^{\rightarrow}, \{Q_n'\})} \leftarrow pr(\binom{T_1}{I_1, \{Q_1\}}, \dots, \binom{T_n}{I_n, \{Q_n\}}) \quad (3.4)$$

The square brackets indicate to which participant a parameter belongs.

The probabilistic execution function of a participant T_i is the projection of the probabilistic execution function of pr (where the n connected ITMs define one single ITM) to T_i 's outputs, internal choices, read-messages and write-messages. More precisely, the probabilistic execution function of a participant T_i is the ensemble of random variables (of the probabilistic execution function of pr) each projected to the sample space times the output domain of respective participant T_i .

The *deterministic execution function* of pr is defined as follows: For each $i \in [1, n]$, input I_i , read-messages C_i^{\leftarrow} and internal choices ρ_i , it returns the output O_i , internal choices ρ_i , read-messages C_i^{\leftarrow} and write-messages C_i^{\rightarrow} :

$$\binom{T_1}{(O_1, \rho_1, C_1^{\leftarrow}, C_1^{\rightarrow}, \{Q_1'\})}, \dots, \binom{T_n}{(O_n, \rho_n, C_n^{\leftarrow}, C_n^{\rightarrow}, \{Q_n'\})} \leftarrow pr(\binom{T_1}{\rho_1, I_1, \{Q_1\}}, \dots, \binom{T_n}{\rho_n, I_n, \{Q_n\}}) \quad (3.5) \quad \blacklozenge$$

Definition 3.5 Equivalence of ITMs

Two ITMs T_1 and T_2 are *equivalent* iff their probabilistic execution functions are equal. \blacklozenge

Definition 3.6 General Composition of ITMs

A *general composition* of $n \in \mathbb{N}$ ITMs, denoted $T_1 ||| \dots ||| T_n$, is any set of n ITMs T_1, \dots, T_n together with an effective description whether each two ITMs T_i, T_j for $1 \leq i \neq j \leq n$ share:

- (a) their input tapes,
- (b) their output tapes,
- (c) T_i 's output tape with T_j 's input tape,
- (d) one of T_i 's read-only communication tapes with one of T_j 's write-only communication tapes,
- (e) their (optional) composition tapes, or
- (f) no tapes.

The ITMs T_1, \dots, T_n are called *member ITMs* of the composition $T_1 ||| \dots ||| T_n$. \blacklozenge

Observation 3.7

We can think of a general composition $T_1 ||| \dots ||| T_n$ as a schedule of computation that can be represented by one interactive Turing machine, but in general such an ITM is NOT uniquely determined. In principle, a general composition $T_1 ||| \dots ||| T_n$ defines a partial order ' $<$ ' of all elementary steps of all its member ITMs as follows. The *direct predecessors* of an elementary step s of T_i ($1 \leq i \leq n$) are all elementary steps that are either a direct predecessor of s in T_i or an elementary step of some other member ITM T_j ($j \neq i$) that writes a message to a shared tape that is read by T_i in step s . A step r is a predecessor of step s ($r < s$) iff it r is in the transitive closure of the direct predecessors of s . We say that an ITM T *represents* a given general composition $T_1 ||| \dots ||| T_n$ iff the total order of elementary steps of T can be embedded into the partial order ' $<$ ' of all elementary steps of $T_1 ||| \dots ||| T_n$ and T reads all its inputs from one input tape, writes all outputs to one output tape, and uses only one work tape, one random tape and one (optional) composition tape. In other words, any two ITMs representing a general composition $T_1 ||| \dots ||| T_n$ can differ only in the order of elementary steps that are independent of each other.

All ITMs representing a general composition $T_1 \parallel \dots \parallel T_n$ have the same probabilistic execution function, because they all produce their messages, which they write to their tapes, according to the same probability distribution. This probabilistic execution function is uniquely defined by $T_1 \parallel \dots \parallel T_n$. By writing $T_1 \parallel \dots \parallel T_n$ we mean the composition of ITMs or just any ITM representing it. Which meaning is intended will always be clear from the context.

3.2.2 Internal and External Interfaces

It is useful now to distinguish the input and output behavior of a protocol from its communication behavior. We distinguish external interfaces, by which different protocols interoperate; and internal interfaces, by which the participants of the same protocol interoperate. The definitions are given in this and the following subsection.

Definition 3.8 Output Functions and External Interface

The *probabilistic output function* of pr is the projection of the probabilistic execution function of pr to the product of the domain of its inputs times the domain of n -tuples of outputs of each participant. Taking a sample according to this distribution is denoted

$$({}^{T_1}[O_1, \{Q_1'\}], \dots, {}^{T_n}[O_n, \{Q_n'\}]) \leftarrow \text{out}(pr({}^{T_1}[I_1, \{Q_1\}], \dots, {}^{T_n}[I_n, \{Q_n\}])), \quad (3.6)$$

where I_i (O_i) denotes the input (output) of T_i . The distribution of these random variables *out* is defined naturally as follows:

$$\begin{aligned} & \text{Prob}[({}^{T_1}[O_1, \{Q_1'\}], \dots, {}^{T_n}[O_n, \{Q_n'\}]) \leftarrow \text{out}(pr({}^{T_1}[I_1, \{Q_1\}], \dots, {}^{T_n}[I_n, \{Q_n\}]))] \\ &= \text{Prob}[\forall i = 1 \dots n : (O_i = \hat{O}_i) \wedge \{Q_i' = \hat{Q}_i'\} \setminus \\ & \quad ({}^{T_1}[\hat{O}_1, \rho_1, C_1^{\leftarrow}, C_1^{\rightarrow}, \{\hat{Q}_1'\}], \dots, {}^{T_n}[\hat{O}_n, \rho_n, C_n^{\leftarrow}, C_n^{\rightarrow}, \{\hat{Q}_n'\}]) \leftarrow pr({}^{T_1}[I_1, \{Q_1\}], \dots, {}^{T_n}[I_n, \{Q_n\}])] \end{aligned} \quad (3.7)$$

The projection of the output function to a particular participant T_j is denoted as

$${}^{T_j}[O_j] \leftarrow \text{out}_{T_j}(pr({}^{T_1}[I_1, \{Q_1\}], \dots, {}^{T_n}[I_n, \{Q_n\}])). \quad (3.8)$$

The *external interface* EF_{pr} of pr is the list, as denoted in (3.6), of the input and output parameters of each participant of pr . The *external interface of a participant* T_j of pr is the list of the input and output parameters of participant T_j .

If we mean the output function and it is clear from the context that we do not mean the execution function, then—in order to enhance readability—we omit the identifier “out”.

If pr_1, \dots, pr_m are m algorithms with matching external interfaces, and p is a predicate of arity $3m + 1$, then the following expression denotes the probability that $p(I_0, \{Q_0\}, O_1, \{Q_1\}, \dots, I_{m-1}, O_m, \{Q_m\})$ holds after the result of running pr_j on input $I_{j-1}, \{Q_{j-1}\}$ has been assigned to $(O_j, \{Q_j\})$, for $j = 1, 2, \dots, m$ in this order:

$$\begin{aligned} & \text{Prob}[p(I_0, \{Q_0\}, O_1, \{Q_1\}, \dots, I_{m-1}, O_m, \{Q_m\}) \setminus (O_1, \{Q_1\}) \leftarrow pr_1(I_0, \{Q_0\}); \\ & \quad (O_2, \{Q_2\}) \leftarrow pr_2(I_1, \{Q_1\}); \\ & \quad \dots; (O_m, \{Q_m\}) \leftarrow pr_m(I_{m-1}, \{Q_{m-1}\})] \quad \blacklozenge \end{aligned}$$

Definition 3.9 Equivalence of Protocols ◆

Two protocols pr_1 and pr_2 are *equivalent* iff their probabilistic execution functions are equal. ◆

Remark: Note that two protocols can only be equivalent if their external interfaces have exactly the same input and output parameters. A *protocol instance* is a protocol where all formal input parameters are assigned actual values. In other words, a protocol instance is a protocol with no formal input parameters. Thus, Definition 3.9 on p.20 applies also to protocol instances.

Definition 3.10 Acceptors, Expected Computation Time, Shared Input/Output

If an ITM is intended to accept or reject executions of a protocol (*acceptor*), then we use the special symbols *TRUE* and *FALSE* for its return values. If an ITM aborts the execution of a protocol, we use the special symbol \perp (*bottom*) as its return value.

The *computation-time* of a protocol pr (see Definition 3.2 on p.16) whose participants T_1, \dots, T_n take respective inputs I_1, \dots, I_n , make internal choices ρ_1, \dots, ρ_n and deliver outputs O_1, \dots, O_n that satisfy a given n -ary predicate $p(\cdot)$ is denoted as follows:

$$Time(p(O_1, \dots, O_n) \setminus ({}^{T_1}[O_1], \dots, {}^{T_n}[O_n]) \leftarrow pr({}_{\rho_1}^{T_1}[I_1], \dots, {}_{\rho_n}^{T_n}[I_n])) . \quad (3.9)$$

The *expected computation-time* of pr whose participants T_1, \dots, T_n take respective inputs I_1, \dots, I_n and deliver outputs O_1, \dots, O_n that satisfy a given n -ary predicate p on inputs O_1, \dots, O_n is defined as follows:

$$\begin{aligned} ExpTime[p(O_1, \dots, O_n) \setminus ({}^{T_1}[O_1], \dots, {}^{T_n}[O_n]) \leftarrow pr({}^{T_1}[I_1], \dots, {}^{T_n}[I_n])] \\ = \sum_{\rho_1, \dots, \rho_n} Time(p(O_1, \dots, O_n) \setminus ({}^{T_1}[O_1], \dots, {}^{T_n}[O_n]) \leftarrow pr({}_{\rho_1}^{T_1}[I_1], \dots, {}_{\rho_n}^{T_n}[I_n])) Prob[\rho_1, \dots, \rho_n] . \end{aligned}$$

Note that this expression is well defined also if an internal choice ρ_i ranges over an infinite domain. This occurs, for example, in protocols that make a random choice within a loop that terminates upon a condition depending on the outcome of the internal choice. (For more details see the remark following Definition 3.15 on p.23.)

Next consider a protocol with two participants S, T which both share an input (output) parameter a (b). Then we call a (b) a *shared input (output) parameter* and its values *shared inputs (outputs)*, respectively. Parameters that are not shared are called private. If all participants of a protocol share an input (output) parameter, then this is called a *common input (output) parameter*. It is convenient to denote the common parameters only once in the external interface. We allow to write common parameter(s) in additional unlabeled square brackets. For example, consider a protocol

$$({}^S[y, c], {}^T[y, d]) \leftarrow pr({}^S[x, a], {}^T[x, b]) , \quad (3.10)$$

where participants S and T take a common input x and respective private inputs a and b and return a common output y and respective private outputs c and d . Often we use the equivalent and more convenient form:

$$([y], {}^S[c], {}^T[d]) \leftarrow pr([x], {}^S[a], {}^T[b]) . \quad \blacklozenge$$

Definition 3.11 View Functions and Internal Interface

The *probabilistic view function* of pr is defined as the projection of the probabilistic execution function to the internal choices, read-messages and write-messages. Taking a sample of pr on inputs I_i and outputs O_i according to this distribution is denoted as follows:

$$(\overset{T_1}{\rho}_1, C_1^{\leftarrow}, C_1^{\rightarrow}), \dots, (\overset{T_n}{\rho}_n, C_n^{\leftarrow}, C_n^{\rightarrow}) \leftarrow \text{view}(pr(\overset{T_1}{I}_1], \dots, \overset{T_n}{I}_n])) \quad (3.11)$$

where ρ_i , C_i^{\leftarrow} and C_i^{\rightarrow} denote the internal choice, read-messages and write-messages of T_i . The *internal interface* IF_{pr} of pr is the list of input and output parameters of all participants of the probabilistic view function of pr as denoted in (3.11). The *internal interface* of participant T_i is the list of input and output parameters of participant T_i of the probabilistic view function of pr .

Let protocol pr have participants S, T_1, \dots, T_n such that each T_i is directly connected to S . Let C_i^{\leftarrow} denote T_i 's read-messages from S , and C_i^{\rightarrow} denote T_i 's write-messages to S . Then the *selective view function* of $T = (T_1, \dots, T_n)$ on S

$$(\overset{T_1}{\rho}_1, C_1^{\leftarrow}, C_1^{\rightarrow}), \dots, (\overset{T_n}{\rho}_n, C_n^{\leftarrow}, C_n^{\rightarrow}) \leftarrow \text{view}_S^{(T_1, \dots, T_n)}(pr(\overset{S}{I}_0], \overset{T_1}{I}_1], \dots, \overset{T_n}{I}_n])) \quad (3.12)$$

is defined as the distribution of the internal choices of T_1, \dots, T_n and their read-messages from S and write-messages to S . The probabilities are over the random coins of all participants. Each sample taken is called a *view* of T on S . A view of T on S is *valid*, if neither participant aborts the execution. The deterministic view function and deterministic selective view function are defined as the respective projections of the deterministic execution function. \blacklozenge

3.2.3 Protocol Composition and Decomposition

Next, we define compositions of protocols. Compared to the definition given by Feige and Shamir [107], our definition is more general in the sense that the composed protocols may have more than 2 participants, and it is more special in the sense that the number of composed protocols is independent of the input to the composition.

Definition 3.12 Composition of Protocols

A *general composition* of the protocols pr_1, \dots, pr_ℓ is any protocol pr defined by an effective description about:

- (a) which value each input parameter of the protocols pr_1, \dots, pr_ℓ takes either from an input parameter of pr or from an output parameter of another protocol of pr_1, \dots, pr_ℓ as long as no cyclic dependencies arise,
- (b) which participants of pr_1, \dots, pr_ℓ share their composition tapes, and
- (c) which value each output parameter of pr takes either from an input parameter of pr or from an output parameter of one of the protocols pr_1, \dots, pr_ℓ .

The protocols pr_1, \dots, pr_ℓ are called *member protocols*, of pr .

A general composition is *sequential* if the member protocols are executed one after another, i.e. each but the first member protocol starts execution only after the previous member protocol has stopped its

execution and pr_{i+1} ($i \in [1, \ell - 1]$) may take as input values only inputs or outputs of the protocols pr_1, \dots, pr_i . \blacklozenge

Observation 3.13 Polynomial Composition

A general composition pr of pr_1, \dots, pr_ℓ is *polynomial-time*, if all member protocols are polynomial-time (Definition 3.2 on p.16). This basically follows from the fact that if k is the security parameter of the scheme containing pr_1, \dots, pr_ℓ , $P(k)$ represents the number of protocols in the composition pr , and $Q(k)$ represents the number of steps of the most complex of protocols pr_1, \dots, pr_ℓ , then the number of steps of the composition pr is bounded above by $P(Q(k))$, which is itself polynomial in k .

The opposite process of protocol composition is protocol decomposition. We only consider the special case of decomposition at participant level. For example, we want to write an n -party protocol pr as an $(n - 1)$ party protocol by regarding two or more of its participants as one ‘composed participant’.

Definition 3.14 Protocol Refinement and Composed Participants

Let pr be a protocol of $n \in \mathbb{N}$ participants T_1, \dots, T_n (with unique parameter identifiers over all square brackets) and let $\{S_1^*, \dots, S_m^*\}$ be a partition of $\{T_1, \dots, T_n\}$ into m (pairwise disjoint) subsets such that

$$S_i^* = \{T_{i_1}, \dots, T_{i_{n_i}}\} \text{ for all } i = 1, \dots, m.$$

Note that n_i denotes the number of participants of subset S_i^* and therefore $\sum_{i=1}^m n_i = n$. Furthermore let $T_{i_1} ||| \dots ||| T_{i_{n_i}}$ be the composition of the interactive Turing machines $T_{i_1}, \dots, T_{i_{n_i}}$ that are connected as defined by pr . More precisely, T_{i_j} and T_{i_k} share an input tape (output tape, mutual communication tapes) iff they do so in pr . Let S_i be an ITM representing the composition $T_{i_1} ||| \dots ||| T_{i_{n_i}}$ (observation Observation 3.7 on p.18). We denote by pr' the m -party protocol consisting of the participants S_1, \dots, S_m and call it a *sub-protocol* of the original protocol pr .

A participant S_i of pr' is also called a participant of pr if $n_i = 1$ and a *composed participant* of pr if $n_i > 1$. We indicate composed participants by *nested square brackets*. For example, consider a 3-party protocol $pr(\overset{A}{[a]}, \overset{B}{[b]}, \overset{C}{[c]})$. If we need to refer to the sub-protocol of pr between the composed participant $A ||| B$ and C , then we write

$$pr(\overset{A}{[[a], [b]]}, \overset{C}{[c]}).$$

\blacklozenge

3.3 Cryptographic Schemes and Mechanisms

Cryptographic solutions for problems related to legitimation systems require more than one protocol. Thus we need a framework in which we can consider a collection of protocols which together satisfy certain requirements. Furthermore, we want to formalize these requirements within the framework, in particular specify categories of basic legitimation systems according to Section 2.4 on p.9. In order to formalize complexity theoretic (sometimes called computational) requirements, the envisioned framework must organize the various protocols, their participants and parameter domains all under one

security parameter k . The role of k is roughly to uniformly determine lower bounds on the amount of computation time certain attackers need to break the respective complexity theoretic requirements.

In this section, we introduce *cryptographic schemes* as a generic framework to specify cryptographic solutions involving several protocols. We call any instance of a cryptographic scheme a *cryptographic mechanism* or construction, i.e., a cryptographic solution that satisfies all the requirements specified by the cryptographic scheme.

An important element of a cryptographic scheme is its *prekey*. It defines how messages are to be interpreted, but only to such an extent necessary to formulate all requirements of the cryptographic scheme. For example, a prekey of a cryptographic scheme can define a family f of member function identifiers, and a security requirement can formulate that f is one-way. The cryptographic scheme will not specify the domains or images of the member functions of f , because one-wayness is defined for a general family f . It is the realm of mechanisms to assign particular member functions to the member function identifiers of the cryptographic scheme such that the requirement of one-wayness is satisfied.

Definition 3.15 Cryptographic Scheme

A *cryptographic scheme* $H = (genPrekey, Dom, Op, Req)$ with security parameter $k \in \mathbb{N}$ consists of the following:

- an external interface $prek \leftarrow genPrekey(k)$ of an algorithm called *prekey generator* that on input a *security parameter* k outputs a string $prek$, called *prekey*. The prekey serves as a common input to all participants of a cryptographic scheme.
- A finite set $Dom = \{D_1, D_2, \dots, D_m\}$ of families $D_i = \{D_{i, prek}\}_{prek \in [genPrekey(k)]}$ of *domains* ($m \in \mathbb{N}$).
- A finite set $Op = \{O_1, \dots, O_\ell\}$ ($\ell \in \mathbb{N}$) of *operations*, i.e., external protocol interfaces (Section 3.2.2 on p.19) defined on the domains in Dom . The prekey is a common input to each of these external interfaces by default. It will not be displayed explicitly.
- A finite set Req of *requirements*, i.e. algebraic and probabilistic expressions over $genPrekey$, member operations of Op and member domains of Dom . Computational security requirements of a cryptographic scheme typically contain phrases like “polynomial size” parameters, negligible probability, etc. All these polynomials are in the security parameter k of the cryptographic scheme. ♦

3.3.1 Notational Conventions

SECURITY PARAMETER

The only input to algorithm $genPrekey$ is the security parameter k . By convention, k is written to the input tape of $genPrekey$ in unary representation, which is usually denoted as

$$1^k = \underbrace{11\dots1}_{k\text{-times}} \quad .$$

The reason for using unary representation for this input k is that otherwise $genPrekey$ cannot be a probabilistic polynomial-time algorithm, i.e., polynomial in the size of its input. Having mentioned this technicality once, we will henceforth use more relaxed notation and simply write $genPrekey(k)$ instead of $genPrekey(1^k)$.

PREKEYS

Since the prekey is an input given to each participant in each operation of a cryptographic scheme, we relax the input parameter lists of all protocols by omitting the prekey.

Each family $D \in Dom$ of domains has the prekey as its index. Most often, we need to refer to family members D_{prek} , not to the family D as a whole. Whenever we mean the family D , we always use the word “family” in front of D . This allows us to omit the index from domain family members where appropriate.

RESOLVING NAME CLASHES ACROSS CRYPTOGRAPHIC SCHEMES

If one cryptographic mechanism is defined in terms of another and both contain structures with the same names (*polymorphic identifiers*), then we may qualify the names by an upper index in brackets. For example, if cryptographic mechanism A provides a protocol *confuse*, which is implemented by using a protocol *confuse* of cryptographic mechanism B , then we write the latter as *confuse*^(B).

3.3.2 Security Requirements and Attackers

Typically, a cryptographic scheme has some security requirements set out in *Req*. A security requirement *req* describes what shall or shall not happen even if some of the participants in some protocols deviate from these protocols. In general, a security requirement is not only concerned with how an attacker can attack one operation of the cryptographic scheme, but rather how an attacker can control several participants in several of the scheme’s operations in order to reach his goals. Thus, a security requirement is expressed by 4 elements:

1. The *attacking schedule* describes which participants are honest, which are under control of the attacker, and how the *attacking participants* can interact with the *honest participants*. For example, consider three participants, a recipient A , a sender B , and an eavesdropper E . Honest participant A generates an encryption key pair and publishes the encryption key of it, then honest participant B encrypts a set of plaintexts using the encryption key, and sends them to A . Finally the attacking eavesdropper E may read all the ciphertexts sent by B (ciphertext-only-attack).
2. The *attacker model* describes the computing resources of the attacking participants. Formally, the attacker model is a set S of ITMs, and the security requirement must be met by all ITMs of S when in place of the attacking participant(s). The larger one chooses the set S , the stronger attacker is considered, and thus the stronger security requirement is formulated. It is common to call an attacking participant
 - *computational* if all ITMs of S are probabilistic polynomial-time.
 - *computationally unlimited*, if all ITMs of S are so.
 - *passive*, if all ITMs of S share the same external interface and same probabilistic execution function as the attacking participant they replace, and *active* otherwise. If we write an attacking schedule with an active attacker \tilde{A} , the private input and output parameters of \tilde{A} at the external interface may depend upon \tilde{A} and are therefore left untyped. We use a composition tape Q to denote the attacker’s private input (*initial content*) and private output. If the active attacker is polynomial-time, then the initial content should be limited to *polynomial-size*, i.e., the number of bits contained on Q before the attack starts is bounded above by a polynomial in the scheme’s security parameter.

3. The *success condition* describes the goal of the attacking schedule, e.g., to figure the private decryption key or a plaintext corresponding to any of the ciphertexts.
4. The *success measure* describes how hard it is for the attacker, i.e., the coalition of all attacking participants, to meet the success condition. In principle, an attacker has at least a small chance of meeting any success condition, simply by guessing correctly. Hence, the success measure is usually formulated by a lower or upper bound of the probability that the attacker meets the success condition. The lower or upper bound may or may not depend on the size of the inputs of the participants under the attacker's control. The former case relates to probabilistic polynomial-time attacking participants, the latter to computationally unlimited attacking participants. The resulting security requirement is called *computational* or *unconditional* (sometimes also called *perfect*), respectively.

Notational Convention: If an attacking schedule identifies a participant X as attacking (X typically a capital letter), then formally, X is replaced by a number of ITMs ranging over a subset of interactive Turing machines that depends on the specified attacker model. Each of the replacing ITMs is usually denoted \tilde{X} (read X crooked). Sometimes, the attacking schedule is described in natural language, sometimes it is specified by using the protocol notation. In the latter case we call the attacking schedule an attacking protocol and we denote it by the letter \mathcal{A} .

Definition 3.16 Cryptographic Mechanisms

A mechanism, implementation, or instance C with security parameter k of a cryptographic scheme $H = (\text{genPrekey}, \text{Dom}, \text{Op}, \text{Req})$ is any 3-tuple consisting of:

- (i) an algorithm with external interface $\text{prek} \leftarrow \text{genPrekey}(k)$ or a corresponding multi-party computation;
- (ii) a set of families d_1, d_2, \dots, d_m of domains $d_{1, \text{prek}}, d_{2, \text{prek}}, \dots, d_{m, \text{prek}}$, i.e., basic structures such as sets, groups, vector spaces, relations, functions, etc., such that the elements of all domains of family d_i ($i = 1, \dots, m$) are represented by $\text{poly}_i(k)$ bits, where $\text{poly}_i(k)$ is some polynomial in k .
- (iii) a set of ℓ efficiently computable protocols pr_1, \dots, pr_ℓ with respective external interfaces O_1, \dots, O_ℓ , where each input and output parameter takes values of one of the domains $d_{1, \text{prek}}, d_{2, \text{prek}}, \dots, d_{m, \text{prek}}$,

such that, after assigning the domain family variables $D_i \leftarrow d_{i, \text{prek}}$ for $i \in [1, m]$, all the requirements in Req are satisfied. ◆

3.3.3 Generating Prekeys

Throughout this work, prekeys are always presumed to be chosen correctly. In particular, we do not consider attacks on generating prekeys. We always presume that the inputs of the prekey generator are somehow negotiated in advance and are then fed to a trusted machine that runs the prekey generator and broadcasts the resulting prekey consistently to all participants of this instance of the cryptographic scheme.

If no trusted process is available to generate a prekey, but at least a majority of all participants is honest, the prekey can always be generated by a *multi-party computation* [10].

3.3.4 Encoding Domain Elements

We assume for all cryptographic mechanisms that the elements of all domains from which elements need to be chosen at random are encoded densely enough such that random choices can be made in polynomial time using the simple pick-and-test algorithm of Definition 3.1 on p.15.

For example, if k is the security parameter of a cryptographic mechanism, $prek$ be prekey generated on input k , and d_{prek} is a domain containing 2^k elements, then each such element shall be encoded as $(k + \nu)$ -bit integers, where ν is some integer constant, typically depending on the word size of the computer to be used. In each round, the simple pick-and-test algorithm then chooses a $(k + \nu)$ -bit integer and the probability that it actually is a modulo p residue is $2^k / 2^{k+\nu} = 2^{-\nu}$. Thus, the pick-and-test algorithm succeeds after expected ν rounds, which is polynomial in k .

3.3.5 Participant Identifiers

The protocols of a cryptographic mechanism (list number (iii) of Definition 3.16 on p.25) are specified by their external interfaces according to Definition 3.8 on p.19. The protocol identifiers of this list must be pairwise different. The identifiers of the input and output parameters and the participant identifiers used to label the square brackets that contain the input and output parameters are formal parameters in the external protocol interfaces in which they occur. Thus within each external interface the same identifier always carries the same value, but an identifier occurring in two different external interfaces, usually takes different values in each external interface.

In the requirements section though, the input parameters, output parameters and participant identifiers of all external interfaces are actual parameters, and therefore the same parameter denotes the same value throughout the entire definition of each requirement. For better readability, we will usually use unique actual identifiers not only throughout each separate requirement definition, but across all requirement definitions of a cryptographic mechanism.

3.3.6 Typed Parameters

The prekey generator of a cryptographic mechanism defines the domain families of the cryptographic scheme. Thus it is more useful to denote the messages read and written by ITMs as *typed parameters*, not as “flat” binary strings. For example, if a cryptographic mechanism implements the input I of participant T of a protocol pr by a tuple (a_1, \dots, a_ℓ) of parameters from the domains A_1, \dots, A_ℓ , then we write in the external interface of T in pr $^T[a_1, \dots, a_\ell]$ instead of $^T[I]$.

3.3.7 Notation of Protocols

Our protocols are denoted as tables. The header of the table gives the external interface of the protocol except the types of the parameters, which are given in the accompanying verbal description of the protocol. Each row of the table represents a round and each column represents an ITM of the protocol. For convenience, rounds may be subdivided into single steps of local computations of one or more ITMs. Steps are numbered globally throughout a protocol. This enables concise and unambiguous references. We use a declarative mathematical syntax to describe each step. A step consists of either of the following actions:

1. Choosing a value $x \in_R X$ from a sample space uniformly at random. To enhance readability, this kind of action is always displayed in the first step of a protocol.

2. Evaluating an expression $expr$ and assigning the result to a variable x : $x \leftarrow expr$. The expression may be arithmetic, it may be the call of an algorithm, or that of another protocol. The latter case is indicated by an arrow sitting below the assignment and pointing to the main recipient⁵ of the protocol.
3. Proceeding the execution if and only if a condition c holds is denoted by the phrase “proceed iff c ”. After each step, there are optional communication actions in which each participant can send the content of one or more variables to one or more other participants. If, for example, after step (5) of a protocol, Alice sends variables $a \in A$ and $b \in B$ to Bob, then we write in or after the row describing step (5) a horizontal arrow pointing from Alice’s to Bob’s column and label it a, b . Implied in this notation is a verification at the recipient’s end whether the received values a' and b' are in fact elements of the expected domains A and B . More generally, any communication arrow in our protocol notation implies that the recipient verifies that all received values are elements of their respective domains. Otherwise the recipient would abort the protocol execution immediately.

Two consecutive actions are separated by a semicolon. If the two actions are in different steps and are therefore displayed on separate lines, the semicolon between them may be omitted.

Some of these conventions are illustrated in Figure 3–1 on p.27 by example of Diffie-Hellman key agreement [94]. Alice and Bob take a common input g and Alice takes private input x from a suitable

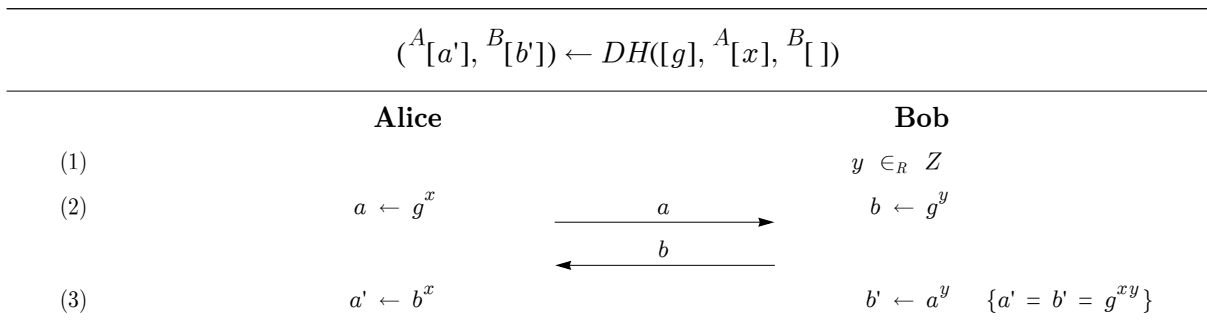


FIGURE 3–1 Diffie-Hellman Key Agreement

domain Z , which is part of the protocol definition, but is omitted here. Bob chooses uniformly at random a value y from Z . After exchanging the intermediate values a and b , they output the same private value $a' = b'$.

To improve readability of protocols, we insert comments in curly brackets where appropriate. A comment may explain a statement or indicate a post condition according to Hoare [132] (see the example after step (3) in Figure 3–1 on p.27). Comments sit in the same step as the statement they explain. Algorithms are denoted as one-party protocols.

3.4 Discrete Log Frameworks and Settings

Throughout this work, we model polynomial-time attackers by probabilistic polynomial-time Turing machines. It has been argued that this may be too weak an attacker model and therefore non-uniform

5) The notion of a main recipient is informal, but in every particular case, it will be clear who the main recipient is.

models, e.g., polynomial-size circuits, have been proposed by Goldwasser, Micali, Rackoff [123] and others. Nevertheless, it has remained somewhat controversial whether uniform or non-uniform attacker models are more adequate. Moreover, as Pfitzmann [180] and Goldreich [117] have pointed out, the uniform model is technically superior in the following sense. Typically, one is interested in reducing an attack on a particular cryptographic mechanism to some complexity theoretic assumption. If these assumptions are stated uniformly and one can find uniform reductions, then the same proof of security carries over to the non-uniform framework, while the converse is not always true. In this work, we rely completely on the uniform framework because wherever we have found reductions, we could find a uniform one.

Throughout the following sections, p denotes a large prime. If all prime factors of $p - 1$ are small, the discrete logarithm modulo p can be computed in time $O(|p|_2^2)$ [190]. (For recent achievements in computing discrete logs see [124,153,145].) We will therefore always require that $p - 1$ has at least one large prime factor, denoted as q .

Definition 3.17 Discrete Log Framework and Settings

Let the *discrete log framework* DLF be the family $\{dls_k\}_{k \in \mathbb{N}}$ of sets

$$dls_k = \{(p, q) \mid q \mid p - 1 ; \lambda(k) \leq |q|_2 < |p|_2 = k\}$$

of *discrete log settings*. Each discrete log setting is a pair (p, q) of different primes, where q divides $p - 1$, The prime p is k bit long. The prime q is at least $\lambda(k)$ bit long, where the function $\lambda(k)$ is depends on the state of the art of algorithms that compute discrete logarithms. A discussion of choosing k and $\lambda(k)$ is given in the remark following Assumption 3.20 on p.29.

Given a prime p , and $x, y \in \mathbb{Z}_p^*$, the smallest non-negative integer e , such that $x^e = y \pmod{p}$ is denoted $\log_x y$ and is called the *discrete logarithm* of y with respect to x . If such an integer does not exist, $\log_x y$ is undefined. ◆

Remarks: With a discrete log setting (p, q) , we associate four algebraic structures, namely, the finite field \mathbb{Z}_p of residues modulo p , the unique cyclic subgroup G_q of order q in $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$, and the finite field \mathbb{Z}_q . Of these algebraic structures, we will use three kinds of arithmetic, namely multiplication in G_q , i.e., modulo p , and addition and multiplication in \mathbb{Z}_q , i.e., modulo q . Throughout the following work, the calculations mod p are more frequent. To make reading easier but keep notation simple, we will sometimes omit “mod p ”, but always display “mod q ”.

Observation 3.18 Generators in Discrete Log Settings

Given a discrete log setting $(p, q) \in dls_k$, the group G_q is cyclic because it is a subgroup of the cyclic group \mathbb{Z}_p^* . Because q is prime, it is efficient to test membership in G_q and to generate members of G_q uniformly at random based on the following facts:

Test membership: $\forall x \in \mathbb{Z}_p^* : x \in G_q \Leftrightarrow x^q = 1 \pmod{p}$,

Generate members: $\forall x \in \mathbb{Z}_p^* : x^{(p-1)/q} \in G_q$.

Generators g of G_q can be chosen uniformly at random by choosing an element $x \in \mathbb{Z}_p^* \setminus \{1\}$ uniformly at random and computing $g = x^{(p-1)/q} \bmod p$. If $g \neq 1$, then it is a generator, otherwise repeat the process with a different x .

These facts have been shown by Pfitzmann [183] in Construction 8.22.

3.5 Double Discrete Log Frameworks and Settings

We will also make use of pairs of discrete log settings that are connected in a special way. We will call such pairs *double discrete log setting*. The underlying idea was used by Stadler to construct publicly verifiable encryption schemes [216] and later by Camenisch and Stadler to construct group signature schemes [51].

Definition 3.19 Double Discrete Log Framework and Settings

Let the *double discrete log framework* $DDL F$ be the family $\{ddl s_{k, \ell}\}_{k, \ell \in \mathbb{N}}$ of sets

$$ddl s_{k, \ell} = \{(P, p), (p, q) \mid (P, p) \in dls_k; (p, q) \in dls_\ell\}$$

of *double discrete log settings*, i.e., pairs of discrete log settings such that the second prime of the first discrete log setting equals the first prime of the second discrete log setting. \blacklozenge

3.6 Complexity Theoretic Assumptions

We will state a couple of complexity theoretic assumptions that we shall use throughout this work. They are formulated in terms of probabilities negligible in a security parameter. Since these assumptions will be used to prove that particular cryptographic mechanisms satisfy certain security requirements of cryptographic schemes, we shall present the assumptions themselves within the framework of cryptographic schemes.

3.6.1 Subgroup Discrete Log Assumption

Assumption 3.20 Subgroup Discrete Log Assumption (SDLA)

For all probabilistic polynomial-time Turing machines \tilde{A} and for each positive constant c , all sufficiently large k , the probability that, after choosing a discrete log setting $(p, q) \in_R dls_k$, two elements $x \in_R G_q \setminus \{1\}$ and $y \in_R G_q$, \tilde{A} takes input (k, p, q, x, y) and finds the discrete logarithm of y with respect to x is bounded above by k^{-c} (cf. [157]). More precisely:

$$\forall PPT \tilde{A} \forall c > 0 \exists k_0 \forall k > k_0:$$

$$Prob[(e = \log_x y) \setminus (p, q) \in_R dls_k; x \in_R G_q \setminus \{1\}; y \in_R G_q; e \leftarrow \tilde{A}(k, p, q, x, y)] < k^{-c} . \quad \blacklozenge$$

Remarks: The assumption that it is hard to compute discrete logarithms modulo p has been widely used in cryptology during the last 20 years [94, 29, 99, 15, 208, 42, 91, 71, 183]. As in Schnorr [208, 209], we use a slightly modified assumption, namely that it is still hard to compute discrete logarithms of primes p where a large prime factor of $p-1$ is known. However, this seems to be no stronger than the usual assumption, as it is generally believed that such primes p are among the hardest for computing discrete logarithms [177].

Given current cryptographic practices, p shall be of binary length $|p|_2 > 1024$ bit. In general, the acceptable minimum length of q is a function of the length of p and is determined by the trade-off between logarithm finding algorithms whose complexity is dominated by q and those whose complexity is dominated by p . In the early 1990's during the discussion of the digital signature standard DSS [165], Hellman [130] and Rivest [202] considered 160-bit a suitable length for q if p is about 1024 bit long. Recent guidance has been given by the Computer Security Resource Center (CSRC) of NIST [166,239] and by Lenstra and Verheul [146].

3.6.2 Subgroup Representation Assumption

We will also need the following closely related assumption about computing representations of a given element of G_q with respect to a tuple of generators of this group.

Assumption 3.21 Subgroup Representation Assumption (SRA)

For every fixed n , every probabilistic polynomial-time attacker \tilde{A}_n , each positive constant c and sufficiently large k , the probability that, after choosing a discrete log setting $(p, q) \in_R dls_k$, a tuple (g_1, \dots, g_n) of generators of G_q , and an element $y \in_R G_q$, \tilde{A}_n takes input $(k, p, q, (g_1, \dots, g_n), y)$ and finds a *representation* $x = (x_1, \dots, x_n)$ of y with respect to (g_1, \dots, g_n) , i.e.

$$y = \prod_{i=1}^n g_i^{x_i} \quad ,$$

is bounded above by k^{-c} (cf. [157]). More formally:

$$\forall n \forall PPT \tilde{A}_n \forall c > 0 \exists k_0 \forall k > k_0$$

$$\text{Prob}[(y = \prod_{i=1}^n g_i^{x_i}) \wedge (p, q) \in_R dls_k ; (g_1, \dots, g_n) \in_R (G_q \setminus \{1\})^n ; y \in_R G_q ; \\ (x_1, \dots, x_n) \leftarrow \tilde{A}_n(k, p, q, (g_1, \dots, g_n), y)] < k^{-c} \quad . \quad \blacklozenge$$

Observation 3.22 Equivalence of SRA and SDLA

In the *DLF*, the subgroup representation assumption is equivalent to the *SDLA*. As Chaum, van Heijst and Pfitzmann have shown in [71] (Lemma 2), the discrete log assumption implies collision resistance of tuple exponentiation, i.e. finding two different representations for one element in groups of prime order is infeasible. This further implies one-wayness of tuple exponentiation and thus the representation assumption. The other direction is easy and is shown for example by Brands [34] (Proposition 7). \square

3.6.3 Subgroup Diffie-Hellman Assumption

Assumption 3.23 Subgroup Diffie-Hellman Assumption (SDHA)

For all probabilistic polynomial-time Turing machines \tilde{A} and all positive constants c , all sufficiently large k , the probability that after choosing a subgroup discrete log setting $(p, q) \in_R dls_k$, a generator $g \in_R G_q \setminus \{1\}$, two elements $a, b \in_R \mathbb{Z}_q$ and an element $h \in_R G_q$, \tilde{A} takes input $(k, p, q, g, g^a \bmod p, g^b \bmod p, h)$ and returns TRUE if $h = g^{ab} \bmod p$ and FALSE otherwise is bounded above by k^{-c} (cf. [157] Section 3.7). More formally:

$$\forall PPT \tilde{A} \forall c > 0 \exists k_0 \forall k > k_0 :$$

$$\text{Prob}[(z = g^{ab} \bmod p) \setminus (p, q); \text{dls}_k; g \in_R G_q \setminus \{1\}; a, b \in_R \mathbb{Z}_q; \\ z \leftarrow \tilde{A}(k, p, q, g, g^a \bmod p, g^b \bmod p)] < k^{-c} . \quad \blacklozenge$$

In the DLF, the Subgroup Diffie-Hellman Assumption implies the Subgroup Discrete Log Assumption. That is, an attacker who can break the SDL Assumption 3.20 on p.29 and is given a Diffie Hellman input $k, p, q, g, g^a \bmod p, g^b \bmod p$ can easily compute the two discrete logs a and b and thus also the power $g^{ab} \bmod p$. Also see [157] Section 3. The converse is not known to hold. \square

3.7 Random Oracle Model

We follow Bellare’s overview of the random oracle model [12], which was introduced by Bellare and Rogaway [14] as a “bridge between theory and practice”. The idea is a simple one: namely provide all parties of a protocol—good and bad alike—with access to a (public) function h ; prove correct the protocol assuming h maps each input to a truly random output, i.e., a random oracle; later, in practice, set h to some specific function derived in some way from a standard cryptographic hash function like SHA-1 [165] or RIPEMD-160 [96].

The random oracle model buys efficiency, and as Rogaway claims, security guarantees, which although not at the same level as those of the standard provable security approach, are arguably superior to those provided by totally ad hoc protocol design.

The overly skeptical might say a security proof in the random oracle model gains nothing because the function h that we actually use in the final protocol is not random. Here is another way to look at it. In practice, attacks on schemes involving a SHA-1 derived h and number theory will often themselves treat h as random. Bellare and Rogaway call such attacks *generic*. In other words, cryptanalysis of these “mixed” protocols is usually done by assuming h is random. But then proofs in the random oracle model apply, and indeed show that such generic attacks will fail unless the underlying number-theoretic problems are easy. In other words, the analysis at least provably excludes a certain common class of attacks, namely generic ones.

It is important to choose carefully the instantiating function h . The intuition stated by Bellare and Rogaway in [14] is that the resulting protocol is secure as long as the protocol and the hash function are sufficiently “independent”, meaning the protocol does not itself refer to the hash function in some way. This is a fuzzy guideline that needs more work in the future.

An important step in our understanding of the random oracle model was taken by Canetti, Goldreich, and Halevi [56]. They show there exist protocols secure in the random oracle model but insecure under any instantiations in which we substitute a function from a small family of efficiently computable functions. Their examples however are somewhat contrived, and this kind of situation does not arise with any of the “real” cryptographic mechanism in the literature and the following work.

In comparison with totally ad hoc design, a proof in the random oracle model has the benefit of viewing the protocol with regard to its meeting a strong and formal notion of security, even if this is assuming some underlying primitive is very strong. This is better than not formally modeling the security of the protocol in any way. This explains why the random oracle model is viewed as a “bridge between theory and practice” [14].

3 NOTATIONS AND ASSUMPTIONS

4

Cryptographic Tools

*“Only those defenses are good, certain and durable,
which depend on yourself alone and your own ability.”
The Prince [150] — Niccolò Machiavelli*

Among the many classes of cryptographic primitives, there are two of particular importance for the implementation of credential schemes in Section 5 on p.59: interactive proofs of knowledge (Section 4.1 on p.34) and blind signatures (Section 4.2 on p.51). Interactive proofs of knowledge are a well-known concept in cryptography and we can mainly rely on existing material. However, we need to extend the concept of proofs of knowledge (see Bellare and Goldreich [13]) from two to three parties.

In contrast to conventional signatures (see Goldwasser, Micali, Rivest [122] and well-known practical examples like RSA [201], Fiat-Shamir [110], Schnorr [209], or Guillou-Quisquater [125]), blind signature schemes output blinded messages and corresponding signatures directly to the recipient and hide them from the signers. They were introduced by Chaum [58,59] as practical tools to design payment and credential schemes (see Section 5.1 on p.60). For offline electronic cash, however, blind signature schemes in general are too liberal, because they allow recipients to obtain signatures for more coins than the signer provides. Therefore, double spending is a notorious threat to these schemes. Independently, Brands [34,35] and Franklin and Yung [113] have introduced the same sort of specialized blind signatures, which they called *restrictive blind signatures* and *oblivious signatures*, respectively. Both their cryptographic mechanisms and all later proposals are one-time restrictive blind signatures in the sense that from each execution of the signing protocol, the recipient can draw a signature for only one message. This exactly is needed for offline electronic cash. We recap the restrictive blind signature scheme by Chaum and Pedersen, which was later used by Brands, in Section 4.2.2 on p.55.

Personal certificate schemes (Section 2.4 on p.9) are another application area of blind signature schemes, where holders can show their own certificates arbitrarily often, but cannot show certificates of other holders. To achieve an efficient personal certificate mechanism, we will use a special kind of ElGamal encryption mechanism and again the restrictive blind signature scheme of Chaum and Pedersen (Section 5.4 on p.90).

4.1 Proof-of-Knowledge Schemes

A proof of knowledge is a (usually) interactive protocol between a prover and a verifier who are given a priori a set Z of candidates, a set of witnesses W and a binary relation $R \subseteq Z \times W$. Before the protocol starts, the prover suggests a candidate $z \in Z$ to the verifier, and then tries to convince the verifier of “knowing” a witness $w \in W$ for z . A proof of knowledge protocol is secure if “whenever” the verifier is “convinced” on input z , then at least one satisfying witness exists and the prover indeed “knows” at least one satisfying witness w such that $(z, w) \in R$. The clue to a formalization of “proofs of knowledge” is an appropriate interpretation of the phrases “whenever” and “knows” which appear in the condition. As usual, the phrase “convinced” means to enter a specified accepting state in the computation.

Suppose for simplicity a good prover, i.e., one that always convinces the verifier on a given candidate z . Saying that the prover “knows” a satisfying witness for the given candidate z means that the prover “can be modified” so that it outputs a witness w such that $(z, w) \in R$. The notion of “possible modifications of the prover ITM” is captured by efficient algorithms that use the prover ITM as an oracle. Hence, saying that a prover knows a witness will be formalized as the feasibility to compute a witness by using the prover as an oracle. Namely, there exists an efficient and uniform algorithm—called the *knowledge extractor*—, that on input a candidate and given oracle access to a good prover is able to output a satisfying witness.

Bellare and Goldreich [13] have elaborated this program and came up with a widely accepted definition of proofs of knowledge. We are going to present their definition within the framework of a cryptographic scheme. Previous and less robust definitions of interactive proofs of knowledge were given by Feige, Fiat, Shamir [106,107], Brassard, Chaum, Crépeau [40], and Tompa, Woll [222].

Note that we will not deal with *proofs of membership*, a concept that was originally introduced by Goldwasser, Micali, Rackoff [123] as *interactive proof systems* and was further developed by Goldreich [234]. The goal of the prover in a proof of membership is to convince the verifier that a given candidate is a member of a given set or language. More precisely, the verifier is given a candidate x and a language L . The prover’s task is to prove to the verifier that $x \in L$. If an honest prover always succeeds to convince the verifier with overwhelming probability in case $x \in L$, the system is said to be *complete*. If a cheating prover can convince the verifier with only negligible probability in case $x \notin L$, the system is said to be *sound*.

4.1.1 Definition of Proof-of-Knowledge Schemes

In a proof-of-knowledge system the prover suggests a candidate z to the verifier, and the prover’s task during the following protocol is to prove to the verifier that the prover ‘knows’ a witness w such that $(z, w) \in R$. If an honest prover convinces the verifier with high probability if only he ‘knows’ a witness matching z , then the system is called *complete* or non-trivial (Bellare and Goldreich in [13]). If whenever a cheating prover convinces the verifier, a witness can feasibly be extracted from the prover, then the system is called *valid*. This will be covered by Definition 4.1 on p.35.

The definition as such is too weak to be useful, because it can be trivially satisfied; for example, by a prover who simply passes a satisfying witness to the verifier. The prover’s whole point to engage in an interactive protocol is to NOT REVEAL the witness to the verifier. Such a confidentiality require-

ment must be stated in addition to Definition 4.1 on p.35. Basically three different confidentiality requirements on witnesses have been proposed in the literature: Zero-knowledge, witness hiding, and witness indistinguishability. They will be addressed in Section 4.1.3 on p.37. Each of them defines what it means to keep the prover's witness confidential from the verifier during the interactive protocol following the input of a candidate. None of these three confidentiality requirements cares about how much information about a witness of a given candidate z is leaked to the verifier by the relation R itself. Apparently, for each candidate z , the relation R should release virtually no information about a witness of z to the verifier. Otherwise it would be pointless to keep a witness confidential during the interactive protocol. In fact, all concrete proof-of-knowledge protocols in the literature that the author is aware of, and all proof-of-knowledge protocols in this work use *one-way functions* [157] Definition 1.12. $R : w \mapsto z$ in place of relation R . Thus, we will define proof-of-knowledge schemes over functions only, and we label these functions as '*make*' instead of the more general relation identifier ' R '. We shall require the one-way property of functions *make* only in the definition of confidentiality requirements in Section 4.1.3 on p.37.

Definition 4.1 Proof-of-Knowledge Scheme

A proof-of-knowledge scheme with security parameter k consists of the following prekey generator, domains, operations and requirements:

GENERATE PREKEY $prek \leftarrow gen.Prekey(k)$

A probabilistic operation that takes as input a *security parameter* $k \in \mathbb{N}$ and outputs a *prekey* $prek$.

DOMAINS

A proof-of-knowledge-scheme has a set $\{Z, W, make\}$ of 3 domain families whose respective members are:

- the *candidate* domains Z_{prek} .
- the *witness* domains W_{prek} .
- the *making functions* or partial functions $make_{prek} : W_{prek} \rightarrow Z_{prek}$. If $z = make_{prek}(w)$, then we say that witness w *makes* candidate z .

In order to ease notation, we do not display the domain index $prek$ in the following.

OPERATIONS

A proof-of-knowledge scheme has a set $\{prove\}$ of one operation satisfying 2 requirements: completeness and validity.

Prove $V[accept] \leftarrow prove([z], P[w])$

A probabilistic operation where the prover P and the verifier V take as common input a candidate $z \in Z$, and the prover takes as private input a witness $w \in W$. The verifier returns a Boolean output. If the output is $accept = TRUE$, we say that the prover *convinces* the verifier of z .

This is a proof-of-knowledge scheme iff each implementation is a proof of knowledge [13] over *make*:

COMPLETENESS

Let a candidate z be given to the prover and the verifier. If the prover follows protocol *prove* honestly and takes a making witness of z as input, then the verifier is always convinced (with zero error probability). More precisely: For all security parameters k , for all prekeys $prek \in [genPrekey(k)]$, a prover P knowing a witness w of z such that $z = make(w)$ always convinces the verifier V , i.e.

$$\forall k \in \mathbb{N}, prek \in [genPrekey(k)], w \in W, z = make(w) : prove([z], P[w]) = V[TRUE] .$$

VALIDITY

Consider any attacking ITM \tilde{P} in place of the prover, and let \tilde{P} take a candidate z and any additional input, which may or may not be a making witness of z . Then, the more likely \tilde{P} is to convince the verifier, the faster can a witness be extracted from \tilde{P} . More formally according to Bellare and Goldreich [13]: There exists a constant $c > 0$ and a PPT E called *knowledge extractor* [107] such that for all polynomial-time attacking provers \tilde{P} , for all sufficiently large security parameters k , for all prekeys $prek \in [genPrekey(k)]$, all candidates $z \in Z$ such that $p(z) > 0$ and all polynomial-size initial contents Q of \tilde{P} , the expected computation time of E (Definition 3.10 on p.20) to extract a witness w from \tilde{P} such that $z = make(w)$ is bounded above by $k^c / p(z)$, where $p(z)$ is the probability¹ that \tilde{P} convinces the verifier on common input z , i.e.

$$\exists c > 0, PPT E \ \forall ITM \ \tilde{P} \exists k_0 \forall k > k_0, prek \in [genPrekey(k)], z \in Z, |Q| < k^c :$$

$$\text{if } p(z) > 0 \text{ then } ExpTime[(z = make(w)) \setminus w \leftarrow E(z, \tilde{P}(z, Q))] < \frac{k^c}{p(z)} ,$$

where $\tilde{P}(z, Q)$, in the parameter list of E , denotes that E has oracle access to \tilde{P} on input z, Q . ♦

Remark 1: Differences to the definition given by Bellare and Goldreich [13].

- (i) We use interactive Turing machines instead of interactive functions in [13] as our model of computation. They are equivalent models of computation according to Church's thesis.
- (ii) A proof-of-knowledge *scheme* comprises a specific prover and a matching verifier. Bellare and Goldreich have defined a "knowledge verifier" by requiring that a prover exists for it such that they both together form a proof-of-knowledge system. Our definition of proof-of-knowledge scheme makes explicit both, the prover and the verifier, which is in a sense more constructive.
- (iii) Bellare and Goldreich give a slightly more general definition that allows the dishonest prover \tilde{P} a small error probability (*knowledge error function*) of convincing the verifier although \tilde{P} does not know a witness. This kind of error probability does not occur in any of the proof-of-knowledge in this work.

Remark 2: Having oracle access to the ITM $\tilde{P}(z, Q)$ one can do two things with \tilde{P} : One can run \tilde{P} on any candidate z and observe the messages \tilde{P} sends during the emerging execution. Moreover, after \tilde{P} 's execution has stopped, one can rewind \tilde{P} back into any state visible from the outside, namely any state where \tilde{P} is waiting for a message to be received from its environment. In other words, one can branch out any execution of \tilde{P} at certain "branch points" and reveal \tilde{P} 's behavior in

1) taken over the random choices of the prover P and the verifier.

all branches. Executing \tilde{P} (regardless from which state) counts as one computation step, and rewinding \tilde{P} does not count as computation time.

Remark 3: Bellare and Goldreich [13] have suggested *soundness* as an optional security requirement for proof-of-knowledge schemes. Informally, soundness is the assurance that for candidates that have no witness the prover can convince the verifier of knowing a witness only with negligible probability. We do not consider soundness further because in all our implementations of proof-of-knowledge schemes, all candidates have at least one witness, and therefore soundness is an obsolete requirement anyway.

4.1.2 Proving Linear Relations of Witness Components

If the witness domains are n -dimensional vector spaces W , the verifier might ask the prover not only to prove mere knowledge of a witness $(w_1, w_2, \dots, w_n) \in W$ for a given candidate z , but also that the components of the witness satisfy certain linear relations. Stefan Brands [38] has proposed efficient protocols for proving knowledge of witnesses that satisfy any Boolean formula over one or more arbitrary linear relations. We will make use of a special case of his nice and general result (see Mechanism 4.5 on p.46).

4.1.3 Zero-Knowledge, Witness Indistinguishability, and Witness Hiding

Additional security requirements of proofs of knowledge are *zero-knowledge* according to Fiat and Shamir [106], *witness indistinguishability* and *witness hiding* introduced by Feige and Shamir in [107].

Zero-knowledge is the strongest of the three as it requires that during the proof protocol the verifier learns no more information than what the verifier had before the proof. Witness indistinguishability requires that the verifier, even if he knows all possible witnesses matching the prover's candidate, cannot distinguish which of those witnesses the prover actually uses in the proof. Witness hiding requires that, during the proof, the verifier learns no more witnesses matching the prover's candidate than those the verifier knew before the proof.

We will use interactive proofs of knowledge for identification purposes and to construct digital signature schemes. We also need to repeat interactive proof protocols polynomially many times without losing security of identification or digital signatures. The security requirement most appropriate for these purposes is witness hiding for a number of reasons.

Digital signatures can be derived from interactive proofs of knowledge by proving knowledge of the signing key. The full or partial transcript of the proof then serves as a digital signature. It has been argued that such interactive proofs of knowledge must not be zero-knowledge for otherwise recipients of signatures were in no better position to show a valid signature to a third party verifier than non-recipients. Obviously, zero-knowledge is too strong a requirement. Witness hiding on the other hand appears necessary for the resulting signature scheme to be secure against a total or universal break (see Section 4.2 on p.51), but probably not sufficient because it does not rule out selective and existential forgery (see Section 4.2 on p.51).

Feige and Shamir [107] have shown that zero-knowledge is not preserved under polynomial composition of interactive proofs of knowledge ([107] Theorem 3.2), while witness indistinguishability is ([107] Theorem 3.1). Furthermore, they have shown that a proof of knowledge scheme that is witness

indistinguishable and whose candidates each have more than one witness, is also witness hiding ([107] Theorem 4.2).

Thus, we will employ interactive proofs of knowledge that are not zero-knowledge, but witness indistinguishable and have more than one witness for each candidate. Following Remark 2 of Feige and Shamir [107] we can argue that even polynomial compositions of these proof protocols are witness indistinguishable and therefore also witness hiding. This is currently one of the most appropriate foundations of interactive proofs of knowledge for the intended applications (identification and digital signature) although this foundation alone will be no sufficient argument for their security.

Definition 4.2 (Perfect) Witness Indistinguishability

Let $H = \{genPrekey, Dom, Op, Req\}$ be a cryptographic scheme with the following domains and operations:

- $Z, W, make \in Dom$ are two respective families Z, W of input domains and a family $make$ of one-way functions such that for all prekeys $prek: make : W \rightarrow Z$.
- $(^V[acc]) \leftarrow prove([z], ^P[w])$ is a two-party operation in Op , with a common input $z \in Z$. P takes private input $w \in W$, and V outputs a Boolean result $acc \in BOOL$.

A protocol $prove$ of an implementation of H is called *witness indistinguishable* for P over $make$ iff for all $c > 0$, all PPT judges J , all polynomial-time attacking verifiers \tilde{V} , for all sufficiently large security parameters k , for all prekeys $prek \in [genPrekey(k)]$, all common inputs $z \in Z$, each two private inputs w_1, w_2 such that $z = make(w_1) = make(w_2)$, all polynomial size initial content Q of \tilde{V} , the views of \tilde{V} on P using w_1 and w_2 are indistinguishable by the judge J^2 :

$$\forall c > 0, ITM J, \tilde{V} \exists k_0 \forall k > k_0, prek \in [genPrekey(k)], z \in Z, z = make(w_1) = make(w_2), |Q| < k^c$$

$$\left| \begin{aligned} & Prob[J(z, w_1, w_2, Q, view_{\tilde{V}}^P(prove([z], ^P[w_1], \tilde{V}[Q])))] = True \\ & - Prob[J(z, w_1, w_2, Q, view_{\tilde{V}}^P(prove([z], ^P[w_2], \tilde{V}[Q])))] = True \end{aligned} \right| \leq k^{-c} .$$

A protocol pr is *perfectly witness indistinguishable* for P over $make$, if the two views resulting from any two witnesses w_1 and w_2 are equally distributed, i.e.,

$$\forall ITM \tilde{V}, prek \in [genPrekey(\mathbb{N})], z \in Z, z = make(w_1) = make(w_2), Q :$$

$$view_{\tilde{V}}^P(prove([z], ^P[w_1], \tilde{V}[Q])) = view_{\tilde{V}}^P(prove([z], ^P[w_2], \tilde{V}[Q])) . \quad (4.1)$$

The operation $prove$ of H is (perfectly) *witness indistinguishable* for P over $make$ iff each protocol implementing $prove$ is so. \blacklozenge

Remark: Note that we require the family $make$ of functions to be *one-way*, such that, under suitable complexity theoretic assumptions, a computational verifier could not figure a witness from a given candidate. This is one way to avoid pathological cases where candidates easily reveal their witnesses.

2) Non-uniform definitions have been proposed by Yao [227] and were later used by Goldwasser and Micali [121] and by Goldwasser, Micali, Rackoff [123]. Their definition used a non-uniform judge, i.e., polynomial-size families of circuits. We follow Goldreich [117] in using a uniform definition where the judge is a probabilistic polynomial-time Turing machine.

For example consider the proof of knowledge protocol where each candidate $z = w$ is its own witness, the protocol *prove* contains no communication between the prover and verifier, and the verifier always outputs TRUE at the end of the protocol. This proof-of-knowledge protocol satisfies the requirements of completeness and validity, and it is also witness indistinguishable, witness hiding and zero-knowledge. Thus our extra requirement on the family *make* of functions.

Theorem 4.1 Witness Indistinguishability under Polynomial Composition

Let $H = \{genPrekey, Dom, Op, Req\}$ be a cryptographic scheme where

- for all $i = 1 \dots \ell$, $Z_i, W_i, make_i \in Dom$ are two families Z_i, W_i of input domains, and a family $make_i$ of functions such that for all prekeys *prek*: $make_i : W_i \rightarrow Z_i$.
- for all $i = 1 \dots \ell$, $(V_i[acc]) \leftarrow prove_i([z], P_i[w])$ are two-party operations in *Op* each of a prover P_i and a verifier V_i .

For all $i = 1, \dots, \ell$ let pr_i be a protocol implementing $prove_i$ such that pr_i is witness indistinguishable for P_i over $make_i$. Then every polynomial composition *pr* of the protocols pr_1, \dots, pr_ℓ (Definition 3.12 on p.21) is witness indistinguishable for the prover $P_1 ||| \dots ||| P_\ell$ (see Observation 3.7 on p.18) of *pr* over the family of functions

$$make : \prod_{i=1}^{\ell} W_i \rightarrow \prod_{i=1}^{\ell} Z_i$$

such that $make((w_1, \dots, w_\ell)) = (z_1, \dots, z_\ell)$ where $z_i = make_i(w_i)$ for all $i = 1, \dots, \ell$. ♦

Reference of Proof See Feige and Shamir [107] Theorem 3.1.

4.1.4 Extended Proof-of-Knowledge Schemes

Consider a (2-party) proof-of-knowledge protocol *prove* according to Definition 4.1 on p.35. Next, we want to extend the protocol *prove* by inserting a co-prover between the prover and the verifier such that the power to prove knowledge of a witness is shared by the prover *P* and the co-prover. Each one of them shall hold a share of the witness and they together can convince the verifier that prover and co-prover together know a witness for a given candidate.

For such a protocol *prove*, we consider 3-party protocols of a prover *P*, a co-prover *P'* and a verifier *V* as shown in (Figure 4-1 on p.40). *P* takes a witness *w* as private input, *P* and *P'* take a candidate *z* as shared input, *P'* takes a co-witness *v* as private input, and *P'* and *V* take a candidate *z'* as shared input. We call such a 3-party protocol *proveExt* an *extended proof-of-knowledge protocol* if the following requirements are met:

- (1) The communication messages between the prover *P* and co-prover *P'* “look and feel” to each of them like they were running the 2-party protocol *prove*. Likewise, the communication messages between the co-prover *P'* and the verifier *V* “look and feel” to each of them like they were running protocol *prove* as well (possibly on different inputs as the protocol instance *prove* before). More precisely, the sub-protocol of *proveExt* between *P* and *P'* ||| *V* and the sub-protocol of *proveExt* between *P* ||| *P'* and *V* each resembles the proof-of-knowledge protocol *prove* on respective inputs. This will be formalized by drawing an equivalence between each of the subprotocol instances of *proveExt* and a corresponding instance of the original 2-party protocol *prove*.

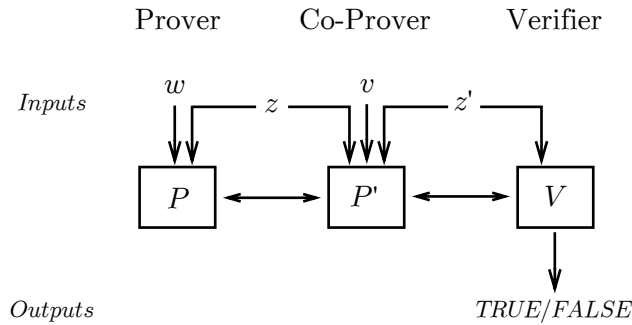


FIGURE 4-1 Communication Topology of a 2-Prover Proof of Knowledge

- (2) Consider an attacking prover \tilde{P} , an attacking verifier \tilde{V} and an honest co-prover P' . Assume a number $n \geq 1$ of parallel executions of *proveExt*, i.e., if in *proveExt* the prover (verifier) sends a message m to the co-prover P' , then in the parallel execution, the prover \tilde{P} (verifier \tilde{V}) sends n messages m_1, \dots, m_n to the co-prover P' . The same holds for messages received by the co-prover P' . In a permuted parallel execution, first a permutation π_n is chosen uniformly at random. During the parallel execution, the n messages in each batch sent by the co-prover P' to the verifier \tilde{V} or received by the co-prover from the verifier is π_n -permuted. In effect, the verifier \tilde{V} “sees” a π_n -permuted order of n executions compared to the prover \tilde{P} . The protocol *proveExt* is called *co-unlinkable* if each two attacking prover \tilde{P} and verifier \tilde{V} who execute the permuted parallel execution have no better chance of figuring out the permutation π_n than by pure guessing.

Roughly, this security property could be formalized using the following approach: If we execute *proveExt* between \tilde{P} , P' , \tilde{V} n times while collecting the resulting views v_1, v_2, \dots, v_n of \tilde{P} on the co-prover and the views w_1, w_2, \dots, w_n of \tilde{V} on the co-prover, then we require that for any pair of views (v_i, w_j) , which may have resulted from two different executions $i, j \in [1, n]$, there are equally many internal choices of the co-prover that would produce the given pair of views in an execution of *proveExt*. This is required to hold for all $n \in \mathbb{N}$.

The concept of unlinkability is of wider applicability. It has been introduced by Chaum and Pedersen [73] for blind signatures (see Definition 4.7 on p.53), where it captures the notion of blindness, and recurs also with credentials schemes (see Definition 5.6 on p.87).

Much of the literature on 3-party proof-of-knowledge protocols is about *divertible* proofs of knowledge, i.e., the special case of extended proofs of knowledge where the sub-protocol between $P ||| P'$ and V is equivalent to that between P and $P' ||| V$ in the sense of Definition 3.9 on p.20. The special case where P , P' and V take the same shared input $z = z'$ was introduced by Desmedt, Goutier and Ben-gio [92], and it was later defined formally by Okamoto and Ohta [173]. Chen [74] has considered the more generalized case where the candidates z, z' are taken from the same domain, but may be different. She calls them *divertible proof-of-knowledge protocols with different input*.

In the following, we consider a more general case of the latter where the witness domains and the co-witness domains may be different. We call them extended proof-of-knowledge schemes.

Definition 4.3 Extended Proof-of-Knowledge Scheme

An extended proof-of-knowledge scheme with security parameter k consists of the following prekey generator, domains, operations and requirements:

GENERATE PREKEY $prek \leftarrow genPrekey(k)$

A probabilistic operation that takes as input a *security parameter* $k \in \mathbb{N}$ and outputs a *prekey* $prek$.

DOMAINS

An extended proof-of-knowledge-scheme has a set $\{Z, W, +, V, make, make'\}$ of 6 domain families whose respective members are:

- the *candidate* domains Z_{prek} .
- the *witness* domains W_{prek} .
- the *making function* or partial function $make_{prek} : W_{prek} \rightarrow Z_{prek}$. If $z = make_{prek}(w)$, then we say that witness w *makes* candidate z .
- the *co-witness* domains V_{prek} .³
- the *witness addition functions* $(w, v) \mapsto w + v$ that map pairs $(w, v) \in W_{prek} \times V_{prek}$ of witnesses and co-witnesses to witnesses in W_{prek} .
- the *co-making function* or partial function $make'_{prek} : Z_{prek} \times V_{prek} \rightarrow Z_{prek}$. If $make'_{prek}(z, v) = z'$, then we say that the co-witness v *matches* the candidates z, z' .
- The making and co-making functions are connected in the following way: For each witness $w \in W_{prek}$, each co-witness $v \in V_{prek}$, and all candidates $z, z' \in Z_{prek}$:

$$\text{If } make'_{prek}(z, v) = z' \text{ then } (make_{prek}(w) = z) \Leftrightarrow (make_{prek}(w + v) = z') \quad (4.2)$$

In order to ease notation, we do not display the domain index $prek$ in the following.

OPERATIONS

An extended proof-of-knowledge-scheme has a set $\{prove, proveExt\}$ of 2 operations such that $(genPrekey, \{W, Z, make\}, \{prove\}, \{completeness, validity\})$ is a proof-of-knowledge scheme and the following extendedness requirement holds.

Prove $\overset{V^*}{[accept]} \leftarrow prove(\overset{P^*}{[z]}, \overset{P^*}{[w]})$

See operation *prove* in Definition 4.1 on p.35.

ProveExt $\overset{V}{[accept]} \leftarrow proveExt(\overset{P}{[z, w]}, \overset{P'}{[z, z', v]}, \overset{V}{[z']})$

A probabilistic operation of a prover P , a co-prover P' and a verifier V , where the prover P and the co-prover P' share as input the candidate $z \in Z$, and the prover takes as private input the witness $w \in W$. Likewise, the co-prover and verifier share as input the candidate $z' \in Z$, and the co-prover takes as private input the witness $v \in V$. If the verifier outputs $accept = TRUE$, we say that P and P' together convince V of candidate z' .

3) Note that V denotes the family of co-witness domains, a particular co-witness domain (if the index $prek$ is omitted) or a verifier. In each case, the meaning of V will be clear from the context.

EXTENDEDNESS

For all prekeys $prek \in [genPrekey(k)]$, all inputs $w \in W$, $v \in V$ and candidates z, z' such that $make'(z, v) = z'$ the following protocol equivalences hold:

$$proveExt([z, w], [P'[z, z', v], V[z']]) \text{ is equivalent to } prove^{(P^*, V^*)}([z], P^*[w]) \text{ .} \quad (4.3)$$

$$proveExt([z, w], [P'[z, z', v], V[z']]) \text{ is equivalent to } prove^{(P^*, V^*)}([z'], P^*[w + v]) \text{ .} \quad (4.4)$$

Actually, these are equivalences of protocol instances (see the remark after Definition 3.9 on p.20).

CO-UNLINKABILITY

An extended *proof-of-knowledge scheme* (with operation $proveExt$) is co-unlinkable, if there is a constant $\kappa \in \mathbb{N}$, which may depend on the prekey of the scheme, such that the following holds for each two attackers \tilde{P} and \tilde{V} in place of the prover and verifier of $proveExt$ and for all candidates $z, z' \in Z$ and co-witnesses $v_1, v_2 \in V$ such that $make'(z, v_1) = make'(z, v_2) = z'$.

If $view_{\tilde{P}}^{(z, v_1, z')}$ is a valid view of the prover \tilde{P} originating from an execution of $proveExt$ on inputs z, v_1, z' , and $view_{\tilde{V}}^{(z, v_2, z')}$ is a valid view of the verifier \tilde{V} originating from another execution of $proveExt$ on inputs z, v_2, z' , then for each co-witness v such that $make'(z, v) = z'$ there are exactly κ internal choices ρ_i ($i \in [1, \kappa]$) of the co-prover such that \tilde{P} and \tilde{V} obtain respective views $view_{\tilde{P}}$ and $view_{\tilde{V}}$ in the same execution of $proveExt$ on inputs z, v, z' and where the co-prover uses any of the internal choices ρ_i . Compare to motivation (2) on p.40. \blacklozenge

Remark: Note that this definition of unlinkability accepts the possibility of an attacking prover or attacking verifier to abort executions of $proveExt$ according to certain patterns they have agreed on in advance. This way, they can transfer one bit of information in either direction per execution. Formally, the definition ignores this problem, because as soon as either \tilde{P} or \tilde{V} aborts the protocol, the resulting view is invalid by Definition 3.11 on p.21, but only pairs of valid views on $proveExt$ are considered. We justify this definition because any such protocol abortion can be detected by the honest co-prover, who can then suspend any further interaction with such suspicious prover or verifier.

4.1.5 Related Work on Proofs-of-Knowledge

We shortly review diverted proofs of knowledge as defined by Okamoto and Ohta in [173]. Their starting question was which class of 2-party proofs-of-knowledge can be diverted by a co-prover such that neither the prover nor the verifier can distinguish talking to each other from talking to the co-prover diverting each other's messages. If the indistinguishability holds for polynomial-time provers and verifiers only, the divertibility is called computational, otherwise perfect. A practical sufficiency criterion for perfect divertibility was given by Blaze, Bleumer, Strauss [22, 20].

In terms of Definition 4.3 on p.41, the set of co-witnesses of divertible proof-of-knowledge schemes is empty and, therefore, the candidate z shared by the prover and the co-prover is always equal to the candidate z' shared by the co-prover and the verifier ($z = z'$). Co-unlinkability for extended proof-of-knowledge protocols is analogous to perfect divertibility of diverted proof-of-knowledge protocols.

4.1.6 Chaum-Pedersen-Van De Graaf Proof-of-Knowledge Mechanism

We recall a proof of knowledge protocol by Chaum, Evertse, van de Graaf [68]. In the simplest version, the witness domain is \mathbb{Z}_q and the candidate domain is G_q , where (\mathbb{Z}_p, G_q) is a discrete log setting. The making function is $make(w) = g^w \bmod p$, where g is some generator of G_q . We propose an extended proof protocol that uses the witness domain $W = \mathbb{Z}_q^\ell$ ($\ell \in \mathbb{N}$), the co-witness domain $V = \mathbb{Z}_q \times W$ and the following addition of witnesses and co-witnesses (note that the plus sign on the right hand side denotes tuple addition in \mathbb{Z}_q^ℓ):

$$w + v = w + (\omega, u) = u + \omega w \bmod q .$$

The special case where $\omega \equiv 1$ is fixed was proposed by Brands [35] building on ideas of Okamoto, Ohta [172]. We name the scheme after the original inventors.

Mechanism 4.4 CEG(ℓ) Mechanism

Consider the discrete log framework (Definition 3.17 on p.28) and let $\ell \in \mathbb{N}$ be some (usually small) integer constant. The domains and operations of the CEG(ℓ) Mechanism are as follows.

GENERATE PREKEY $(p, q, \vec{g}) \leftarrow genPrekey(k)$

Pick a discrete log setting (p, q) uniformly at random from dls_k (Definition 3.17 on p.28). Then pick a tuple $\vec{g} = (g_1, \dots, g_\ell)$ of ℓ generators of G_q (Observation 3.18 on p.28).

DOMAINS

The candidate domains are $Z_{prek} = G_q$, and the witness domains are $W_{prek} = \mathbb{Z}_q^\ell$. The making function is defined as follows:

$$make(\vec{w}) = z = \prod_{j=1}^{\ell} g_j^{w_j} \bmod p .$$

Note that the witnesses \vec{w} of a candidate z are precisely its representations with respect to \vec{g} .

OPERATIONS

Prove

$$V_{[accept]} \leftarrow prove_{g_1, \dots, g_\ell}([z], P_{[\vec{w}]})$$

<i>prove</i>	Prover		Verifier
(1)	Choose $\vec{s} \in_R \mathbb{Z}_q^\ell$		Choose $c \in_R \mathbb{Z}_q$
(2)	$a \leftarrow make(\vec{s})$	\xrightarrow{a}	
(3)		\xleftarrow{c}	
(4)	$\vec{r} \leftarrow c\vec{w} + \vec{s} \bmod q$	$\xrightarrow{\vec{r}}$	
(5)			$accept \leftarrow (az^c = make(\vec{r}))$

FIGURE 4-2 Chaum-Evertse-Van De Graaf Proof of Knowledge: *prove*

The prover chooses uniformly at random a witness \vec{s} from W (step (1)), computes the corresponding candidate a and sends the result to the verifier (step (2)). The verifier chooses uniformly at random a challenge c from \mathbb{Z}_q (step (1)) and returns it to the prover (step (3)). Finally the prover forms the linear combination $\vec{r} \leftarrow \vec{s} + c\vec{w} \pmod q$ and sends it to the verifier (step (4)). The verifier accepts the proof iff $az^c = \text{make}(\vec{r})$ (step (5)). \blacklozenge

Lemma 4.1 Homomorphy of Making Function

For all $\ell \in \mathbb{N}$ and all prekeys $(\mathbb{Z}_p, G_q, \vec{g}) \in [\text{genPrekey}(\mathbb{N})]$, the function $\text{make}: W \rightarrow Z$ of $\text{CEG}(\ell)$ is a (vector space) epimorphism. For all witnesses $v, w \in W$ and $\alpha, \beta \in \mathbb{Z}_q$ the following equation holds:

$$\text{make}(\alpha\vec{v} + \beta\vec{w}) = \text{make}(\vec{v})^\alpha (\text{make}(\vec{w}))^\beta \pmod p .$$

Proof

This is immediate from the construction of make :

$$\text{make}(\alpha\vec{v} + \beta\vec{w}) = \prod_{j=1}^{\ell} g_j^{\alpha v_j + \beta w_j} = \left(\prod_{j=1}^{\ell} g_j^{v_j} \right)^\alpha \left(\prod_{j=1}^{\ell} g_j^{w_j} \right)^\beta = \text{make}(\vec{v})^\alpha (\text{make}(\vec{w}))^\beta \pmod p . \quad \square$$

Proposition 4.1 CEG(ℓ)

Under the SR-Assumption, $\text{CEG}(\ell)$ is a proof-of-knowledge scheme for all $\ell \in \mathbb{N}$. \blacklozenge

Proof

COMPLETENESS

The verifier is convinced whenever $z = \text{make}(\vec{w})$. This follows from basic rewritings:

$$\begin{aligned} az^c &= \text{make}(\vec{s})(\text{make}(\vec{w}))^c && \text{by step (1) and presumption} \\ &= \text{make}(\vec{s} + c\vec{w}) && \text{by Lemma 4.1 on p.44} \\ &= \text{make}(\vec{r}) \pmod p && \text{by step (4).} \end{aligned}$$

VALIDITY

Now we need to consider an arbitrary prover \tilde{P} and all candidates $z \in Z$ on which \tilde{P} is expected to convince the verifier with probability $p(z) > 0$. We construct a universal knowledge extractor $E_{\text{CEG}(\ell)}$ as follows. $E_{\text{CEG}(\ell)}$ runs the prover \tilde{P} through the protocol *prove* (Figure 4-2 on p.43) so many times until the prover has convinced $E_{\text{CEG}(\ell)}$ by a proper response \vec{r}_1 in step (5). Let a denote \tilde{P} 's message after step (2) and c_1 denote the challenge sent to \tilde{P} after step (3). Afterwards, $E_{\text{CEG}(\ell)}$ rewinds \tilde{P} back into step (3), thereby re-using the message a , and runs \tilde{P} down to the end of *prove*. $E_{\text{CEG}(\ell)}$ continues rewinding and running \tilde{P} until \tilde{P} delivers a second convincing response \vec{r}_2 in step (5). Let c_2 denote the challenge corresponding to \vec{r}_2 . The extractor will make sure to use a different challenge after each rewinding, to make sure that $c_1 \neq c_2 \pmod q$. The extractor finds the following two conditions satisfied:

$$az^{c_1} = \text{make}(\vec{r}_1) \quad \text{and} \quad az^{c_2} = \text{make}(\vec{r}_2) .$$

Then follows $z^{c_1 - c_2} = \text{make}(\vec{r}_1 - \vec{r}_2)$ and so a witness of z is found: $w = \frac{\vec{r}_1 - \vec{r}_2}{c_1 - c_2} \pmod{q}$. (4.5)

Thus, $E_{CEG(\ell)}$ needs expected $1/p(z)$ tries to get a first convincing response \vec{r}_1 in step (5). After each rewinding, the prover \tilde{P} starts execution of *prove* in step (3), and therefore the probability $p^*(z)$ to convince the verifier is the probability $p(z)$ to convince the verifier in a complete execution of *prove* under the condition that \tilde{P} sends message a after step (2). In general, we have no information about the probability $p_a(z)$ that \tilde{P} sends a after step (2) but the trivial bounds $0 < p_a(z) < 1$. Thus we can put a lower bound on $p^*(z) = p(z)/p_a(z) \geq p(z)$, which means $E_{CEG(\ell)}$ needs expected $1/p^*(z) \leq 1/p(z)$ tries to get a second convincing response \vec{r}_2 in step (5). Altogether, $E_{CEG(\ell)}$ is expected to find two successful responses, and thereby a witness, after challenging the prover \tilde{P} after

$$\frac{1}{p(z)} + \frac{1}{p^*(z)} \leq \frac{2}{p(z)}$$

executions of *prove* and finally computing the quotient (4.5). Each execution of *prove* costs the universal knowledge extractor mainly $\ell + 1$ modular exponentiations (step (5)), each computable in $O(k^3)$ elementary steps. This is a very generous upper bound because modular multiplication of k -bit factors is faster than $O(k^2)$ and the exponents c, w_1, \dots, w_ℓ occurring in step (5) are in \mathbb{Z}_q , i.e., of length around 160 bit, which is less than a fifth of k . We thus expect the universal knowledge extractor to find a witness in expected $(\ell + 1) \cdot O(k^3) = O(k^3)$ elementary steps:

$\forall ITM \tilde{P} \exists k_0 \forall k > k_0, \text{prek} \in [\text{genPrekey}(k)], z \in Z, Q :$

$$\text{if } p(z) > 0 \text{ then } \text{ExpTime}[(\text{make}(\vec{w}) = z) \setminus \vec{w} \leftarrow E_{CEG(\ell)}(z, \tilde{P}(z, Q))] < \frac{1}{p(z)} k^3 .$$

Obviously, this bound satisfies Definition 4.1 on p.35 (validity) for $c = 3$. □

Proposition 4.2 CEG(ℓ)

Protocol *prove* of CEG(ℓ) is perfectly witness indistinguishable for all $\ell > 1$. ◆

Proof

The functions $\text{make}(\vec{w})$ for $\vec{w} \in W$ are a family (with index *prek*) of one-way functions because they are hard to invert according to the SR-Assumption 3.21 on p.30.

We show that for each common input $z \in Z$ protocol *prove* produces verifier views (a, c, \vec{r}) with the same probability distribution regardless which witness of z the prover uses (see Definition 4.2 on p.38). Because $\ell > 1$, we find that for each candidate z there are $q^{\ell-1} \geq q > 1$ witnesses $\vec{w}_j \in W_{\text{prek}}$ ($j \in [1, q^{\ell-1}]$) such that $z = \text{make}(\vec{w}_j)$. We consider any one of these witnesses $\vec{w} = \vec{w}_j$ and any one valid view (a, c, \vec{r}) of the verifier in *prove*, i.e., one that satisfies the verifier's accepting condition in protocol *prove* step (5):

$$az^c = \text{make}(\vec{r}) \pmod{p} . \quad (4.6)$$

We find that there is exactly one internal choice \vec{s} for the prover in protocol *prove* that produces the given view (a, c, \vec{r}) if the prover takes as input the witness $\vec{w} = (w_1, \dots, w_\ell)$ and the candidate z : The prover's internal choice $\vec{s} = (s_1, \dots, s_\ell)$ is determined by a system of linear equations according to protocol *prove* step (4), namely:

$$cw_i + s_i = r_i \pmod{q} \quad \text{for all } i = 1, \dots, \ell . \quad (4.7)$$

Next we need to see, if this uniquely determined internal choice \vec{s} also matches with how the internal choice \vec{s} is determined by protocol *prove* step (2): We evaluate the expression $make(\vec{s})$

$$make(\vec{s}) = \frac{make(\vec{r})}{make(c\vec{w})} = \frac{make(\vec{r})}{(make(\vec{w}))^c} = \frac{make(\vec{r})}{z^c} = a \pmod{p} ,$$

by using from left to right: the assertion (4.7), the homomorphy of *make* (Lemma 4.1 on p.44), the fact that $z = make(\vec{w})$ and finally the assertion (4.6). This matches with protocol *prove* step (2). In other words, no valid view of a verifier gives any information about which witness the prover uses. \square

4.1.7 Brands Proof-of-Knowledge Mechanism

Next we recall a proof-of-knowledge mechanism by Brands [38] by which a verifier can verify that the prover knows a CEG(ℓ) witness of a candidate and that the witness satisfies a particular linear relation, namely that the witness component w_1 referring to the first generator does not disappear. For example for $\ell = 2$, Brands Mechanism can be used to verify that the prover knows a representation (α_1, α_2) of a candidate $z = g_1^{\alpha_1} g_2^{\alpha_2} \pmod{p}$ with $\alpha_1 \neq 0$. Hence, the verifier is convinced, that the prover does not know the discrete log of z with respect to g_2 (whereas the prover may know the discrete log of z with respect to g_1). Brands has proposed efficient protocols for the general case of proving that the witness satisfies any Boolean formula of linear relations over the witness components.

Mechanism 4.5 Brands(ℓ) Mechanism

The Brands(ℓ) Mechanism for some (usually small) integer constant $\ell \in \mathbb{N}$ is defined by the same prekey generating algorithm and the same domains as the CEG(ℓ) Mechanism except that the making function is partial because it is defined only for arguments $w \in W$ such that $w_1 \neq 0$:

$$make(\vec{w}) = \prod_{j=1}^{\ell} g_j^{w_j} \pmod{p} .$$

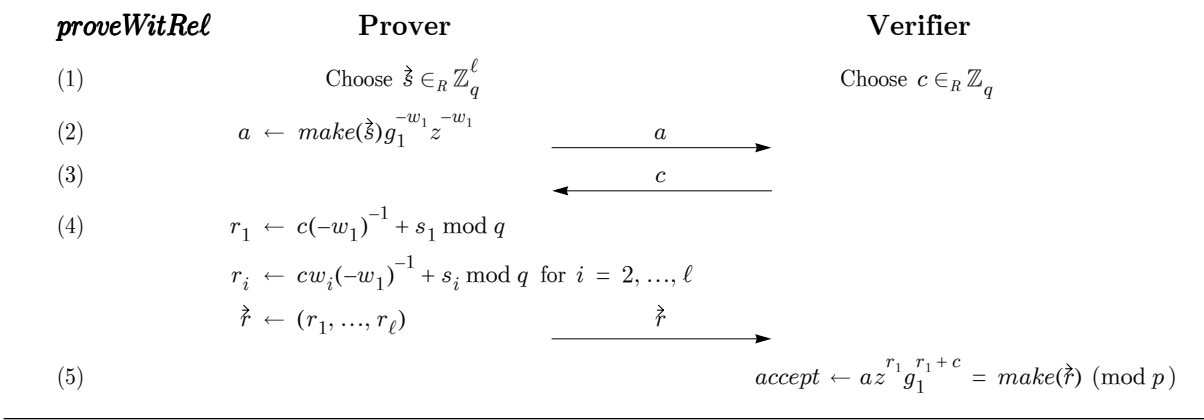
The protocol *proveWitRel* is redefined as follows:

ProveWitRel

Prover and verifier take a candidate $z = make(\vec{w})$ as common input, where $\vec{w} = (w_1, \dots, w_\ell)$ is the prover's private input such that $w_1 \neq 0$. The prover chooses uniformly at random a witness \vec{s} from W (step (1)), computes the corresponding candidate a and sends the result to the verifier (step (2)). The verifier chooses uniformly at random a challenge c from \mathbb{Z}_q (step (1)) and returns it to the prover (step (3)). The prover computes the components r_1, \dots, r_ℓ according to (step (4)). Note that the expression $(-w_1)^{-1}$ is defined because $w_1 \neq 0$. The verifier accepts in the proof step (5) iff $az^{r_1} g_1^{r_1+c} = make(\vec{r}) \pmod{p}$, which can be re-written as

$$az^{r_1} g_1^c = \prod_{i=2}^{\ell} g_i^{r_i} \pmod{p} . \quad \blacklozenge$$

$$V[\text{accept}] \leftarrow \text{proveWitRel}_{g_1, \dots, g_\ell}^{[w_1 \neq 0]}([z], P, [\vec{w}])$$


 FIGURE 4-3 Brands Proof of Knowledge: *proveWitRel*

Proposition 4.3 Brands(ℓ)

Under the SR-Assumption, Brands(ℓ) is a proof-of-knowledge scheme for all $\ell \in \mathbb{N}$. That is, whenever the verifier is convinced on common input a candidate z , then the prover knows a witness \vec{w} with $w_1 \neq 0$ that makes the candidate z . Protocol *proveWitRel* of Brands(ℓ) is perfectly witness indistinguishable for all $\ell > 1$. \blacklozenge

A proof is given by Brands in [38].

4.1.8 Extended Chaum-Pedersen-Van De Graaf Proof-of-Knowledge Mechanism

We present an extended proof-of-knowledge Mechanism based on the CEG(ℓ) Mechanism 4.4 on p.43.

Mechanism 4.6 Extended CEG(ℓ) Mechanism (ECEG(ℓ))

The Extended CEG(ℓ) Mechanism (ECEG(ℓ)) employs the same prekey generating algorithm and all the domains of the CEG(ℓ) Mechanism and the following additional domains and operations:

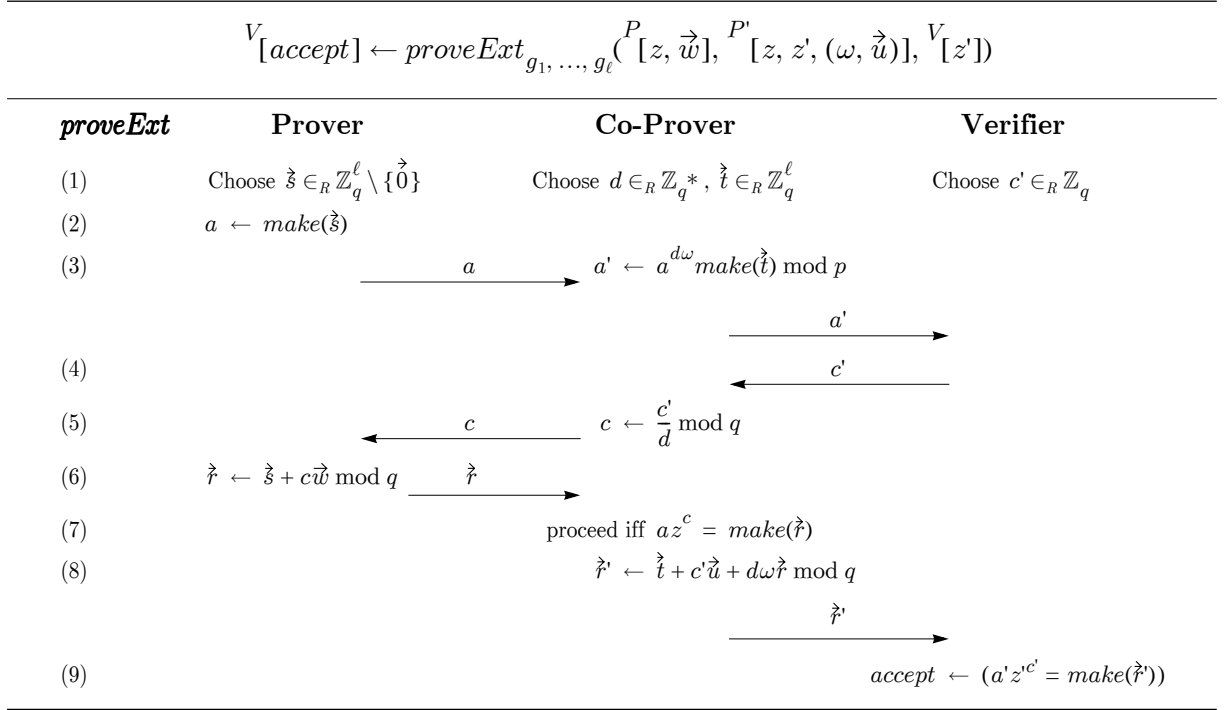
DOMAINS

- The co-witness domains are $V_{\text{prek}} = \mathbb{Z}_q \times W_{\text{prek}}$.
- The addition of witnesses and co-witnesses is defined as $\vec{w} + (\omega, \vec{u}) \stackrel{\text{Def}}{=} \vec{u} + \omega\vec{w} \bmod q$.
- The co-making homomorphisms are $\text{make}'(z, (\omega, \vec{u})) = z^\omega \text{make}(\vec{u}) \bmod p$.

OPERATIONS

ProveExt

The prover chooses a temporary witness \vec{s} uniformly at random (step (1)), makes the corresponding temporary candidate a and sends it to the co-prover (step (2)). The co-prover P' also chooses a challenge $d \in_R \mathbb{Z}_q^*$ and a temporary witness \vec{t} uniformly at random (step (1)), constructs the corresponding temporary candidate a' (step (2)). The co-prover sends a' to the verifier V (step (3)). V chooses the challenges c' uniformly at random (step (1)), and returns it to the co-prover (step (4)). The co-prover forms a new challenge c and passes it to the prover (step (5)). The prover forms the linear


 FIGURE 4-4 Extended Proof of Knowledge: *proveExt*

combination $\vec{r} = \vec{s} + c\vec{w}$ (step (6)) and sends it to the co-prover. If the co-prover finds the condition in step (7) satisfied, then he constructs his final message $\vec{r}' = \vec{t} + c'\vec{u} + d\omega\vec{r}$ and sends it to the verifier (step (8)). V accepts if the condition in step (9) is satisfied. \blacklozenge

Lemma 4.2 Properties of the Co-Making Function

For all $\ell \in \mathbb{N}$, all prekeys $(\mathbb{Z}_p, G_q, \vec{g}) \in [\text{genPrekey}(\mathbb{N})]$ of $ECEG(\ell)$, all witnesses $\vec{w} \in W_{\text{prek}}$, all co-witnesses $(\omega, \vec{u}) \in V_{\text{prek}}$, all $\omega_0 \in \mathbb{Z}_q$ and all $\psi \in G_q$ the following assertions hold:

$$1.) \quad \text{make}'(\text{make}(\vec{w}), (\omega, \vec{u})) = \text{make}(\vec{u} + \omega\vec{w}) \quad ,$$

$$2.) \quad (\text{make}'(\psi, (\omega, \vec{u})))^{\omega_0} = \text{make}'(\psi, (\omega_0\omega, \omega_0\vec{u})) \quad . \quad \blacklozenge$$

Proof

Both assertions follow from basic rewritings:

$$\text{make}'(\text{make}(\vec{w}), (\omega, \vec{u})) = \left(\prod_{j=1}^{\ell} g_j^{w_j} \right)^\omega \prod_{j=1}^{\ell} g_j^{u_j} = \prod_{j=1}^{\ell} g_j^{u_j + \omega w_j} = \text{make}(\vec{u} + \omega\vec{w}) \quad .$$

$$(\text{make}'(\psi_O, (\omega, u)))^{\omega_0} = \left(\psi_O^\omega \prod_{j=1}^{\ell} g_j^{u_j} \right)^{\omega_0} = \psi_O^{\omega_0\omega} \prod_{j=1}^{\ell} g_j^{\omega_0 u_j} = \text{make}'(\psi_O, (\omega_0\omega, \omega_0 u)) \quad \square$$

Security Suggestion 4.4 $ECEG(\ell)$

Under the SDL-Assumption, $ECEG(\ell)$ is an extended proof-of-knowledge scheme. \blacklozenge

Security Consideration

We check condition (4.2) first. Let $z, (\omega, \vec{u}), z'$ be such that the premise (*) $z' = \text{make}'(z, (\omega, \vec{u})) = z^\omega \text{make}(\vec{u}) \pmod{p}$ in (4.2) is satisfied, then we have to show that

$$(\text{make}(\vec{w}) = z) \Leftrightarrow (\text{make}(\vec{u} + \omega\vec{w}) = z') .$$

\Rightarrow : If \vec{w} is such that $\text{make}(\vec{w}) = z$ then we get:

$$\begin{aligned} \text{make}(\vec{u} + \omega\vec{w}) &= (\text{make}(\vec{w}))^\omega \text{make}(\vec{u}) && \text{by Lemma 4.1 on p.44} \\ &= z^\omega \text{make}(\vec{u}) && \text{by way how } \vec{w} \text{ is given} \\ &= z' \pmod{p} . && \text{by premise (*) about } z' \end{aligned}$$

\Leftarrow : If \vec{w} is such that $\text{make}(\vec{u} + \omega\vec{w}) = z'$ then we get:

$$\begin{aligned} \text{make}(\vec{w}) &= (\text{make}(\vec{w})^\omega \text{make}(\vec{u}) \text{make}(-\vec{u}))^{1/\omega} && \text{expand expression} \\ &= (\text{make}(\omega\vec{w} + \vec{u}) \text{make}(-\vec{u}))^{1/\omega} && \text{by Lemma 4.1 on p.44} \\ &= (z' \cdot \text{make}(-\vec{u}))^{1/\omega} && \text{by way how } \vec{w} \text{ is given} \\ &= (z^\omega \text{make}(\vec{u}) \cdot \text{make}(-\vec{u}))^{1/\omega} && \text{by premise (*) about } z' \\ &= z \pmod{p} . && \text{collapse expression} \end{aligned}$$

Next, we verify the two requirements on an extended proof-of-knowledge mechanism according to Definition 4.3 on p.41.

EXTENDEDNESS

We argue that for all prekeys $\text{prek} \in [\text{genPrekey}(k)]$, all inputs $w \in W$, $v \in V$ and candidates z, z' such that $\text{make}'(z, v) = z'$ the subprotocol instance

$$\text{proveExt}_{g_1, \dots, g_\ell} \left([z, w], [{}^P [z, z', v], {}^V [z']] \right) \quad (4.8)$$

of *proveExt* (Figure 4-4 on p.48) between the prover P on the one hand and the composed verifier $P' ||| V$ on the other hand is equivalent (Definition 3.9 on p.20) to the protocol instance $\text{prove} \left({}^{P^*} [z, w], {}^{V^*} [z] \right)$ (Figure 4-2 on p.43). In order to conclude this equivalence we show that on respective inputs the two provers are equivalent, and the two verifiers are equivalent, too:

- (i) The provers P and P^* are equivalent because they are equal ITMs.
- (ii) The composed verifier $P' ||| V$ and the verifier V^* in both protocols are equivalent because they both send a randomly chosen challenge $c \in \mathbb{Z}_q$ in step (3) of *prove* and step (4) of *proveExt*, respectively, and they both accept at the end of the respective protocol if and only if they find the condition (***) $az^c = \text{make}(\vec{r}) \pmod{p}$ satisfied. For the verifier V^* in protocol *prove*, its condition in step (5) of Figure 4-2 on p.43 literally matches condition (***). For the composed verifier $P' ||| V$ of *proveExt*, its condition in step (9) of Figure 4-4 on p.48 can be rewritten into condition (***) as follows:

4 CRYPTOGRAPHIC TOOLS

By presumption, the input to $P' ||| V$ is (\ddagger) : $z' = \text{make}'(z, (\omega, \vec{u})) = z^\omega \text{make}(\vec{u}) \pmod p$. Thus:

$$\begin{aligned}
& a' z'^{c'} = \text{make}(\vec{r}) \pmod p \\
\Rightarrow & a^{d\omega} \cdot \text{make}(\vec{t}) \cdot z'^{c'} = \text{make}(\vec{t} + c'\vec{u} + d\omega\vec{r}) \pmod p && \text{step (3), step (8)} \\
\Rightarrow & a^{d\omega} \cdot \text{make}(\vec{t}) \cdot z'^{c'} = \text{make}(\vec{t})\text{make}(c'\vec{u})\text{make}(d\omega\vec{r}) \pmod p && \text{Lemma 4.1 on p.44} \\
\Rightarrow & a^{d\omega} \cdot (z^\omega \text{make}(\vec{u}))^{c'} = \text{make}(c'\vec{u})\text{make}(d\omega\vec{r}) \pmod p && \text{cancel } \text{make}(\vec{t}), \text{presumption } (\ddagger) \text{ for } z' \\
\Rightarrow & a^{d\omega} \cdot z^{\omega c d} \text{make}(c'\vec{u}) = \text{make}(c'\vec{u})\text{make}(d\omega\vec{r}) \pmod p && \text{step (5)} \\
\Rightarrow & a^{d\omega} \cdot z^{\omega c d} = (\text{make}(\vec{r}))^{d\omega} \pmod p && \text{cancel } \text{make}(c'\vec{u}), \text{Lemma 4.1 on p.44} \\
\Rightarrow & a \cdot z^c = \text{make}(\vec{r}) \pmod p . && \omega, d \neq 0 \pmod q
\end{aligned}$$

Thus, the composed verifier $P' ||| V$ on input (z, z', v) is equivalent to the ITM V^* on input (z) .

Likewise, we show that for all prekeys $\text{prek} \in [\text{genPrekey}(k)]$, all inputs $w \in W$, $v \in V$ and candidates z, z' such that $\text{make}'(z, v) = z'$ the subprotocol instance

$$\text{proveExt}_{g_1, \dots, g_\ell}([z, \vec{w}], [z, z', v], [z']) \quad (4.9)$$

of proveExt (Figure 4–4 on p.48) between the composed prover $P ||| P'$ on the one hand and verifier V on the other hand is equivalent (Definition 3.9 on p.20) to the protocol instance $\text{prove}^{P^*}([z', \vec{w} + v], [z'])$ (Figure 4–2 on p.43). We show on respective inputs that the two provers are equivalent, and the two verifiers are equivalent, too:

- (i) First consider the composed prover $P ||| P'$ on input (z, \vec{w}, z', v) , where $v = (\omega, \vec{u})$, and internal choices \vec{s}, d, \vec{t} (see step (1) of Figure 4–4 on p.48). The same messages that this composed prover sends are also produced by the prover P^* on input $(z', \vec{w} + v)$ and internal choice $d\omega\vec{s} + \vec{t}$ (see step (1) of Figure 4–2 on p.43). We show this for the two messages a' (step (3) of Figure 4–4 on p.48) and r' (step (8) in turn:

$$\begin{aligned}
a' &= a^{d\omega} \text{make}(\vec{t}) && \text{by step (3) of } \text{proveExt} \\
&= (\text{make}(\vec{s}))^{d\omega} \text{make}(\vec{t}) && \text{by step (2) of } \text{proveExt} \\
&= \text{make}(d\omega\vec{s} + \vec{t}) . && \text{by Lemma 4.1 on p.44} \\
r' &= \vec{t} + c'\vec{u} + d\omega\vec{r} && \text{by step (8) of } \text{proveExt} \\
&= \vec{t} + c'\vec{u} + d\omega(\vec{s} + c'\vec{w}) && \text{by step (6) of } \text{proveExt} \\
&= \vec{t} + d\omega\vec{s} + c'\vec{u} + c'\omega\vec{w} && \text{by step (5) of } \text{proveExt} \\
&= (d\omega\vec{s} + \vec{t}) + c'(\vec{w} + v) . && \text{by definition of witness addition}
\end{aligned}$$

For each given input $v = (\omega, \vec{u})$, the composed prover $P ||| P'$ chooses \vec{s}, d, \vec{t} uniformly at random from their respective domains. Hence, the random variable $d\omega\vec{s} + \vec{t}$ is uniformly distributed,

just like the random choice of the prover P^* . Thus, the composed prover $P ||| P'$ on input (z, \vec{w}, z', v) is equivalent to the ITM P^* on input $(z', \vec{w} + v)$.

(ii) The verifiers V and V^* are equivalent because they are equal ITMs.

CO-UNLINKABILITY

For each $\ell \in \mathbb{N}$, consider any two valid views $view_{\tilde{P}} = (\rho, a, c, \vec{r})$ of \tilde{P} on P' and $view_{\tilde{V}} = (\rho', a', c', \vec{r}')$ of \tilde{V} on P' resulting from two executions of *proveExt* on the same shared inputs z, z' and two arbitrary private inputs $(\omega_1, \vec{u}_1), (\omega_2, \vec{u}_2)$ such that $z' = make'(z, (\omega_1, \vec{u}_1)) = make'(z, (\omega_2, \vec{u}_2))$.⁴ So the co-prover is presumed not to abort in step (7), i.e., $az^c = make(\vec{r})$, and it always sends \vec{r}' after step (8) such that $a'z'^{c'} = make(\vec{r}')$.

We show that for each given co-witness (ω, \vec{u}) such that $z' = make'(z, (\omega, \vec{u}))$, there is exactly one internal choice (d, \vec{t}) of the co-prover such that the co-prover matches both views $view_{\tilde{P}}$ and $view_{\tilde{V}}$ when it executes *proveExt* on common input z, z' , private input (ω, \vec{u}) and uses the internal choice (d, \vec{t}) .

“ ≤ 1 ”: From step (5) follows that there is one unique choice $d = c'/c \bmod q$ because the challenges c, c' are part of the given views $view_{\tilde{P}}$ and $view_{\tilde{V}}$. Then follows from step (8) that there is one unique choice $\vec{t} = \vec{r}' - c'\vec{u} - d\omega\vec{r} \bmod q$.

“ ≥ 1 ”: We are left to show that if the co-prover uses its co-witness (ω, \vec{u}) and the internal choice (d, \vec{t}) determined above, then he will also meet the remaining messages a, a' of the given views $view_{\tilde{P}}$ and $view_{\tilde{V}}$, respectively. We verify the co-prover’s computation in step (3) as follows:

$$\begin{aligned}
 a^{d\omega} make(\vec{t}) &= (z^{-c} make(\vec{r}))^{d\omega} make(\vec{t}) && \text{insert } z^{-c} make(\vec{r}) \text{ for } a \text{ by presumption} \\
 &= z^{-cd\omega} make(d\omega\vec{r}) make(\vec{r}' - c'\vec{u} - d\omega\vec{r}) && \text{insert } \vec{r}' - c'\vec{u} - d\omega\vec{r} \text{ for } \vec{t} \\
 &= z^{-c'\omega} make(\vec{r}' - c'\vec{u}) && \text{insert } \frac{c'}{c} \text{ for } d \\
 &= (z^\omega make(\vec{u}))^{-c} make(\vec{r}') && \text{re-arrange terms using homomorphism of } make \\
 &= z'^{-c} make(\vec{r}') && \text{insert } z' \text{ for } z^\omega make(\vec{u}) \text{ by presumption} \\
 &= a' \pmod{p} . && \text{insert } a' \text{ for } z'^{-c} make(\vec{r}') . \quad \square
 \end{aligned}$$

4.2 Blind Signature Schemes

In this section we recall *blind signatures* as introduced by Chaum [58, 59] and some important special cases which are of particular interest for privacy-oriented legitimation schemes. A blind signature scheme has a signing and a verifying operation (among others). The signing operation takes as input a signing key and a message from the signer Alice and returns a blinded message and corresponding signature to the recipient Bob. The key feature is that Bob alone obtains the signed message, not Alice.

A well-known formal framework for ordinary digital signatures is the *GMR-Definition* by Goldwasser, Micali and Rivest [122]. It is of limited use for blind signatures, because their notions of

4) The component ρ of $view_{\tilde{P}}$ and ρ' of $view_{\tilde{V}}$ denote respective internal choices of the attackers \tilde{P} and \tilde{V} .

unforgeability do not apply to blind signatures. According to the GMR-Definition there is only one way of producing valid signatures, namely, an algorithm that takes as input a signing key and a message and returns a signature for that message. Therefore, any message whose signature is not forged is learned by the signer. The GMR-Definition identifies 4 types of forgery against active and passive attackers. Ordered by decreasing severity, the types of forgery are (i) to figure out the signing key (*total break*), or (ii) any equivalent key, which also produces valid signatures (*universal break*), or (iii) to find a signature for any given message (*selective forgery*) or (iv) for any message whatsoever (*existential forgery*). An attacker may be restricted to access the verifying key and a number of signed messages (passive attack), or he may also be allowed to ask the signer for signatures on messages of his choice (active attack). The strongest signature scheme is secure against the weakest type of forgery under the strongest attacker model, i.e., existential forgery against active attackers. A general framework for such signature schemes, including fail-stop signatures, has been presented by Pfitzmann [183].

In contrast, blind signature schemes allow recipients to obtain signatures for messages that the signer never learns. The notions of total break and universal break apply here as well. However the notions of selective and existential forgery are pointless for active attacks, because they intentionally let the recipient obtain a signature for a new message that the signer has not seen before. For blind signatures one is interested in other notions of unforgeability, namely *one-timeness* and *restrictiveness*. We will see that one-timeness implies security against existential forgery under passive attacks (see the remark following Definition 4.7 on p.53). One-time blind signatures have attracted attention since Chaum, Fiat, Naor [69] and Chaum [63] used them to build practical offline and online untraceable electronic cash schemes. Most if not all electronic cash schemes employ one-time blind signatures, where Bob can obtain only one signed message from each interaction with Alice. This helps to avoid the problem of counterfeiting electronic coins [48,43,49,217,52]. The more fundamental problem inherent in any electronic coin system is to avoid multiple undetected copies of an issued coin. This replay problem is usually solved by tracking the unique serial number of each coin spent, where the serial number is contained in the blinded message. Propelled by the popularity of untraceable electronic cash, formal definitions of one-time blind signatures have been proposed by Franklin, Yung [113] and by Pointcheval, Stern [191,192,193].

For offline untraceable electronic cash, double spending of coins should be detectable after the fact. So double spenders should be identifiable if they use a coin more than once. Chaum, Pedersen [73], Brands [34], Ferguson [108,109], Frankel et al. [112] and Radu et al. [197,198] addressed this problem by using restrictive one-time blind signatures. A formal definition of a special type of restrictive blind signatures has been given by Pfitzmann and Sadeghi [186]. We present a more general formal definition of restrictive blind signatures in Section 4.2.1 on p.52. This is joint work with Birgit Pfitzmann. It is left as an open problem to make this definition even more general and applicable. See the last paragraphs of Section 5.7 on p.121.

4.2.1 Definition of Blind Signature Schemes

We now give a definition of one-time and restrictive blind signature schemes.

Definition 4.7 Blind Signature Scheme

A one-time restrictive blind signature scheme with security parameter k consists of the following prekey generator, domains, operations and requirements: $(genPrekey, genKey, sign, verify)$ satisfying the effectiveness and security requirements given afterwards.

GENERATE PREKEY $prek \leftarrow genPrekey(k)$

is a probabilistic polynomial-time algorithm that on input the *security parameter* $k \in \mathbb{N}$ outputs a *prekey* $prek$.

DOMAINS

A one-time restrictive blind signature scheme has a set $\{RK, PK, M, \Omega, \Sigma, W, make, E\}$ of 8 domain families whose respective members are:

- the *private* and a *public key* domains RK_{prek}, PK_{prek} ,
- the *message* domains M_{prek} ,
- the *blinder* domains Ω_{prek} ,
- the *signature* domains Σ_{prek} ,
- the *witness* domains W_{prek} ,
- the *making functions* $make_{prek}: W_{prek} \rightarrow M_{prek}$ that are efficient to evaluate. If $m = make(w)$, then we say that witness w *makes* message m .
- the *witness equivalence* relations $E_{prek} \subseteq W_{prek} \times W_{prek}$ that is efficient to test. If two witnesses w_1 and w_2 are equivalent according to the relation E_{prek} , then we simply write $w_1 \equiv w_2$. Witness equivalence is used to formalize restrictiveness.

In order to ease notation, we do not display the domain index $prek$ in the following.

OPERATIONS

A one-time restrictive blind signature scheme has a set $\{genPrekey, genKey, signBlind, verify\}$ of 4 operations satisfying 4 requirements: effectiveness, one-timeness, restrictiveness and blindness.

Generate Key $(rk, pk) \leftarrow genKey(prek)$

A probabilistic operation that takes as input a prekey $prek \in [genPrekey(k)]$ and returns a pair of corresponding private signing and public verifying keys $(rk, pk) \in RK \times PK$.

SignBlind ${}^B[m', \sigma'] \leftarrow signBlind([pk, m], {}^A[rk], {}^B[\{m^*\}, \omega])$

A probabilistic operation of a signer Alice and a recipient Bob. Common input is a verifying key $pk \in PK$ and a message $m \in M$. In addition, Alice takes a signing key $rk \in RK$, while Bob takes an optional private input $m^* \in M$ and a blinder $\omega \in \Omega$. Bob returns a message $m' \in M$ and a so-called *blind signature*⁵ $\sigma' \in \Sigma$.

Verify $accept \leftarrow verify(pk, m, \{m^*\}, \sigma)$

5) The term “blind signature” is rather intuitive and (probably therefore) well-established, but nevertheless misleading. “Blind signatures” as such are ordinary visible signatures. It is the message which is blinded and the signer who is “blind” during and after the signing operation.

4 CRYPTOGRAPHIC TOOLS

A deterministic operation that takes a public key $pk \in PK$, a message $m \in M$, an optional message $m^* \in M$ and a signature $\sigma \in \Sigma$ as input. It returns a Boolean output. If it is $accept = TRUE$, then we call σ *valid* for m with respect to public key pk , otherwise invalid. Pairs $(m, \sigma) \in M \times \Sigma$ are called *valid* with respect to pk iff σ is valid for m with respect to pk .

EFFECTIVENESS

For each correctly generated prekey $prek \in [genPrekey(k)]$, each correctly generated key pair $(rk, pk) \in [genKey(prek)]$, each message $m \in M$, and each blinder $\omega \in \Omega$: signing m interactively, ${}^B[m', \sigma'] \leftarrow signBlind([pk, m], {}^A[rk], {}^B[\{m^*\}, \omega])$, yields a message m' and a signature σ' valid for m' with respect to pk .

ONE-TIMENESS

The notion of one-timeness was suggested by Pointcheval and Stern [192]. A blind signature scheme is *one-time*, i.e., secure against one-more forgeries, if for all integers ℓ a polynomial-time attacker who has access to the verifying key of the signer can, after ℓ execution of $signBlind$ with the signer, find signatures for at most ℓ messages with non-negligible probability.

RESTRICTIVENESS

We follow ideas of Brands [34,35], Franklin, Yung [113] and Pfitzmann and Sadeghi [186]. A blind signature scheme is *restrictive* if a polynomial-time attacking recipient \tilde{B} has only a negligible chance to succeed in the following attack \mathcal{A} . After the signer has chosen a key pair $(rk, pk) \in_R [genKey(prek)]$ for some correctly generated prekey $prek \in [genPrekey(k)]$, the attacker performs the following three steps $n \in \mathbb{N}_0$ times for n bounded above by a polynomial in k :

- (1) for adaptively chosen messages m_1, \dots, m_n request from honest signer A respective blind signatures valid with respect to pk ,⁶
- (2) output n witnesses $w_1, \dots, w_n \in W$, and if $n > 0$ an additional witness $w' \in W$, and
- (3) output a pair (m', σ') of a message and a signature to the verifier.

The attack is successful if

- (i) the signature σ' is valid for m' and if $n > 0$ then the following two conditions also hold
- (ii) for all $i = 1 \dots n$, $make(w_i) = m_i$ and $make(w') = m'$, and
- (iii) the witness w' is not equivalent to any of the witnesses w_i ($i = 1 \dots n$).

BLINDNESS

A *signature scheme* is blind, if for each public key pk there is a $\kappa \in \mathbb{N}$, which may depend on the prekey and on pk , such that the following holds for a computationally unlimited attacking signer \tilde{A} in $signBlind$:

Let $view_{\tilde{A}} \in [view_{\tilde{B}}^{\tilde{A}}(signBlind([pk, m], {}^A[rk], {}^B[\omega]))]$ be a valid view (Definition 3.11 on p.21) of the attacking signer \tilde{A} in an execution of $signBlind$ on common input pk , private input rk

6) The term ‘adaptively chosen’, we mean that the attacker may choose each message depending on any previously chosen messages and signatures received.

of \tilde{A} and private input ω of honest recipient B . In addition, let (m', σ') be an arbitrary valid pair with respect to pk . Then there are exactly κ tuples (ω_i, ρ_i) ($i \in [1, \kappa]$) of blinders and internal choices for the recipient in *signBlind* such that the recipient outputs (m', σ') whenever \tilde{A} gets the view $view_{\tilde{A}}$ and the recipient takes input ω_i and makes the internal choice ρ_i .

OPTIONAL PRIVATE INPUT OF THE RECIPIENT

For each pair $(m, \sigma) \in M \times \Sigma$ of message $m \in M$ and signature $\sigma \in \Sigma$ valid with respect to a public key $pk \in PK$, a polynomial-time ITM has only negligible chance of finding two different optional messages m_1^*, m_2^* such that

$$verify(pk, m, \{m_1^*\}, \sigma) = verify(pk, m, \{m_2^*\}, \sigma) = TRUE \quad \blacklozenge$$

Remark: An immediate consequence of one-timeness is that passive attackers, who are allowed no interactions with the signer, cannot obtain signatures for any new messages. In other words, one-time blind signature schemes are secure against existential forgery under passive attacks.

Also a restrictive blind signature scheme is secure against universal forgery under an active attack. Would an active attacker with better than negligible probability figure a signing key by which it can produce valid signatures for any message, regardless how the messages are represented, then he had broken the restrictiveness property.

4.2.2 Chaum-Pedersen Signature Mechanism

We revisit the one-time restrictive blind signature scheme introduced by Chaum and Pedersen in [73], who built on ideas of Schnorr [208, 209]. Due to its inventors we refer to it as the Chaum-Pedersen Signature Mechanism. It will be used as a black box in the constructions of credential mechanisms in Section 5.4 on p.90 and Section 5.5 on p.110. It will also be the basis of the construction of blind group signature mechanisms in Section 6.4 on p.129.

Mechanism 4.8 Chaum-Pedersen(ℓ) Signature Mechanism

The domains and operations of the Chaum-Pedersen(ℓ) Signature Mechanism are as follows.

GENERATE PREKEY $(p, q, g, g_1, \dots, g_\ell, hash) \leftarrow genPrekey(k)$

Pick a discrete log setting (p, q) uniformly at random from dls_k (Definition 3.17 on p.28). Then pick $\ell + 1$ generators $g, g_1, \dots, g_\ell \in_R G_q$ (Observation 3.18 on p.28). Furthermore, choose a hash function $hash : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ that takes binary strings and returns elements of \mathbb{Z}_q^* . Typically, we use tuples (a, b, c) of numbers as input, and we then simply write $hash(a, b, c)$. The only assumption here is that a certain one-to-one conversion between the representations of tuples and binary strings is used. For example, for each security parameter k , the numbers can be represented as binary strings of fixed length, which in turn can be concatenated into larger strings suitable as input to *hash*. Alternatively, the numbers can be represented as binary strings of variable length by encoding each number in three fields: length of number | appropriate padding | number.

DOMAINS

The private and public key domains are $RK = \mathbb{Z}_g$ and $PK = G_q$, respectively. The message and signature domains are $M = G_q$ and $\Sigma = G_q \times G_q^2 \times \mathbb{Z}_q$, i.e., triples of components from $G_q, G_q^2,$

4 CRYPTOGRAPHIC TOOLS

and \mathbb{Z}_q , respectively. The blinder domains are $\Omega = \mathbb{Z}_q^*$. The witness domains are $W = \mathbb{Z}_q^\ell \setminus \{(\underbrace{0, \dots, 0}_{\ell \text{ zeroes}})\}$, the making functions are defined such that:

$$\text{make}(w) = \prod_{i=1}^{\ell} g_i^{w_i} \pmod{p}, \quad (4.10)$$

and the witness equivalences are defined such that

$$(v \equiv w) \Leftrightarrow (\forall \lambda, \mu \in [1, \ell] : v_\lambda w_\mu = v_\mu w_\lambda \pmod{q}) . \quad (4.11)$$

OPERATIONS

Generate Key

$$(rk, pk) \leftarrow \text{genKey}(prek)$$

Pick the private key $rk \in \mathbb{Z}_q$ uniformly at random and compute the public key $pk = g^{rk}$.

SignBlind

$${}^B[m', (z', a', b', r')] \leftarrow \text{signBlind}([pk, m], {}^A[rk], {}^B[\{m^*\}, \omega])$$

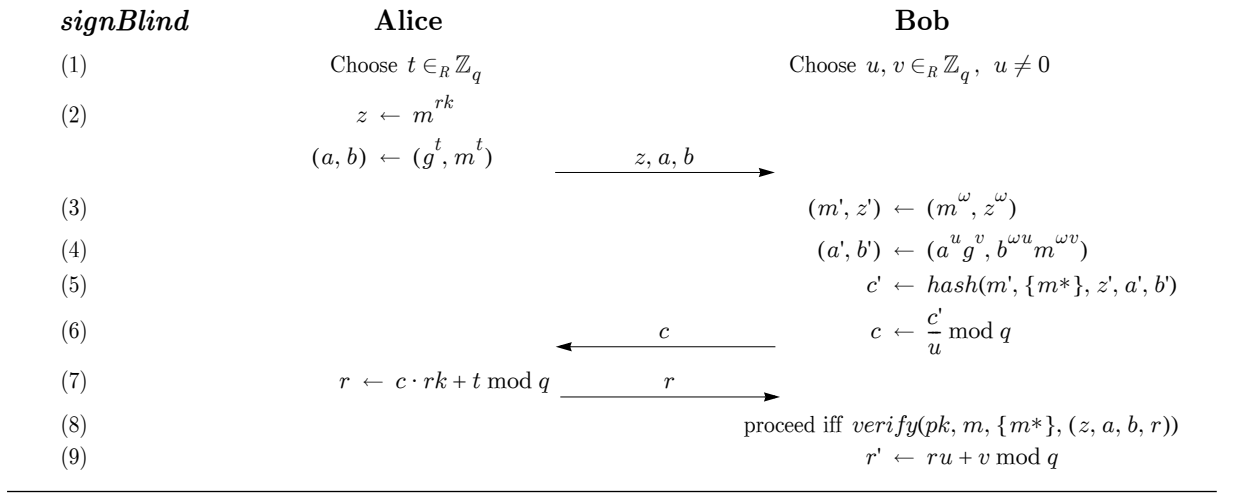


FIGURE 4-5 Producing a Chaum-Pedersen Signature

Bob may take the message $m^* \in M$ as an optional private input. Alice chooses $t \in \mathbb{Z}_q$ uniformly at random and Bob chooses $u \in \mathbb{Z}_q^*$ and $v \in \mathbb{Z}_q$ (step (1)). Alice computes the three values $z \leftarrow m^{rk} \pmod{p}$ and $(a, b) \leftarrow (g^t \pmod{p}, m^t \pmod{p})$ (step (2)), where g is the generator contained in the prekey. She sends the three values to Bob. Bob forms his new message m' and blind signature components z', a', b' in step (3) and (4). He then applies the hash function hash to the results $(m', \{m^*\}, z', a', b')$ (step (5)), prepares his challenge in step (6) and sends it to Alice. Alice computes a corresponding response r (step (7)) and returns it to Bob. If the signature (z, a, b, r) is valid for message m (step (8)), Bob finally computes the missing blind signature component r' (step (9)) and outputs the new message and signature $(m', (z', a', b', r'))$.

The signing protocol allows Bob to have an optional message $m^* \in M$ (enclosed in braces) signed directly. If the optional message is used, we refer to *signBlind* as the *extended signing protocol* of the Chaum-Pedersen(ℓ) Signature Mechanism. It is used in Section 5.5 on p.110 and Section 7.3 on p.143.

Verify

A message $m' \in M$, optional message $m^* \in M$ and signature $\sigma' = (z', a', b', r') \in \Sigma$ are verified by the following algorithm.

$$\begin{aligned} \text{verify}(pk, m', \{m^*\}, (z', a', b', r')) &\stackrel{\text{Def}}{=} (g^{r'} = pk^{c'} a' \pmod{p}) \wedge (m'^{r'} = z'^{c'} b' \pmod{p}) \\ &\text{where } c' = \text{hash}(m', \{m^*\}, z', a', b') \text{ .} \end{aligned} \quad \blacklozenge \quad (4.12)$$

Proposition 4.5 Witness Equivalence

For each prime q , \equiv is an equivalence relation on the domain $\mathbb{Z}_q^\ell \setminus \{(0, \dots, 0)\}$. \blacklozenge

Proof

The witness equivalence is obviously reflexive and symmetric. In order to check transitivity, consider three witnesses $u, v, w \in \mathbb{Z}_q^\ell \setminus \{(0, \dots, 0)\}$ that satisfy

$$\forall \lambda, \mu \in [1, \ell] : u_\lambda v_\mu = u_\mu v_\lambda \pmod{q} \text{ and} \quad (4.13)$$

$$\forall \lambda, \mu \in [1, \ell] : v_\lambda w_\mu = v_\mu w_\lambda \pmod{q} \text{ .} \quad (4.14)$$

By multiplying equations (4.13) and (4.14) we obtain the following equation

$$\forall \lambda, \mu \in [1, \ell] : u_\lambda v_\mu v_\lambda w_\mu = u_\mu v_\lambda v_\mu w_\lambda \pmod{q} \text{ .} \quad (4.15)$$

If all $v_1, \dots, v_\ell \neq 0$, the latter equation (4.15) simplifies to the desired result:

$$\forall \lambda, \mu \in [1, \ell] : u_\lambda w_\mu = u_\mu w_\lambda \pmod{q} \text{ .} \quad (4.16)$$

We are left to show that equation (4.16) holds also if any of the v_1, \dots, v_ℓ disappears. For those index pairs $(\lambda, \mu) \in [1, \ell]^2$ where both $v_\lambda, v_\mu \neq 0$, the respective $v_\lambda v_\mu$ cancel out on both sides of equation (4.15), and thus equation (4.16) holds for these index pairs (λ, μ) . For each $\lambda \in [1, \ell]$ where $v_\lambda = 0$, equation (4.13) implies that $v_1 = v_2 = \dots = v_\ell = 0$ or $u_\lambda = 0$, and equation (4.14) implies that $v_1 = v_2 = \dots = v_\ell = 0$ or $w_\lambda = 0$. Since the witness $v = (0, \dots, 0)$ is not in the domain on which the relation is defined, we are left with the condition $u_\lambda = w_\lambda = 0$, which satisfies equation (4.16) for all index pairs $(\lambda, 1), (\lambda, 2), \dots, (\lambda, \ell)$. Thus, equation (4.16) is satisfied. \square

Security Suggestion 4.6

For all $\ell \in \mathbb{N} \setminus \{1\}$ the Chaum-Pedersen(ℓ) Signature Mechanism 4.8 on p.55 is a blind signature scheme according to Definition 4.7 on p.53. It is conjectured by Brands [34] and others [192] to be one-time and restrictive in the random oracle model under the SDL Assumption 3.20 on p.29, the SDH Assumption 3.23 on p.30 and possibly further assumptions that have not been made explicit in the open literature. \blacklozenge

Security Considerations**EFFECTIVENESS**

We show that the output of *signBlind* satisfies the algorithm *verify*:

$$g^{r'} = g^{ru+v} = g^{(c \cdot rk+t)u+v} = g^{(\frac{c'}{u} \cdot rk+t)u+v} = g^{rk \cdot c' a^u g^v} = pk^{c'} a' \pmod{p} ,$$

by steps (8), (7), (6), (2), (4) in turn and using $g^{rk} = pk \pmod{p}$. Analogously:

$$m^{r'} = m^{\omega(ru+v)} = m^{\omega((c \cdot rk+t)u+v)} = m^{\omega((\frac{c'}{u} \cdot rk+t)u+v)} = m^{rk \cdot \omega c' b^{\omega u} m^{\omega v}} = z'^{c'} b' \pmod{p} ,$$

by steps (3) and (8), (7), (6), (2), (3) and (4) in turn.

BLINDNESS

This has been shown by Chaum, Pedersen in [73] using the fact that signing is a proof of knowledge of the signing key.

ONE-TIMENESS AND RESTRICTIVENESS

Brands has conjectured in [34] Section 9 and [35] that the Chaum-Pedersen(ℓ) Signature Mechanism is one-time and restrictive. He uses reasonable heuristics to argue for both, but no formal proofs have been given yet. Pointcheval and Stern have proved a close variant of the Chaum-Pedersen(ℓ) Signature Mechanism to be one-time in the random oracle model [192]. The problem with their result was that the reduction is not polynomial in the number ℓ of interactions between signer and attacking recipient. So it does not cover all polynomial-time active attacks. A polynomial-time reduction has later been revealed by Pointcheval in [193].

Let an arbitrary pair (m, σ) be given such that $m \in G_q$ and $\sigma = (z, a, b, r) \in \Sigma$ is a valid signature for m with respect to some public key $pk \in G_q$. In order to find two different optional messages m^*_1, m^*_2 such that $verify(pk, m, \{m^*_1\}, \sigma) = verify(pk, m, \{m^*_2\}, \sigma) = TRUE$ an attacker had to find m^*_1, m^*_2 such that

$$hash(m, m^*_1, z, a, b) = hash(m, m^*_2, z, a, b) ,$$

which is infeasible in the random oracle model. □

Remarks: Brands has conjectured in [34] Section 11 that the Chaum-Pedersen(ℓ) Signature Mechanism is as secure for $\ell \geq 2$ as for $\ell = 1$, which is plausible, because the signing protocol is independent of the parameter ℓ . In fact Brands has published his off-line e-cash mechanism in [35] using the Chaum-Pedersen(1) Signature Mechanism. The definition of restrictiveness, however, does not cover the case of $\ell = 1$ because then the witness relation collapses to mere identity leaving no room for blinding signatures any more. It is an interesting open question, how to define restrictiveness in a more general way than above.

It has been shown for the Chaum-Pedersen(ℓ) Signature Mechanism that *signBlind* is a proof of knowledge of the signing key. In particular, it is a variant of *CEG*(2), with candidate (pk, z) and witness (rk, m) . (Compare the signer Alice in Figure 4–5 on p.56 with the prover in Figure 4–2 on p.43.) The blind signature protocol in Figure 4–5 on p.56 can then be interpreted as a diverted proof of knowledge where the recipient plays the roles of both the co-prover (steps (3), (4), (6), (8)) and the verifier (step (5)). For more details see Bleumer [22] and Blaze, Bleumer, Strauss [20].

5

Credential Schemes

*“The decision we make about communication security today
will determine the kind of society we live in tomorrow.”*
— Whitfield Diffie [93]

In the following sections, we develop advanced cryptographic implementations of certificates that are unforgeable, non-repudiable, protect holder privacy and optionally have any of the other security requirements introduced in Section 2.2 on p.6. We do not consider repudiable certificates, which are based on symmetric authentication mechanisms. For example, the *Kerberos* tickets [142] are this type of certificates. Non-repudiable certificates without privacy are easily implemented by means of ordinary *digital signatures* [94]. Typical examples are *public key certificates* according to X.509 [138,155], which employ RSA [201], DSA [164] or ECDSA. The first conceptual proposal for non-repudiable certificates protecting the privacy of holders, even against coalitions of computationally unlimited certifiers and verifiers, was introduced by Chaum [58,59]. He also coined the term “*credential*” for this type of certificates. An overview of these and other practically relevant credentials has been given by Jones et al [141]. As credentials are a special case of certificates, the terminology introduced for certificates in Section 2 on p.5 carries over to credentials.

We introduce the new concept of credentials with anonymous certifiers who may be re-identified in case of dispute later on. Since a certifier is anonymous relative to a certain group of potential certifiers, we call this new concept *group credentials*. This section and the following lay down the basic building blocks in order to prepare the cryptographic mechanism presented in Section 7 on p.141.

Section 5.1 on p.60 presents an overview of the existing literature on credentials, in particular offline credentials, and the topology of the most advanced decentralized systems implementing them (*wallets with observers*). In Section 5.2 on p.71 and Section 5.3 on p.81 we layout a formal framework for offline credentials and their security requirements. We propose in Section 5.4 on p.90 a new cryptographic mechanism for *personal credentials*. To prepare more advanced cryptographic mechanisms in Section 7 on p.141, we revisit Brands’ proposal for *coin credentials* [34,35] in Section 5.5 on p.110. Bond credential schemes combine the integrity requirements of personal and coin credentials (Section 5.6 on p.121). Showing personal AND coin credentials is discussed in Section 5.7 on p.121.

5.1 Overview

Chaum’s pioneering work [58,59] was the starting point for numerous papers dedicated to privacy of digital certificates from both a practical and a theoretical point of view. We will now give a structured survey over the last 15 years of (public) literature on credential schemes and closely related topics such as untraceable payment schemes. This might be of particular help to the novel reader because credential schemes are among the small and delicate topics of current cryptographic research. Early credential schemes were covered by Brassard et al in their tutorial on “modern cryptology” [41] (about 3.7% of the pages). Recent monographs and books on cryptography cover credential schemes poorly, e.g. Simmons [215], Stinson [218], Salomaa [204], Menezes, van Oorschot, Vanstone [157], Schneier [206], and Goldreich [120]. Only Simmons and Schneier mention credentials in the above sense; 3 out of 13 chapters in Simmons’ book cite Chaum’s work, and Schneier dedicates about 1.2% of his pages to a special kind of credential scheme: electronic cash (under the heading “esoteric protocols”).

In principle, certificates threaten the holders’ privacy if certifiers and verifiers together can link different actions on the same or related credentials. They could use such data to compile for example profiles of certain holders’ behaviors or they could trace credentials from holder to holder (*traceability*). The basic idea to prevent such collusions of certifiers and verifiers is to let holders use randomly chosen one-time pseudonyms for their interaction with certifiers and verifiers [58,59].

Unlinkability is an important holder privacy requirement (Section 2.2.5 on p.7) of all credential schemes. Consider a typical activity of a holder: He first obtains a credential from an issuer and afterwards shows it one or more times to a verifier¹.

- If the issuer and the verifier together cannot link all “shows” of one credential to the corresponding “issue”, we call the credential scheme *issue-wise unlinkable*.
- If, furthermore, they cannot link any two “shows” (of one credential), we call the credential scheme *show-wise unlinkable*.

Pseudonyms seen by issuers or verifiers are called *source pseudonyms* and *target pseudonyms*, respectively. For example, if credentials can be shown more than once but only for the same target pseudonym, then the credential scheme can be issue-wise unlinkable but not show-wise. If credentials can be shown only once, then issue-wise unlinkability and show-wise unlinkability are no different.

If basic legitimations are implemented by unlinkable credentials using random one-time pseudonyms then the *holder authorization problem* (Section 2.2.3 on p.7) and the *certification problem* (Section 2.2.4 on p.7) can be rephrased as follows:

- **Holder authorization problem:** The verifier must check whether the actual holder is legitimated to use the credential at hand.
- **Certification problem:** The verifier must check that the credential content is certified for the target pseudonym (although the issuer knows the holder only by the source pseudonym).

1) In fact, the holder may show his credential to several verifiers. However, if we talk about holder privacy, we consider them all as one big collusion V .

The **holder authorization problem** is solved by implementing a set of rules that characterize the intended credentials. In this work, we deal only with two very basic kinds of holder authorizations: Whether a credential is acceptable to a verifier can depend on the biometric identity of the holder or on how often it has been used before. If $K \in \mathbb{N}$ is the limit of how often a credential may be used, we call it a *K-show credential*, or a *nonce-credential* if $K = 1$. A typical example are electronic coins. In general, *K-show credentials* ($K > 1$) can be simulated by K separate nonce-credentials.

The classical approach to solving the holder authorization problem is by online control where some *control center* oversees all credential shows, and denies them if necessary. See work of Miller [158], Davida, Frankel, Matt [86], Davida, Frankel [87] and Jain, Hong, Pankanti [140]. For example, holders can simply be asked to store all their credentials in a safe database accessible only by the control center. Such a centralized online solution, though, is unacceptable for many kinds of credentials. Even if the control center is distributed, it poses a severe threat to the availability of credentials and the privacy of holders, and thus must be carefully designed. A good example are the online solutions for privacy protecting cash checks proposed by Chaum in [63]. We are going to take the requirements of availability and privacy seriously and therefore confine ourselves primarily to off-line solutions of the legitimation problems. That is

- holders shall keep their credentials locally, and
- holders shall be able to obtain and show their credentials without depending on remote third parties being online. See Bleumer [23,24].

Efficient solutions to the **certification problem** can be obtained by some kind of digital signature, thereby providing holders and verifiers with a proof of origin of certification. The meaning of such a signature, i.e., the intended content of a credential, could be associated to the secret key used to produce the signature [60].² For simplicity, we assume in this overview that there is only one possible content.³ The question left is how a digital signature received for a source pseudonym can be shown (a) without revealing that source pseudonym or any other identifier that is easily linkable to the source pseudonym and (b) in a way that the verifier is provided with a proof of the holder's certification that convinces not only the verifier but third parties as well. The second condition reflects the fact that for most applications such proofs are strongly desirable as legal evidence. Conceptually, two solutions have been proposed in the literature to solve both problems at the same time.

- (i) The signature received for the source pseudonym is somehow converted into a signature for the target pseudonym. Signature schemes with this conversion property have been introduced by Chaum as *blind signatures* [58,59,60,61,64]. The first complete credential scheme was based on an RSA signature scheme using the fact that RSA is an automorphism on \mathbb{Z}_n^* , where n is the public (composite) modulus [60]. Assume (n, e) is the public verifying key (public modulus and expo-

2) The reason for this perhaps slightly surprising suggestion is that the content of a credential needs to be carefully separated from its owner, i.e., the source pseudonym certified. The former shall not be subject to any change by the owner, whereas the latter is intended to be so. If the content is associated to the secret key of the certifier, then changing the content of a credential is equivalent to forging the whole credential, which is presumed to be infeasible.

3) This assumption is without any restriction unless, given a certain security parameter, the number of potential contents exceeds the number of possible secret keys. In fact, the most interesting case in practice is a limited (finite) number of potential contents, because the average unlinkability holders can achieve is reciprocal in the number of potential contents.

ment for verification) of some issuer and (p, q, d) the corresponding private key (the two prime factors of n and the corresponding exponent for signing such that d is the multiplicative inverse of e in $\mathbb{Z}_{\phi(n)}^*$, where $\phi(n) = (p-1)(q-1)$ is Euler's totient function). If some customer wants to get issued a credential, he chooses a blinding factor $b \in \mathbb{Z}_n$ and a target pseudonym $\psi' \in \mathbb{Z}_n$ under which he wants to show the credential later on. Then he asks the issuer to issue a credential for the source pseudonym $\psi = b^e \psi' \bmod n$. The issuer issues the credential $\chi = \psi^d \bmod n$, and the customer computes the desired credential $\chi' = \chi b^{-1} \bmod n$ for his target pseudonym. This credential satisfies the verification predicate $\psi' = \chi'^e \pmod n$. Alternatively, the customer might start with a given source pseudonym ψ and end up with a new target pseudonym ϕ from the protocol. In fact, the source and target pseudonym are unconditionally unlinkable by any provider who might see them. Since that proposal, blind versions have been constructed for many digital signature schemes; by Chaum, Pedersen [73] and Brands [36] for Schnorr signatures [209], by Camenisch, Piveteau, Stadler [49] for Nyberg-Rueppel signatures [168,169] and for a variant of DSA [164], by Horster, Michels and Petersen [135] for ElGamal signatures [99], and by Pointcheval and Stern [192] for certain adaptations of Schnorr and Guillou-Quisquater signatures. This is not surprising because most digital signature schemes are built around a homomorphic one-way function [25], which is also useful for blinding purposes.

- (ii) An alternative approach is based on *zero-knowledge proofs of knowledge* [123]. The certifier uses an ordinary digital signature for certification and the holder proves to the verifier in zero-knowledge that he knows a signature for one of his pseudonyms [83,187]. A proof of the holder's certification can be obtained by using *non-interactive zero-knowledge proofs of knowledge* [205]. The security of this type of scheme can be proven under general cryptographic assumptions. However, the message complexity and encryption complexity of existing solutions are prohibitively large, e.g., the length of a proof of holding a certification (i.e., valid signature) depends on the complexity of the signing function itself [113]. Although no lower bound is proven yet, we will not pursue this approach further. However, we will employ interactive proof-of-knowledge protocols in a different way.

5.1.1 Core Credential Schemes

In the following, we consider four core credential schemes. If a credential scheme solves the certification problem, it is called a free credential scheme. If a free credential scheme solves the holder authorization problem by checking the biometric identity of holders, it is called a personal credential scheme. If a free credential scheme solves the holder authorization problem by checking that a credential has not been shown more than a predetermined number of times, it is called a coin credential scheme. If a free credential scheme solves the holder authorization problem by checking both conditions it is called a bond credential scheme. The relations are depicted in Figure 5-1 on p.63 with the more special schemes depicted closer to the bottom. Existing cryptographic implementations for each of these core credential schemes are discussed now in more detail.

Free Credentials By definition, there is no holder authorization problem in free credentials because free credentials are valid regardless how often they are shown or by whom. Free credentials without holder unlinkability are well-known. They only require a digital signature to bind the certificate con-

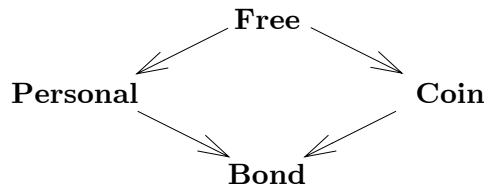


FIGURE 5-1 Relations of the Core Credential Schemes

tent to the issuer. We will not consider them further. The first proposal of a free credential scheme with holder unlinkability is due to Chaum [60]. We have reviewed it in the previous subsection. That paper caused some confusion because the motivating examples suggested that it is a personal credential scheme, which it is not. It does not address the transfer prevention problem against active attackers (see the paragraph after the next one).

Credential schemes that allow to show credentials more than once in an unlinkable way have been proposed by Damgård [83] and Chaum [60,64]. Damgård’s construction is mainly of theoretical interest because it is based on unconditionally secure bit commitment schemes and general 2-party computations. While the protocols are highly impractical, Damgård achieves a provably secure construction with minimal complexity theoretic assumptions. On the other hand, Chaum’s solution [64] is based on blind RSA signatures and is quite efficient. Reconsider the setup of the blind signature scheme sketched under “certification problem (i)” above. Here, pseudonyms are products of a unique person identifier u and a randomly chosen blinding factor b . Let Bob’s pseudonyms for provider Alice and Charlie be $\psi_A = ub_A^e \pmod n$ and $\psi_C = ub_C^e \pmod n$, respectively. Alice gives a credential $c = \psi_A^d = u^d b_A \pmod n$ for pseudonym ψ_A to Bob. Bob, knowing his blinding factors b_A and b_C , *re-blinds* his credential into $c' = c(b_C/b_A) = u^d b_C \pmod n$ and shows this for pseudonym ψ_C to Charlie. Finally, Charlie verifies that $c'^e = \psi_C \pmod n$.

Chaum suggests that some globally trusted initialization center shall overlook the generation of person identifiers u , to link them to the legitimate holders, and to guarantee pseudonyms to be formed correctly. The protocols employ quite expensive *cut-and-choose* methods in order to reduce the trust that holders have to put into the initialization center.

These cut-and-choose methods ensure that pseudonyms are well-formed although they cannot be inspected and checked directly. The idea is that during the initial step, customers are asked to produce and submit more pseudonyms than they really need and then have to open a certain fraction q of them for spot checking. If those pseudonyms opened are found to be formed correctly, the remaining (private) pseudonyms are expected to be correct with error probability $1 - q$. These basic cut-and-choose methods are inefficient in the sense that the overhead of producing and checking pseudonyms is inverse in the error probability achieved. More importantly, this initialization center can of course not prevent holders from simply giving away or exchanging their pseudonyms and even their identifiers.

Personal Credentials Credentials that are not transferable between human owners defined by a biometric identity are called personal. An imposter seeking a personal credential of Alice cannot be distinguished from Alice, unless he is verified biometrically. Likewise, an imposter showing a personal credential of Alice, the owner, cannot be detected unless the verifier refers to the biometric identity of

the imposter. Since in both cases the owner and the masquerading holder are digitally indistinguishable, proper holder authorization can be enforced only by biometric identification of the recipient (holder) before issuing (showing) a credential [83,181].

Chaum has mentioned the idea of biometric verification in an earlier article [61] published in 1985, in order to solve the problem of restricting individuals to use only one pseudonym with each organization such that each organization could link all its correspondence with each individual whereas different organizations were prevented from linking their views on any individual. Chaum's idea was to setup a central organization whose sole purpose was to issue special "is-a-person" credentials after identifying persons biometrically. Once, a person is issued an "is a person" credential, the person will never receive a second one. An organization can then require to see an "is a person" credential before issuing its own credential. The approach worked, if only for one organization, but the biometrics was not suggested to prevent individuals from transferring their credentials among one another.

When Chaum published his paper [64] in 1990, he was probably aware of the fact that only by biometric recognition individuals can be prevented from sharing or transferring their credentials. This insight is sketched in his patent [62], which was filed in 1988. Damgård [83] was probably the first to mention it in the open literature.

Yet, Chaum's tutorial paper [65] published in 1992 did not mention that personal credentials need some sort of biometric recognition. As late as 1996, Chen still suggested another all-digital personal credential scheme [76,74] based on the SDL-Assumption. Chaum and Pedersen [73] proposed that users hold their personal databases in personal devices (see Section 5.1.3 on p.67), which need to be equipped with so-called observers. These are tamper resistant security modules (see Section 5.1.4 on p.68) that authorize transactions of their host devices. Although not mentioned by Chaum and Pedersen in [73], their solution could be turned into a personal credential scheme by adding a biometric sensor to the observer. Unless observers are broken holders cannot show each other's credentials, not even if they lend away their MUDs and observers. If attackers break their observers, though, they can also forge new types of credentials, which they have never got issued before. Chaum and Pedersen's proposal also requires an initialization center that is needed to certify millions of one-time public keys in advance, which are needed by the observers each time they authorize their host devices to show a credential. Once an observer has exhausted its supply of certified public keys, it needs to go back to the initialization center in order to reload certified one-time public keys. This is computationally inefficient and inconvenient from a user's perspective.

An approximating approach to non-transferable credentials has been proposed by Lysyanskaya, Rivest, Sahai and Wolf in [147]. They tie personal credentials to the private master keys of individuals in such a way that individuals can share their credentials only by sharing their full digital identity. The approach approximates non-transferability in the sense that the more personal signing and decryption capabilities are associated with an individual's master key, the less incentive there is to transfer or share credentials. This approximating approach has been proposed earlier by Dwork, Lotspiech and Naor for protecting digital content [98] and later by Goldreich, Pfitzmann and Rivest for controlled self-delegation [103]. Lysyanskaya et al propose two constructions, an impractical mechanism based solely on one-way functions, which is an extension of Damgård's result [83] mentioned

under “Free Credentials”, and an efficient mechanism for one-time credentials, which is very similar to Chen’s result [76,74] mentioned above.

Coin Credentials Coin credentials do not require biometric means. By digital means alone, a holder showing a K -show credential $K+1$ times (*overshowing*) can at least be detected. Note, that in case of overshowing there is an additional transaction (i.e., the $(K + 1)$ st show), that can be used to distinguish authorized from unauthorized use, and in particular identify the cheater. In their seminal paper [69], Chaum, Fiat and Naor presented the first construction based on RSA signatures. This inspired plenty of variations and improvements [67,174,128, 175,3,131]. All these schemes are feasible but no proofs of security have been given that rely on the standard RSA assumption. Franklin and Yung made the holder authorization problem of coin credentials more explicit. They identified pure blind signature primitives as too liberal and introduced the more appropriate primitive of *oblivious authentication* [113] (not to be confused with *oblivious signatures* [75]). The additional constraint is to prevent holders from choosing their messages directly and thereby letting them encode no or false identities into their coins. All constructions of offline payment schemes proposed up to and including [113] enforce this correct encoding of holders’ identities by a *cut-and-choose* method [196]. Oblivious authentication that is free of cut-and-choose has been proposed by Ferguson and Brands. Their approach is to have the holders choose preimages of their messages under a previously agreed one-way function (instead of choosing their messages directly). Ferguson started from work with Chaum and van Antwerpen [66,3] and came up with a *randomized blind signature scheme* [108,109] based on non-standard RSA assumptions. Building on the signature schemes of Schnorr [209] and Chaum et al [39], Brands came up with a *restrictive blind signature scheme* [34,35] based on the SDL-Assumption. Radu et al presented another restrictive blind signature scheme [197] based on the same assumption that is derived from Okamoto’s identification scheme type I [172]. Randomized blind signatures and restrictive blind signatures considered as abstract cryptographic primitives appear rather similar if not equivalent. However, since they have been introduced as concrete cryptographic mechanisms, we do not analyze their functional relationship in more detail. With respect to computation and communication complexity, Brands scheme is superior. More recent work has based the above primitives upon other well-established signature schemes [36,37]. Security against holder framing, i.e., falsely accusing the holder of double spending, has been considered by Ferguson [108] and Brands [34,35].

Since electronic cash is in fashion, most privacy protecting credential schemes proposed in the open literature deal with some kind of electronic coin that can be spent once. These schemes require to fix a target pseudonym at issuing time. Each credential can be shown only for this pseudonym and thus credentials can be shown essentially to only one verifier. If they were shown to two different verifiers, these could immediately link their views. Discrete log based credential schemes giving show-wise unlinkability have been posed as an open problem by Chen [76,74].

Overspending detection is a weak countermeasure as actual damage tends to be detected too late to be limited effectively. Building on work of Chaum et al [39,73] on tamper resistant observers, Ferguson and Brands extended their schemes [34,35] in order to *prevent* holders from overspending as a first line of defense and still provide overspending detection in case that first line is broken (see Section 5.1.4 on p.68).

Naccache and van Solms pointed out that coin credential schemes with unconditional unlinkability probably encourage money laundry, perfect blackmailing and related crimes [160]. This problem is probably the bigger the larger amounts of money can be moved with a coin credential scheme. Several solutions have been suggested to strive a better balance between customer privacy and law enforcement. Several solutions and improvements have been suggested by Stadler, Piveteau and Camenisch [50,52], Brickell, Gemmell, Kravitz [43], Camenisch, Maurer, Stadler [53,54], Frankel, Tsiounis, Yung [85,112]. They all use an additional trusted center that can de-anonymize the suspects of big money transactions. However, the proposals differ in requiring or not the trustee to be on-line for each withdrawal, efficiency of identifying the customer and various other efficiency measures.

There are additional features possible for coin credentials, namely *transferability* and *dividability* (also called *divisibility*). Transferable credentials⁴ can be re-spent by the payee, and dividable credentials can be split by their holders into “smaller” coin credentials with their sum matching the original credential. Okamoto and Ohta [175] considered the ideal payment scheme to be an offline transferable dividable coin credential scheme that prevents overspending and called it *universal cash*. A fundamental theorem by Chaum and Pedersen [72] however, reveals that transferable coins must grow linearly in the size of the number of transfers if they shall keep identifying a double spender. A weaker form of dividable credentials has been introduced earlier as *pre-paid offline checks* [31,67,35]. From such a check, only one “smaller” credential can be extracted whereas the rest of the original check is not payable, but can only be refunded by the bank. (*Post-paid offline checks* cannot protect the privacy of holders in the strong sense that we aim at, because they have to carry information about their account holders.) Pedersen introduced *micro payment schemes* [178] (also called *phone ticks*) that offer improved efficiency if small amounts are successively paid to the same payee so rapidly that it makes little sense to keep these payments unlinkable.

In order to focus on the new extensions (Section 5.1.2 on p.66), we disregard such additional features like dividability, transferability, checks and micro payments. All of them are interesting additions to our refinements and it remains a challenge to combine them with our new ideas.

Bond Credentials According to their specification as personal and coin credentials (Section 2 on p.5) they can be implemented by combining the techniques used to achieve these requirements separately. Such combinations have not been studied in the literature yet.

5.1.2 Issuer Groups and Issuer Anonymity

All previous concepts and constructions of credential mechanisms (including payment systems) assumed that issuers of credentials are identified parties, known by the holders as well as by the verifiers. We introduce the concept of *relative anonymity* for issuers (Section 2.2.6 on p.7); it can be applied to all the four core credential schemes. Relative anonymity was first introduced by Chaum and van Heijst [70] for digital signature schemes. The idea is to form groups of signers and to publish public keys of each group rather than each individual signer. A signer of a group produces signatures that can be publicly verified to originate from the signer’s group—but not from which member. In case of dispute later on, the group is able to identify the signer by his signature. The group can identify each

4) Typically, they are called “coins” in the literature.

of its members, because members have to deposit a partial private key before they are admitted to join the group.

We translate the concept of relative anonymity to credential schemes by constructing *group oriented blind signature schemes*. We will also compare our new constructions to alternative solutions by Lysyanskaya and Ramzan [149]. The resulting credential schemes allow the issuers to remain anonymous against verifiers of their credentials. We have shown how this new concept can be applied, e.g., to a healthcare system funded by compulsory health insurances [26,27] in order to obtain strong privacy of the patient-doctor relation against health insurers (see Section 8 on p.145).

5.1.3 Mobile User Devices

For reasons set out in Section 5.1 on p.60 (holder authorization), we are interested in legitimation systems where users employ offline devices that can store credentials for their users and support receiving and showing credentials without relying on any other device, e.g., smart card reader, or online service, e.g., authentication server.

Clearly, such user devices must have at least a power supply, a CPU, RAM and non-volatile memory, a user interface (input/output facilities) and communication facilities in order to interact with other pieces of hardware. If users want these devices to perform legally relevant actions in behalf of them, they will want to have these devices under their exclusive control. Preferably, such devices are small and lightweight enough so that users find it easy to carry their devices by them. (Any kind of stationary computer is difficult to protect from being shared by several users.) We call such hardware *mobile user devices (MUD)* according to [185]. Practical examples of mobile user devices are *computer cards, palmtops, organizers, personal digital assistants, or cell phones*. Practical examples of communication facilities are direct plugging, serial or infrared connection, local area network or radio wave.

Even, Goldreich and Yacobi [101,102,100] were one of the first who considered *electronic wallets* as an application-specific mobile user device. Such an electronic wallet allows its user to inspect the current amount of money actually stored, to pay and to receive money without having to ask a bank or other third party for assistance. Chaum also discussed multi-purpose mobile user devices [65].

In order to be acceptable as user agents in legitimation systems, mobile user devices have to justify *personal agent trust* (i.e., as long as a user holds his MUD in his hands it should be credible to function correctly) and *captured agent trust* (i.e. as soon as a user loses control over his MUD, it should block and stop to deliver any service). The former is usually achieved by proper quality control and diversity during the design process whereas the latter is achieved by sufficient tamper-resistance and passphrase protection of all security critical transactions and the local memory itself. Whenever we apply mobile user devices in the following, we presuppose that they justify personal agent trust and captured agent trust. A fair source how this can practically be achieved is [185].

Typically, providers offer their services to holders by means of *stationary provider machines (SPM)* that are capable to interact with mobile user devices and are usually more powerful than MUDs. Typical examples are *automatic teller machines (ATM)* or *point-of-sale terminals (POS)*.

5.1.4 Embedded Security Modules

In the context of legitimation schemes, it is the providers' and the verifiers' interest to enforce holder authorization. For instance, in the case of personal credential schemes it is usually the providers and

5 CREDENTIAL SCHEMES

the verifiers who want to prevent credentials from being transferred between owners. In case of coin credential schemes it is the banks' and the merchants' interest to prevent holders from double spending. In principle, holder authorization can be enforced either by a provider or by an agent that is trusted by the provider(s) and verifier(s) and is involved in all *critical transactions*, i.e., transactions where holder authorization could be breached. In an offline credential scheme such a trusted agent can only be distributed over a number of *security modules* [185] that are embedded into the mobile user devices themselves and each security module participates in its host's critical transactions.

In order to be acceptable to providers and verifiers of legitimation systems, such security modules have to justify *undercover agent trust*, i.e., providers and verifiers need to be sufficiently assured that these security modules function correctly even though they are embedded in hostile environments. This is typically achieved by proper tamper-resistance.

Figure 5–2 on p.68 (a)-(d)) depicts the simple configuration of a stationary provider machine

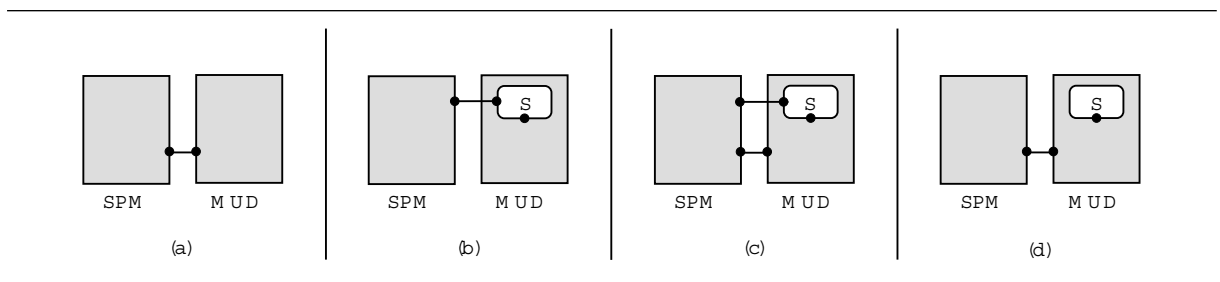


FIGURE 5–2 Configuring an SPM, a MUD and a Security Module S

(SPM) and a MUD (a) and three possible configurations of how a security module S acting as a provider agent could be embedded into the MUD (b-d). The configurations (b-d) could be extended to several security modules (per MUD) trusted by different providers that possibly distrust each other.

In (a) there is a single communication line between the SPM and the MUD. The stationary provider machine has no control over the MUD and thus cannot trust in the MUD at all.

In the following configurations, the providers trust in the security modules (S) that are embedded into the MUDs.

In (b) all communication between SPM and MUD is mediated by the security module S , whereas in (c) each two of the three participants have a direct connection. The latter has an advantage if the communication bandwidth between SPM and MUD is larger than that between SPM and S . More importantly, both configurations leave the MUD highly vulnerable of getting profiled. As long as the MUD maintains its connection to the SPM, it cannot prevent the security module from sending profiling data to the SPM.

In (d) all communication between SPM and MUD is by direct connection, with the security module sitting in the back of the MUD. This enables the MUD to shield the security module against *in-band inflow* and *in-band outflow*, i.e., the MUD can mediate all communication from the SPM to S and back. At the same time the SPM can require a co-operation of the security module in critical transactions in order, for example, to enforce correct holder authorization. Since the MUD shields its security modules against inflow and outflow it makes sense to consider MUDs using different security

modules in the same transaction or one after another in different transactions. This configuration (d) was first considered by Brands, Chaum, Cramer, Ferguson, and Pedersen [39,73,65] who called it *wallet with observer*. Cramer and Pedersen [80] aimed at privacy even in the (realistic) case that some observers should once be captured or returned to the providers, thereby causing *out-of-band outflow*. In this case, protection against inflow is particularly crucial. Like outflow, inflow can occur in-band and out-of-band. *Out of band inflow* can be surprisingly subtle; for example by internal high precision clocks that are initially synchronized with an outside attacker clock, or by internal radio micro-receivers that pick up broadcast signals from satellites (GPS) or cell phone networks, which can also be received by the providers.

In the following, we will only consider the wallet-with-observer configuration depicted in Figure 5–2 on p.68 (d). Hence, we call the security module an *observer*. We assume observers to be properly manufactured and not equipped with any hidden means to synchronize with their environment. Brands [34,35] proposed a coin credential scheme for this configuration. An improved version of his proposal underlies the concept of the EU funded project CAFE (Conditional Access For Europe) [30].

5.1.5 Biometric Sensors

Implementing personal and bond credential schemes requires biometric sensors. The strictest and also most intrusive option is to prevent a holder to get rid of his personal mobile user device by attaching it to or implanting it into the physical body of its holder in some irremovable way. For example, this method has been adopted to trace arrested criminals [136]. Obviously, the approach raises severe ethical questions and, in the following, we assume that mobile user devices are normal devices like pocket calculators or watches that holders can store, carry, replace, sell or destroy as they wish.

For positive credentials, it suffices to detect whether a device belongs to the holder who presents it. Biometric sensors that can verify this kind of holder authorization are called *biometers* in the sequel. Conceptually, biometers can be regarded as security modules (Figure 5–3 on p.69) that

- work as provider agents when embedded into MUDs (d-1) or
- as provider and holder agents when embedded into SPMs (d-2).

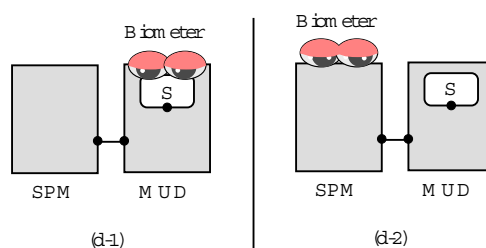


FIGURE 5–3 Embedding a Biometer

In (d-1) the biometer is physically embedded into the observer, so we call it a *biometric observer*. The eyes represent a biometric sensor. Biometer and observer(s) co-operate in behalf of the provider. If each holder has her own personal MUD, the biometers only need to *biometrically verify* their respective holders, they do not need to *biometrically recognize* different identities. (The latter is more difficult and expensive to achieve.) So personal devices allow their holders to perform their transactions

remotely and without any need to present themselves physically to the respective providers. Furthermore, the biometric data accumulated by the biometer can be shielded by the MUD against outflow (see discussion of Figure 5–2 on p.68 (d)).

In (d-2), the biometer is physically embedded into the SPM. Again the eyes represent a biometric sensor such as a video camera sitting in an ATM. Any more or less centralized biometric identification mechanism has to recognize different holders and, therefore, aggregates a lot of personal information over time. Analogous to Figure 5–2 on p.68 (d), each SPM could host several identifying mechanisms each trusted by different holders or groups of holders who possibly distrust each other. In contrast to (d-1), holders cannot be recognized remotely; they have to present themselves physically to the SPM. The accumulated biometric data is protected against inspection by the SPM only by the tamper resistance of the biometers. More analysis of the approaches (d-1) and (d-2) and more overview of existing literature is found in [24].

In practice, some trust in tamper-resistance against providers might be justified, but in general, this is a much stronger assumption than tamper resistance against holders, because an average provider is economically — and therefore technologically — much more powerful than an average holder. Moreover, providers can afford to install second biometric recognition mechanisms that they can fully control. Therefore, configuration (d-2) is regarded to pose a significantly higher risk on holders' legitimate privacy than (d-1). If sufficient biometric enhancements of MUDs are not available, (d-2) might be the only option affordable, but we do not consider it further. We propose offline personal and bond credential schemes based on biometric observers in Sections 5.4 on p.90 and 5.6 on p.121.

Off-the-shelf biometric products sense fingerprints, faceprints, face thermograms, voiceprints, signature verification, retina patterns. Refer to [88,89,158] for a thorough overview and to [16] for latest results in audio- and video-based methods. The reliability of biometric identification is commonly measured in statistical terms of false rejection rate (FRR) and false acceptance rate (FAR). In practice, a reasonable trade-off between efficiency and reliability is achieved by fingerprinting, and latest devices (about the size of a cigar box) are reported to achieve a single scan in less than half a second with $FAR < 10^{-6}$ and $FRR < 10^{-2}$ [233]. Face recognition is still less reliable [134,237]. The next generation of smart cards, so called “computer cards” are expected to have a biometric fingerprint sensor built in by default. One chip fingerprint sensors are already available [229].

5.1.6 Summary and New Achievements

We summarize in Table 5-1 on p.71 the security requirements for offline credential schemes as introduced in Section 2.2 on p.6. They will be made precise in Section 5.3 on p.81. The two kinds of *interest groups* relevant to credential schemes are providers and holders of credentials. The security requirements of providers are proper holder authorization and certification. Proper holder authorization means transfer prevention in the case of personal credentials, overshadowing detection and/or prevention in the case of coin credentials and both in case of bond credentials. Proper certification requires that the content of a credential cannot be forged (type unforgeability). Issuers may require issuer anonymity. Holders require credentials to be available and working (effectiveness). Holders require unlinkability. In case a credential scheme allows overshadow detection, holders also require holder framing prevention, i.e., they require protection against being falsely accused of having overshadowed.

5.2 DEFINITION OF CREDENTIAL SCHEMES

Interest Group	Free	Personal	Coin	Bond
Providers	—	Transfer prevention	Overshow prevention/detection	Transfer prev. and overshow prev./det.
	Type Unforgeability			
	Issuer Privacy			
Holders	Effectiveness			
	Unlinkability			
	—	—	holder framing prevention	

TABLE 5-1 Summary of Security Requirements

A systematic excerpt of the pioneering references about credentials and our new results is presented in Table 5-2 on p.71. Previous proposals by Chaum and Pedersen [73] can be used to implement personal

Category of Credential	Without/with Observer	Issuer Privacy	
		no	yes
Free	without	[60,61,64,74]	Section 7.2 on p.143
	with	not necessary	not necessary
Personal	without	impossible	impossible
	with	[65], Section 5.4 on p.90	open problem
Coin	without	[69,67,175,108,34,35]	Section 7.2 on p.143
	with	[65,73,109,34,35,30] Section 5.5 on p.110	Section 7.2 on p.143
Bond	without	impossible	impossible
	with	Section 5.6 on p.121	Section 7.2 on p.143

TABLE 5-2 Overview of References about Offline Credential Schemes

credentials if observers are equipped with biometers. In that case, both type unforgeability and transfer prevention rely on the tamper resistance of observers. (Also see the paragraph on personal credentials of Section 5.1.1 on p.62.) We propose a personal credential scheme in Section 5.4 on p.90 that

- (i) relies on tamper resistance of observers only with respect to transfer prevention,
- (ii) requires very little precomputation for each MUD and observer, and
- (iii) produces credentials that can be efficiently AND combined with each other and with coin credentials according to Brands (see Section 5.7 on p.121).

5.2 Definition of Credential Schemes

We propose a uniform framework for the core credential schemes introduced in Section 5.1.1 on p.62. As a preparation of the formal definition, we make some initial assumptions about the keys, pseudonyms and operations being used (Section 5.2.1 on p.72). The general definition of credential schemes follows in Section 5.2.2 on p.74. The security definitions are given in Section 5.3 on p.81.

5.2.1 Preparation

In order to make unlinkability a reasonable requirement, the contents of credentials should be standardized in such a way that only a limited number of different contents occur at all. Otherwise, there are probably many credential contents that are issued to only one or very few recipients such that issuing to and showing by these holders is easily linkable. Moreover, holders should not add any information to the credential content they are issued. Rao and Rohatgi [199] have explained by several famous examples how even short amounts of prose text can be used to identify the originating author.

Hence in the following, each such content defines a separate type of credential. For each type of credential, we assume that issuers use specific issuing key(s). As outlined for the certification problem in Section 5.1 on p.60, the idea is to certify a credential of type t by signing a holder's source pseudonym with a signing key that has previously been registered to establish a type t credential. So for every type t that an issuer may provide credentials for, he has to have a corresponding issuing key. If more than one issuer may issue credentials of the same type, then they can register different issuing keys for this type or—less advisable—they can share issuing keys. Conceptually, pseudonyms are used as follows.

- (1) During *intro*, a MUD interacts with an issuer and with its observer. The new *source pseudonym* established between issuer and MUD is denoted ψ . The pseudonym used for the interaction with the observer is called *intro pseudonym* ψ_O . As part of this step, the issuer identifies the holder by means of a photo ID or by other biometric means, in order to link the digital pseudonym ψ to a legal identity of the holder. The observer needs to be involved in choosing the pseudonym ψ to guarantee it cannot be used by MUDs using different observers.
- (2) During *issue*, the issuer issues a credential to a source pseudonym ψ , which should only be successful if the MUD is supported by the same observer that was involved in introducing pseudonym ψ . The term “should” indicates that the issuer trusts in the observer during this transaction. The MUD ends up with an *intermediate pseudonym* ψ' that it may use for its own book keeping. The intermediate pseudonym may or may not be learnt by the observer.
- (3) During *show*, the verifier verifies a credential for some target pseudonym ϕ , which should only be successful if the MUD is supported by an observer that was involved earlier in getting a type t' credential issued. The term “should” indicates that the verifier trusts in the observer during this transaction. Observers enforce that holders comply to the holder authorization rules. In personal credential schemes for example, observers enforce that a certain type of credential can only be shown if the same human holder identifies to the observer as before when that observer was involved in getting issued a credential of the same type. In order to show its credential, the MUD uses its intermediate pseudonym.

Figure 5–4 on p.73 summarizes the pseudonyms involved.

Before providers will accept an observer within one of their customer's MUDs as a provider's agent safeguarding certain transactions within the MUD, the providers need to convince themselves that a customer, who may connect online to their provider infrastructure in fact employs an observer of the right kind. All the following is based on the idea that observers come from their manufacturer(s) with verified software installed, tamper detection mechanisms activated, and with one or more native sign-

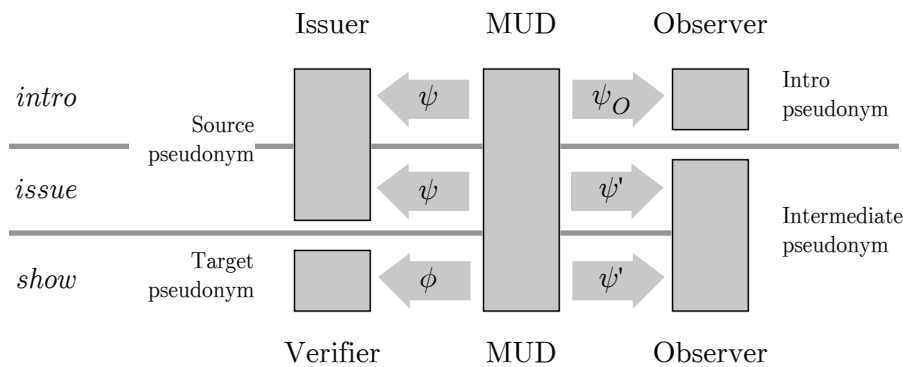


FIGURE 5-4 Overview of Pseudonyms

ing key(s) built-in, which can later be used to strongly authenticate an observer. Before observers leave the manufacturer, their native verifying keys are certified and posted to public key directories. We discuss important aspects of this approach:

How can issuers authenticate observers? In addition to the operations of issuing and showing a credential, a third operation (*intro*) is needed by which an issuer verifies an observer at the time of first contact. The observer uses its native signing key in order to authorize the MUD's request to introduce a pseudonym. The issuer verifies the response by looking up the respective native verifying key from the public key directory of the manufacturer. If the verification is successful, the issuer establishes the proposed source pseudonym for the MUD. The holder can later apply for a credential for this source pseudonym.

Why observers should use individual native keys? An observer's native key is a means to prove to an issuer that the observer is properly manufactured, runs the right software version and has its tamper resistance in place. Breaking an observer means to compromise its key and thus to compromise all observers that share the same key. If observers use individual native keys then their keys can also be revoked individually in case they are compromised. If some or all observers would share a native key, then all of them had to switch to a new native key in case any of them were broken.

Who generates native keys? Naturally, each observer generates its own private native key(s) after it is fully manufactured and its tamper resistance is in place. This is certainly the most reliable way of not releasing information about private keys. If justified by sufficient competition among manufacturers, then native keys could be pre-generated and downloaded into the observers during the manufacturing process. In either case, manufacturers have the information which public native keys belong to which of their observers. However, customers can always resort to using several observers and only one native key of each. As we will see in the final remark below, it is probably necessary for each MUD to have different observers available anyway. The generation of issuing and native keys is summarized in Figure 5-5 on p.74:

Who certifies native keys? It must be some authority trusted by the intended issuer(s). Since these issuers have to trust in the proper generation of the native keys themselves, which in turn implies

	Issuer	Observer (Manufacturer)
	For all types t that this issuer is authorized for :	
(1)	$(rk_t, pk_t) \leftarrow \text{genKey}(prek, t)$	$(rk_O, pk_O) \leftarrow \text{genKey}(prek)$
(2)	Certify and broadcast pk_t .	Certify and broadcast pk_O .

FIGURE 5-5 Setup of Issuer and Observer Keys

some trust in the manufacturers, the manufacturers are the natural candidates for certifying the native keys of their observers. If the issuers so require, second certifiers can be licensed.

How to maintain privacy despite individual native keys? If observers endorse introduction requests by individual native keys (explicit introduction), then all introduction requests of one MUD can be linked by the respective issuers. So the number of such introductions should be kept to a minimum. Alternatively, pseudonyms can be introduced implicitly (see Definition 5.1 on p.76 operation *show*) by re-using a target pseudonym as the next source pseudonym. This is what happens in practice most of the time. Usually, source pseudonyms need to be introduced explicitly to different issuers only if the issuers distrust each other. Otherwise, a credential of one of them can be used to implicitly introduce a pseudonym to the other. From a privacy point of view, mutually distrusting issuers are unlikely to link their information, and so explicit introductions are relatively safe. An alternative is to find a third party in which the mutually distrusting issuers both trust, and to obtain a credential from the third party in order to implicitly introduce pseudonyms to each of the mutually distrusting issuers.

5.2.2 General Definition

We give a definition for credential schemes in general and then identify each of the core schemes as a special case. This approach is motivated by the many commonalities of the core schemes. However, there are also significant differences between the core schemes. For example, personal credential schemes and coin credential schemes need specific additional operations compared to free credential schemes. Personal credential schemes require an operation to personalize the biometer of an observer with some initial biometric identity (usually called a *biometric template*), whereas coin credential schemes require instead an operation to identify an overshooting participant after the fact. Likewise, the core schemes require specific additional parameters in the external interfaces of the common operations; and the additional parameters cannot naturally be regarded as specializations of the existing parameters.

Essential in the following definition are the integrity and privacy requirements according to Section 2.2 on p.6. They will be formalized by the Definitions 5.2-5.6. We will state both kinds of requirements in the framework of cryptographic schemes as we have done for interactive proof-of-knowledge schemes (Section 4.1 on p.34) and blind signature schemes (Section 4.2 on p.51) before. We will see that this approach works, but we will also see that this approach somehow misses to define credential schemes at the highest level of abstraction reasonable. For proof-of-knowledge schemes it is natural that the definition talks about candidates and witnesses because they are essential quantities of proofs of knowledge. Without them, the security requirements of proof-of-knowledge schemes could simply not be stated. Our definition of credential schemes will contain pseudonyms and witnesses for pseudonyms as well. We simply have not found another way of expressing the privacy requirements of

credential schemes. Nevertheless, we feel that pseudonyms are more an auxiliary technical detail of some implementations of credential schemes, rather than an essential quantity that every credential scheme must have. This holds even more for pseudonym witnesses. In other words, we admit there may be implementations of credential schemes in the future that make no use of pseudonyms, let alone of witnesses for pseudonyms. Clearly, our definition will not capture such implementations. The problem is that we try to define integrity and confidentiality or privacy requirements at the same time.

Integrity requirements alone can be expressed quite adequately in temporal logics or process algebras (for example see Manna and Pnueli [151,152]). There is a rich theory of formally verifying implementations against integrity requirements. Very roughly, integrity requirements and implementations are represented as sets of event sequences. All of the event sequences representing the integrity requirement satisfy a certain integrity predicate. A general theorem says that any subset (of event sequences) of a security requirement is a secure implementation, simply because any event sequence of the subset also satisfies the integrity predicate mentioned above. In the area of cryptographic systems, Pfitzmann has applied a similar approach to specify various categories of digital signature systems [182,183], Pfitzmann and Waidner [189] have specified cryptographic payment systems, and Schunter [211] has developed specifications of fair exchange systems.

Far less is known about how to express confidentiality and privacy requirements at this level of abstraction because confidentiality and privacy requirements do not behave in a monotonous way as sketched above (see McLean [154]). In fact, confidentiality and privacy requirements are typically expressed by predicates that put an upper bound on the probability of some event, where the probability is taken over the random choices of all participants. If we translate this back to the level of abstraction of event sequences, it becomes clear that a confidentiality or privacy predicate does not apply to single event sequences, but rather to a large set of event sequences that evolve from participants making different internal choices.

One remarkable example of expressing integrity and confidentiality (or privacy) requirements at the same high level of abstraction has been given by Beaver [9,11] for the class of multi-party computations of functions. Unfortunately, it is not obvious if his technique of using a “trusted host” to express integrity and confidentiality (or privacy) requirements carries over to reactive systems, which have no beginning and end like a function computation.

In order not to entirely give up on a formal definition of credential schemes, we will resort to a definition at a lower level of abstraction, where pseudonyms (and pseudonym witnesses) remain visible. This unsatisfactory approach is also taken by Lysyanskaya, Rivest, Sahai and Wolf in [147]. We strongly encourage to seek for a more abstract way of defining credential schemes in particular and complex cryptographic systems in general.

Definition 5.1 Credential Scheme

A credential scheme with security parameter k , credential types $T = \{t_1, \dots, t_m\}$ ⁵ and recognition characteristic ($0 \leq FAR, FRR \leq 1$) consists of the following prekey generator, domains, operations and requirements:

5) The set T of credential types is a finite set of labels specifying the contents of credentials, e.g., driver’s licences, passports, etc.

GENERATE PREKEY $prek \leftarrow genPrekey(k)$

is a polynomial-time algorithm that on input the *security parameter* k outputs a *prekey* $prek$. The parameter k determines the issuers' security against forgery of credentials (usually under a complexity theoretic assumption).

DOMAINS

A credential scheme has a set $\{RK, PK, \Psi, W, V, make, make', C\}$ of 8 domain families and 2 optional domain families $\{TR, \mathcal{B}\}$ whose respective members are:

- The *private key* domains RK_{prek} .
- The *public key* domains PK_{prek} .
- The *pseudonym* domains Ψ_{prek} .
- The *pseudonym witness* domains W_{prek} , and the *pseudonym co-witness* domains V_{prek} .
- The *making functions* $make_{prek} : W_{prek} \rightarrow \Psi_{prek}$. A witness $w \in W$ is said to *make* a pseudonym $\psi \in \Psi$ iff $\psi = make(w)$.
- The *co-making functions* $make'_{prek} : \Psi_{prek} \times V_{prek} \rightarrow \Psi_{prek}$. A co-witness $v \in V$ is said to *co-make* a pseudonym $\phi \in \Psi$ with respect to a pseudonym $\psi \in \Psi$ iff $\phi = make(\psi, v)$.
- The *credential* domains C_{prek} .
- The *transcript* domains TR_{prek} , whose elements represent digital transcripts that verifiers receive in coin credential schemes every time a holder pays a coin to them.
- The *biometric identity* domains \mathcal{B}_{prek} whose elements represent biometric identities of human holders, such as thumbprints, irisprints, voiceprints, etc. (cf. Section 5.1.5 on p.69). These elements are written illustratively as $\ominus, \odot, \otimes, \dots$. Different faces just mean different variables. There is no hidden meaning implied by different expressions of the faces.

OPERATIONS:

A credential scheme has a set $\{genKey, intro, issue, show, show*, verify, persObs, recog, extract\}$ of 5 operations $genKey, intro, issue, show, verify$ and 4 optional operations $show*, persObs, recog, extract$ satisfying the following requirements: effectiveness, type unforgeability, and weak unlinkability.

The following operations are introduced by using the characteristic roles of credential schemes: manufacturers (M), issuers (I), mobile user devices (D), observers (O), and verifiers (V). Since the index $prek$ is omitted as usual in the sequel, the identifier V denotes a verifier and the domain of pseudonym co-witnesses. Nevertheless, it is always clear from the context what V means.

Generate Key $(rk_t, pk_t) \leftarrow genKey(prek, t)$

A probabilistic operation that takes a prekey and a credential type $t \in T$ and outputs a key pair consisting of a private key $rk_t \in RK$ and a public key $pk_t \in PK$. The public key pk_t is going to be certified by a legitimated organization and is afterwards published as a verifying key of type t credentials. As discussed in Section 5.2.1 on p.72, it can be convenient in practice to have more than one key pair for the same credential type t . Without loss of generality and to keep notation simpler, we assume in

the following there is only one key pair generated for each type t , and it is this key pair we denote as (rk_t, pk_t) .

Introduce Pseudonym Explicitly

$$({}^I[\psi], {}^D[\psi, \psi_O, v], {}^O[\psi_O, w]) \leftarrow intro([pk_O], {}^I[], {}^D[], {}^O[rk_O, \{\ominus\}])$$

A 3-party operation of issuer I , MUD D and observer O . Common input is the observer's (public) native key pk_O . In addition, O takes its private native key rk_O , and, if it is a biometric observer, a holder identity $\ominus \in \mathcal{B}$ as input. I and D output the shared pseudonym $\psi \in \Psi$, and D and O output the shared pseudonym $\psi_O \in \Psi$. Moreover, D outputs a co-witness $v \in V$, and O outputs a witness $w \in W$.

We say that D *explicitly introduces* pseudonym ψ (to I). On successful execution, I is convinced that the holder is using an authorized MUD. The issuer learns (one of) the public native keys pk_O of the observer. The MUD and the observer will later need their respective co-witness v and witness w in order to use ψ in further operations.

Issue $({}^D[\psi', \chi', v'], {}^O[w']) \leftarrow issue([pk_t], {}^I[rk_t, \psi], {}^D[\psi, \psi_O, v], {}^O[\psi_O, w])$

A 3-party operation of an issuer I , a MUD D and an observer O . Common input is an issuer's public keys $pk_t \in PK$ of credential type t . In addition, I takes a private key $rk_t \in RK$ and a source pseudonym $\psi \in \Psi$ shared with D . D and O take the shared pseudonym $\psi_O \in \Psi$, D takes a pseudonym co-witness $v \in V$, and O takes a pseudonym witness $w \in W$. On successful execution, D outputs a credential $\chi' \in C$ for pseudonym $\psi' \in \Psi$ and a pseudonym co-witness $v' \in V$ for ψ' . O outputs an intermediate witness $w' \in W$. We say that I *issues* χ' for the source pseudonym ψ' .

Show For New Target Pseudonym

$$({}^V[\phi, \chi'', \{\tau\}], {}^D[\phi, v'']) \leftarrow show([pk_t], {}^D[\psi', \psi_O, \chi', v'], {}^O[\psi_O, w', \{\ominus\}])$$

A 3-party operation of a MUD D , an observer O and a verifier V . Common input is an issuer's public key $pk_t \in PK$ of credential type t . In addition, D takes a pseudonym $\psi' \in \Psi$, a pseudonym $\psi_O \in \Psi$ shared with O , a credential $\chi' \in C$, and a pseudonym co-witness $v' \in V$. O takes a pseudonym witness $w' \in W$ and, if it is a biometric observer, a holder identity $\ominus \in \mathcal{B}$. On successful execution, the verifier outputs a target pseudonym $\phi \in \Psi$, a credential $\chi'' \in C$ and an optional transcript $\tau \in TR$. D outputs the same target pseudonym ϕ and a pseudonym co-witness v'' . We say that D *implicitly introduces* target pseudonym ϕ (to V) and *shows* credential χ'' for ϕ .

Verify $acc \leftarrow verify(pk_t, \psi, \chi)$

A deterministic operation that on input a public key $pk_t \in PK$ of type $t \in T$, a pseudonym $\psi \in \Psi$ and a credential $\chi \in C$ returns TRUE or FALSE. If it returns TRUE, we say that χ is a type t credential *valid* for pseudonym ψ .

OPTIONAL OPERATIONS

Show For Re-used Target Pseudonym

A MUD can use this operation in order to show a credential for a pseudonym that it has introduced

5 CREDENTIAL SCHEMES

before, either explicitly (using *intro*) or implicitly (using *show*). The MUD can show the credential for any target pseudonym that has been introduced by using the same shared input ψ_O .

$$(V[\phi, \chi'', \{\tau\}], D[\phi]) \leftarrow \text{show}^*([pk_t], D[\psi', \psi_O, \chi', v', v''], O[\psi_O, w', \{\ominus\}])$$

A 3-party operation of a MUD D , an observer O and a verifier V . Common input is an issuer's public key $pk_t \in PK$ of credential type t . In addition, D takes a pseudonym $\psi' \in \Psi$, a pseudonym $\psi_O \in \Psi$ shared with O , a credential $\chi' \in C$, and two pseudonym co-witnesses $v', v'' \in V$. O takes a pseudonym witness $w' \in W$ and, if it is a biometric observer, a holder identity $\ominus \in \mathcal{B}$. On successful execution, the verifier outputs a target pseudonym $\phi \in \Psi$, a credential $\chi'' \in C$ and an optional transcript $\tau \in TR$. D outputs the same target pseudonym ϕ . We say that D shows credential χ'' for the re-used target pseudonym ϕ , i.e., a pseudonym that D has introduced to I before.

Personalize Observer $(rk_O, pk_O) \leftarrow \text{persObs}(\text{prek}, \{\ominus\})$

A probabilistic operation provided only by observers. The observer takes a prekey, and if the observer is biometric, it also takes a *biometric template* $\ominus \in \mathcal{B}$ as input. The observer outputs a *native* key pair consisting of a private *native* key $rk_O \in RK$ and a public *native* key $pk_O \in PK$. (Native keys were first introduced by Cramer and Pedersen [80]). In addition, the observer writes the biometric template $\ominus \in \mathcal{B}$ onto its composition tape for further reference (see operation *recog* below). Each observer performs this operation at most once. Thus no observer can be re-personalized to use a second native key or use a second biometric template. Note that this operation combines all necessary initialization steps of an observer. In real life, observers will generate their native key pairs right after manufacturing, whereas their biometric templates can only be implanted after the observer has been embedded into its mobile user device and is distributed to a customer. By using the above operation *persObs*, we will consider an idealized initialization without such intermediate states.

Recognize

This operation is provided by biometric observers only. Each biometric observer is assumed to physically host exactly one biometer (see Section 5.1.5 on p.69). Therefore, operations provided by biometers are only available to the respective observers, not to issuers, verifiers or mobile user devices. Because of the one-to-one correspondence between observers and biometers, we will use O as a role identifier both for observers and their respective biometers.

$$O[\text{acc}] \leftarrow \text{recog}(O[\ominus])$$

The biometer takes an actual identity $\ominus \in \mathcal{B}$, compares it to the unique biometric template it was personalized for and returns a Boolean result.⁶ If it returns TRUE, we say that O recognizes \ominus .

Extract

Operation *extract* is used in coin and bond credential schemes only. It allows to detect if a MUD has shown a credential more often than it is entitled to.

6) We model the recognition characteristic of practical biometric recognition systems, i.e., false acceptance rate (FAR) and false rejection rate (FRR), by a probabilistic algorithm.

$$w \leftarrow \text{extract}(\tau)$$

A deterministic operation that on input a finite set $\tau \subseteq TR$ of transcripts (as output by *show*), returns a pseudonym witness $w \in W$ or the undefined output \perp . For typical applications like K -show coin credentials, the arguments are sets of $K + 1$ transcripts. The output \perp indicates that the argument reveals no witness and therefore does not identify any holder. This operation allows verifiers or their delegates to identify holders who have shown their credentials too often. The witness can then be used to recover the source pseudonym of the overshower (see Definition 5.5 on p.84). Linking that source pseudonym to a “real” identity is outside the scope of this definition.

These nine operations satisfy the following effectiveness requirements.

EFFECTIVENESS

For each instance of a credential scheme with all honest participants, each of the 5 protocols and 4 optional protocols are executed successfully.

For all correctly generated prekeys $prek \in [\text{genPrekey}(\mathbb{N})]$, consider a correctly executed key setup. Let (rk_O, pk_O) be the native key pair of observer O , and if O is biometric, let it be personalized to the biometric template \odot . Let (rk_t, pk_t) ($t \in T$) be an issuer’s key pair of credential type t . Then the following assertions (1)-(5) hold:

- (1) If a MUD D with observer O introduces a source pseudonym explicitly to an issuer I

$$(I[\psi], D[\psi, \psi_O, v], O[\psi_O, w]) \leftarrow \text{intro}([pk_O], I[], D[], O[rk_O, \{\odot\}]) ,$$

then the witness w of O makes the intro pseudonym $\psi_O = \text{make}(w)$, and the co-witness v of D co-makes the source pseudonym $\psi = \text{make}'(\psi_O, v)$ with respect to the intro pseudonym ψ_O .

- (2) If an issuer I issues a credential χ' successfully to D , then D can show χ' successfully to any verifier V . More precisely:

Let D have either introduced a source pseudonym ψ to I explicitly,

$$(I[\psi], D[\psi, \psi_O, v], O[\psi_O, w]) \leftarrow \text{intro}([pk_O], D[], O[rk_O, \{\odot\}]) , \quad (5.1)$$

or introduced a pseudonym ϕ implicitly by operation *show*:

$$(V[\psi, \bullet, \{\bullet\}], D[\psi, v]) \leftarrow \text{show}([\bullet], D[\bullet, \psi_O, \bullet, \bullet], O[\psi_O, w', \{\bullet\}]) .$$

(Note that we use output parameter names ψ instead of ϕ and v instead of v'' in order to meet the names of the analogous parameters above in external interface (5.1) on p.79. In addition, we suppress parameters that are not relevant in this context by the symbol ‘ \bullet ’.)

Then if I issues a credential to D (and O) as follows

$$(D[\psi', \chi', v'], O[w']) \leftarrow \text{issue}([pk_t], I[rk_t, \psi], D[\psi, \psi_O, v], O[\psi_O, w]) , \quad (5.2)$$

then the co-witness v' output by D co-makes the intermediate pseudonym $\psi' = \text{make}'(\psi_O, v')$ with respect to the intro pseudonym ψ_O .

- (3) If D (with help of O) uses its output from *issue* to show a new type t credential to V as follows

5 CREDENTIAL SCHEMES

$$(V[\phi, \chi'', \{\tau\}], D[\phi, v'']) \leftarrow show([pk_t], D[\psi', \psi_O, \chi', v'], O[\psi_O, w', \{\ominus\}]) ,$$

then V accepts, i.e., V does not abort the protocol, the co-witness v'' output by D co-makes the target pseudonym $\phi = make'(\psi_O, v'')$ with respect to the intro pseudonym ψ_O , and χ'' is a type t credential valid for pseudonym ϕ , i.e., $verify(pk_t, \phi, \chi'') = TRUE$.

OPTIONAL EFFECTIVENESS

- (4) (Applies only if a protocol $show^*$ is defined) Likewise, if D (with help of O) uses its output from *issue* to show a new type t credential to V for a target pseudonym used earlier as follows

$$(V[\phi, \chi'', \{\tau\}], D[\phi]) \leftarrow show^*([pk_t], D[\psi', \psi_O, \chi', v', v''], O[\psi_O, w', \{\ominus\}])$$

then the co-witness v'' output by D co-makes the target pseudonym $\phi = make'(\psi_O, v'')$ with respect to the intro pseudonym ψ_O , and χ'' is a type t credential valid for pseudonym ϕ , i.e., $verify(pk_t, \phi, \chi'') = TRUE$.

- (5) (Applies only in case of biometric observers) Unless a biometric observer O is personalized, it does not recognize any holder's identity, i.e., for all $\ominus \in \mathcal{B}$

$$recog^O[\ominus] = FALSE$$

As soon as an observer O is personalized ($persObs(prek, \ominus)$) to some holder identity $\ominus \in \mathcal{B}$, it recognizes this identity according to the following specification, where $FAR, FRR \in [0, 1]$ are constants:

$$Prob[recog^O[\ominus] = TRUE \mid \ominus \neq \ominus] \leq FAR ,$$

$$Prob[recog^O[\ominus] = FALSE \mid \ominus = \ominus] \leq FRR .$$

For example, we can describe a credential scheme that perfectly recognizes biometric identities by a *recognition characteristic* of $(FAR, FRR) = (0, 0)$. Another special case is a credential scheme that ignores biometric identities altogether. This is characterized by $(FAR, FRR) = (1, 0)$. ♦

Remarks: Why is the observer not checking the biometric identity of its holder in *issue* (note the missing face in the list of input parameters)? In general, issuers will issue credentials only for pseudonyms that have been introduced to them before. A pseudonym can be introduced *explicitly* by using *intro*, or *implicitly* by using *show*. In the latter case, the target pseudonym of the *show* operation becomes the source pseudonym in the following *issue* operation. In each case, the introduction of a pseudonym ψ includes a check of the biometric identity of the holder. Therefore, no second biometric check is required before issuing a credential for ψ .

What information is kept on the composition tapes of MUDs and observers? MUD (D) and observer (O) return a shared pseudonym ψ_O , a corresponding co-witness v and a witness w after an explicit introduction. The MUD writes ψ_O and v to its composition tape, and the observer writes ψ_O and w to its own composition tape. These values are needed every time the MUD needs to issue or show a credential for a pseudonym that evolved from this initial explicit introduction. In Mechanism 5.10 on p.111, the observer generates an auxiliary witness w^* for an auxiliary intro pseudonym ψ_O^* ,

in step (1) of *intro*, which are both written to the observer's and the MUD's composition tape, respectively.

Why is an extra operation *verify* needed? This operation allows a third party to check the validity of credentials offline. For example, credentials can be verified by a judge in order to achieve non-repudiation of origin.

5.3 Security Definitions of Credential Schemes

Our next goal is to define the characteristic security requirements of each category of credential schemes.

5.3.1 Overview of Security Requirements

Type Unforgeability: Even after verifying a polynomial number of credentials of type $t \in T$, it is infeasible for an attacking verifier to obtain a type t credential himself (Definition 5.2 on p.83).

Transfer Prevention: Whenever an attacking MUD successfully shows a type t credential after all type t credentials so far have been issued by using intact observers personalized to respective biometric identities \odot_1, \odot_2, \dots , then the attacking MUD must have available some biometric identity $\odot \in \{\odot_1, \odot_2, \dots\}$ (Definition 5.3 on p.83).

Overshow Prevention: After obtaining $k \in \mathbb{N}$ K -show type t credentials, an attacking MUD with an intact observer cannot successfully show type t credentials more than kK times, i.e., holders cannot overshadow (Definition 5.4 on p.84).

Overshow Detection: If the attacking MUDs, which host broken observers, have obtained $k \in \mathbb{N}$ K -show type t credentials and later show type t credentials more than kK times, then at least one of the attacking MUDs will be identified by means of operation *extract* (Definition 5.5 on p.84).

Schemes that allow overshadow detection may have an optional operation by which holders can defend themselves against false accusations of having overshadowed (*holder framing*). This requirement is not formalized further, but the coin credential construction given in Section 5.5 on p.110 has it. For more detail see Brands [34,35] and Pfitzmann, Waidner [189].

Holder Privacy: If an attacking issuer issues $n_I \in \mathbb{N}$ credentials and an attacking verifier verifies $n_V \in \mathbb{N}$ credentials, then the collaborating issuer and verifier cannot tell with better chance of success than pure guessing which of the n_V verifications correspond to which of the n_I issues. Further refinements of this informal notion of unlinkability are made in Definition 5.6 on p.87.

5.3.2 Overview of Attacker Models

The attacker model for all integrity requirements (see Table 5-3 on p.82) is polynomial-time (Section 3.3.2 on p.24); that for holder privacy is computationally unlimited (Section 3.3.2 on p.24). If issuers, verifiers and MUDs are dishonest, they are considered active attackers. For observers we distinguish four different attacker models:

- (i) *Intact Observer Assumption:* Intact observers honestly follow their protocols and resist any attempts to compromise or to modify their internally stored information until their internal information is destructed. If observers are equipped with biometers, the biometers also follow their pro-

5 CREDENTIAL SCHEMES

tocols honestly and resist any attempts of compromising or modifying the biometric templates they have stored internally.

- (ii) *Broken Observer Assumption*: All observers are properly personalized by honestly generating a native key pair. If the observers are equipped with biometers, they are properly personalized to a biometric template. After being personalized, the attacker may break one or more observers at any time and afterwards control them completely including to extract the private native key and/or the biometric template. Such observers are called *broken*.
- (iii) *Shielded Observer Assumption*: All observers are shielded by their hosts against out-of-band inflow and out-of-band outflow. They have only one communication channel, namely to their respective hosts. This is modeled by the read-only and write-only communication tape shared with their respective hosts.
- (iv) *Leaking Observer Assumption*: All observers are shielded by their hosts against out-of-band inflow, but may leak information to issuers and verifiers, for example, after they are lost or stolen or returned by the holder. Thus, in addition to the communication channels with their hosts, observers have additional unidirectional communication channels to the issuers and verifiers.

Assumptions (i) to (ii) are used for integrity requirements. Assumptions (iii) and (iv) are used for holder privacy requirements. Every combination of the former two and the latter two assumptions is meaningful, except the assumption that an observer is broken and still leaking. If a holder has broken its observer, he can effectively cut off any communication channel of his observer, by importing the observer's data into itself and using it as appropriate. We finally summarize in Table 5-3 on p.82 which security requirements make which assumptions about issuers/verifiers, MUDs, and observers.

Security Requirements		Assumption about . . .		
		Issuer / Verifier	MUD	Observer
Integrity Requirements	Type Unforgeability	honest/active ^a	active	broken
	Transfer Prevention	honest/none ^b	active	intact
	Overshow Prevention	honest/none	active	intact
	Overshow Detection	honest/none	active	broken
Holder Privacy Requirements	Unlinkability	active/active	honest	shielded, active
	Fail-Safe Unlinkability	active/active	honest	leaking, active

TABLE 5-3 Overview of Attacker Models for given Security Requirements

a. "active" is short for active attacker.

b. "none" is short for no assumption, i.e., can be active, passive or honest

5.3.3 Definitions of Integrity Requirements

Next we give informal definitions of the four important integrity requirements introduced in Section 5.3.1 on p.81. The participants under consideration are one issuer I , one verifier V , one MUD D , which may use a number of different observers O_1, \dots, O_L where $L \in \mathbb{N}$. In each of the four definitions, the issuer is honest, whereas the MUD is an active attacker. Thus in the following, we denote the MUD as \tilde{D} (read D crooked) and give it an untyped interface (see Section 3.3.2 on p.24: Active Attacker). The attacker models of the verifier and the one or more observers depend on the integrity

requirement according to Table 5-3 on p.82. In those definitions where the verifier or the observers are considered attackers, they are denoted as \tilde{V} or \tilde{O} , respectively. The privacy requirements will be defined in more detail in Section 5.3.4 on p.84.

We summarize the presumptions about the setup of any credential scheme, which hold regardless of which integrity requirement is defined and which participants are considered to be attackers. According to Section 3.3.3 on p.25, the prekey is always chosen honestly. For each credential type $t \in T$ there is one unique issuing key pair (rk_t, pk_t) . The public key pk_t is certified to be used for verifying type t credentials. Each observer is honestly personalized, i.e., each observer O_ℓ ($\ell \in [1, L]$) has generated a *native key pair*

$$(rk_{O_\ell}, pk_{O_\ell}) \leftarrow \text{persObs}^{O_\ell}[\text{prek}, \{\odot\}]$$

and, if they are biometric, their biometers have been personalized with a unique biometric template. (Note that this is in line with both the intact observer assumption (i) and the broken observer assumption (ii) in Section 5.3.2 on p.82.) The manufacturer of the observers has published the respective public native keys through an authentic channel, for example by means of a public key infrastructure. Afterwards, observers may or may not be broken depending on the attacker model under consideration.

Definition 5.2 Type Unforgeability

Informally, a credential scheme is *type-unforgeable* if an attacker coalition of a MUD \tilde{D} , a number of observers \tilde{O}_ℓ ($\ell \in [1, L]$) under the broken observer assumption and a verifier \tilde{V} who are given the verifying key pk_t and who may verify type t credentials, but have never issued one, cannot forge a valid type t credential on their own. Other MUDs that host honest observers may have been issued type t credentials, but the attacker coalition has no access to these honest observers. \blacklozenge

The following definition applies to personal and bond credential schemes.

Definition 5.3 Transfer Prevention

Informally, a credential scheme is *transfer preventing* if the following holds for any polynomial-time attacking MUD and any number $L \in \mathbb{N}$ of observers O_1, O_2, \dots, O_L , which have been personalized to respective biometric identities $\odot_1, \odot_2, \dots, \odot_L$ under the intact observer assumption. After credentials of type t were issued by using exactly the observers O_1, O_2, \dots, O_L , and afterwards the attacking MUD has successfully shown a type t credential, then the attacking MUD has used at least one of the biometric identities $\odot \in \{\odot_1, \odot_2, \dots, \odot_L\}$ while showing a type t credential.

Nothing is required about type t credentials that have been issued to observers that are broken before or after the issuing. Thus we allow attacking MUDs, for example, to pool or transfer credentials of type t they have received by using observers that were broken before or after the issuing.

In the following, we assume biometers with perfect recognition characteristics, i.e., $(FAR, FRR) = (0, 0)$. This way, we keep the definition focused on the cryptographic aspects, rather than complicating it with modelling statistical errors of the recognition mechanism. \blacklozenge

The following definitions apply to coin and bond credential schemes: If observers are assumed to be intact, we can expect to prevent overshooting because each observer can keep track of the number of

shows for each credential of its host MUD. We require this also for credentials that can be shown K times under different pseudonyms in an unlinkable way (Definition 5.4 on p.84). Also see the framework of definitions by Pfitzmann and Waidner [189].

Definition 5.4 *K-Overshow Prevention*

Informally, a credential scheme prevents K -overshowing ($K \in \mathbb{N}$) if, under the intact observer assumption, $k \in \mathbb{N}_0$ type t credentials cannot be shown more than kK times. ♦

If observers are broken, we can still expect to identify holders after they have overshown. In our framework of unlinkable credentials, holders can be identified only by their source pseudonyms because this is the only quantity that is specific to a holder and visible to the honest issuer. Even though broken observers have a unique biometric template or native key, these characteristics are never visible to the honest issuer. An immediate implication is the following. If there are several broken observers, we cannot define to identify the “right” overshower, because the honest participants of the scheme cannot distinguish individual attacking holders at all. A first definition has been given by Franklin and Yung [113] and later by Pfitzmann and Waidner [189].

Definition 5.5 *K-Overshow Detection*

Informally, a credential scheme detects K -overshowing ($K \in \mathbb{N}$) if, under the broken observer assumption, $k \in \mathbb{N}_0$ type t credentials cannot be shown more than kK times without revealing a source pseudonym of at least one of the attacking MUDs. ♦

5.3.4 *Definitions of Holder Privacy*

Before getting a credential issued, the holder needs to provide one of his pseudonyms to the issuer. This pseudonym may either be introduced explicitly by operation *intro*, or implicitly by the holder showing an enabling credential whose target pseudonym is then used as a source pseudonym for the credential to be issued. In the latter case, the issuer of the new credential acts only upon permission by the verifier of the enabling credential. Hence, an explicitly introduced pseudonym may develop into a tree-like composition of alternating instances of *issue* and *show* (Figure 5–6 on p.85). This tree-like

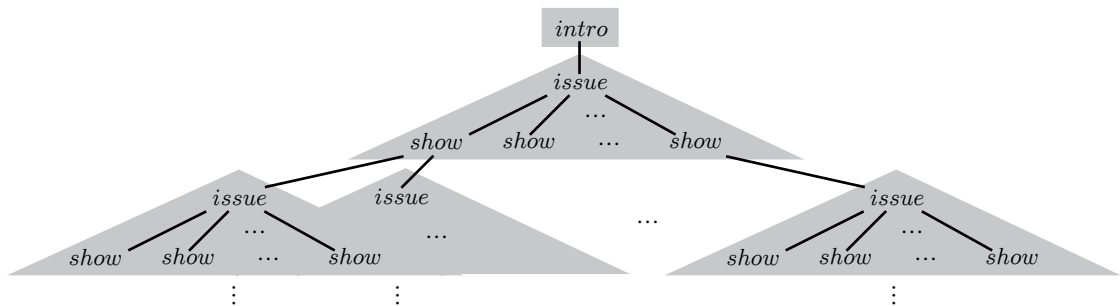


FIGURE 5–6 Tree of credentials evolving from an explicitly introduced root pseudonym

composition consists of

$$\text{||| } \left(\tilde{V}_n[\phi_n, \chi_n, \{\tau_n\}], D[\phi_n, v_n] \right) \leftarrow \text{show}([pk_t], D[\psi', \psi_{\tilde{O}}, \chi', v'], \tilde{O}[\psi_{\tilde{O}}, w', \{\ominus\}]) \text{)} \quad (5.3)$$

Likewise, $life_n$ is the following sequential composition of $show$, followed by $issue$, followed by a general composition of n executions of protocol $show$.

$$\begin{aligned} & life_n([pk_t], \tilde{I}[rk_t], D[\psi_0, \psi_{\tilde{O}}, \chi_0, v_0], \tilde{O}[\psi_{\tilde{O}}, w, \{\ominus\}], V_1[], V_2[], \dots, V_n[]) \\ &= \quad \left(\tilde{I}[\psi, \chi, \tau], D[\psi, v] \right) \leftarrow \text{show}([pk_t], D[\psi_0, \psi_{\tilde{O}}, \chi_0, v_0], \tilde{O}[\psi_{\tilde{O}}, w, \{\ominus\}]) \\ & \quad ; \quad \left(D[\psi', \chi', v'], \tilde{O}[w'] \right) \leftarrow \text{issue}([pk_t], \tilde{I}[rk_t], \psi, D[\psi, \psi_{\tilde{O}}, v], \tilde{O}[\psi_{\tilde{O}}, w]) \\ & \quad ; \quad \left(\tilde{V}_1[\phi_1, \chi_1, \{\tau_1\}], D[\phi_1, v_1] \right) \leftarrow \text{show}([pk_t], D[\psi', \psi_{\tilde{O}}, \chi', v'], \tilde{O}[\psi_{\tilde{O}}, w', \{\ominus\}]) \\ & \quad \text{||| } \left(\tilde{V}_2[\phi_2, \chi_2, \{\tau_2\}], D[\phi_2, v_2] \right) \leftarrow \text{show}([pk_t], D[\psi', \psi_{\tilde{O}}, \chi', v'], \tilde{O}[\psi_{\tilde{O}}, w', \{\ominus\}]) \\ & \quad \quad \quad \vdots \\ & \quad \text{||| } \left(\tilde{V}_n[\phi_n, \chi_n, \{\tau_n\}], D[\phi_n, v_n] \right) \leftarrow \text{show}([pk_t], D[\psi', \psi_{\tilde{O}}, \chi', v'], \tilde{O}[\psi_{\tilde{O}}, w', \{\ominus\}]) \text{)} \quad (5.4) \end{aligned}$$

Neither of the protocols $Life_n$ and $life_n$ returns an output because we will only be interested in the views of certain participants in these protocols.

In each member protocol of $Life_n$, i.e., $intro$, $issue$ and $show$, the honest MUD D uses the same attacking observer \tilde{O} , which is personalized to a native key pair $(pk_{\tilde{O}}, rk_{\tilde{O}})$. If the observer \tilde{O} contains a biometer, it is personalized to a biometric template \ominus and has a perfect recognition characteristic, i.e., $(FAR, FRR) = (0, 0)$. The attacking observer \tilde{O} is assumed to be either shielded or leaking, depending on which definition uses the protocol $Life_n$. The same holds for the protocol $life_n$ and its member protocols.

We are going to define two types of unlinkability in more detail. Issue-wise fail-safe unlinkability is achieved if $life_n$ blocks any in-band outflow from the observer. Show-wise fail-safe unlinkability is achieved if $life_n$ also blocks in-band inflow to the observer, which, under the leaking observer assumption, could be revealed afterwards to the issuer and verifiers through an out-of-band communication channel (*out-of-band outflow*) and may thus help them to link their views. We define both types of unlinkability against computationally unlimited attacking issuers and verifiers and computational observers.

After presenting the definitions, we will discuss why it is appropriate not to deal with out-of-band inflow in the definitions themselves, but rather keep it outside of the definition by using the shielded and leaking observer assumptions.

Definition 5.6 Unlinkability

SHOW-WISE FAIL-SAFE UNLINKABILITY

A credential scheme is *show-wise fail-safe* unlinkable if the following two conditions hold:

- (1) For all $n \in \mathbb{N}$, the honest MUD of $Life_n$ as well as that of $life_n$ blocks any in-band inflow to the observer, i.e., any information transmitted to the observer during $intro$, $issue$, $show$ or $show^*$

beyond the fact that *some* pseudonym is introduced and/or *some* type t credential is being issued or shown.

(2) For all $n \in \mathbb{N}$, the honest MUD of $Life_n$ as well as that of $life_n$ blocks any in-band outflow from the observer show-wisely, i.e., for each prekey $prek \in (genPrekey(\mathbb{N}))$ and each type $t \in T$ there are constants $\kappa_{Life}, \kappa_{life} \in \mathbb{N}$, which may depend on the prekey $prek$ and the type t , such that for each $n \in \mathbb{N}^7$ the following two statements hold:

(2a) Statement for $Life_n$: For each

- attacking issuer \tilde{I} of $Life_n$,
- n -tuple of not necessarily distinct attacking verifiers $\tilde{V}_1, \dots, \tilde{V}_n$,
- $(n+1)$ -tuple of not necessarily distinct attacking observers $\tilde{O}_0, \tilde{O}_1, \dots, \tilde{O}_n$ under the leaking observer assumption,
- issuing key pair $(rk_t, pk_t) \in [genKey(prek, t)]$,
- $(n+1)$ -tuple of native key pairs $(rk_{\tilde{O}_0}, pk_{\tilde{O}_0}), \dots, (rk_{\tilde{O}_n}, pk_{\tilde{O}_n})$ where $(rk_{\tilde{O}_i}, pk_{\tilde{O}_i}) \in [persObs(prek, \{\oplus_i\})]$ for all $i \in [0, n]$,
- $(n+1)$ -tuple (E_0, E_1, \dots, E_n) of successful executions of $Life_n$, where E_i is an execution of

$$Life_n([pk_t], \tilde{I}, [pk_{\tilde{O}_i}, rk_t], D, [\], \tilde{O}, [pk_{\tilde{O}_i}, rk_{\tilde{O}_i}, \{\oplus_i\}], \tilde{V}_1[\], \tilde{V}_2[\], \dots, \tilde{V}_n[\]) , \quad (5.5)$$

and the respective views of \tilde{I} , $\tilde{V}_1, \dots, \tilde{V}_n$ and $\tilde{O}_0, \dots, \tilde{O}_n$ on D are denoted as in Table 5-5 on p.87.

Attacker	E_0	E_1	...	E_n
Issuer \tilde{I}	$View_0^{\tilde{I}}$	$View_1^{\tilde{I}}$...	$View_n^{\tilde{I}}$
Verifier \tilde{V}_1	$View_0^{\tilde{V}_1}$	$View_1^{\tilde{V}_1}$...	$View_n^{\tilde{V}_1}$
\vdots	\vdots	\vdots	\ddots	\vdots
Verifier \tilde{V}_n	$View_0^{\tilde{V}_n}$	$View_1^{\tilde{V}_n}$...	$View_n^{\tilde{V}_n}$
Observers $\tilde{O}_0, \dots, \tilde{O}_n$	$View_0^{\tilde{O}_0}$	$View_1^{\tilde{O}_1}$...	$View_n^{\tilde{O}_n}$

TABLE 5-5 Views on MUD D resulting from n executions of $Life_n$

There are exactly κ_{Life} internal choices $\rho_1, \dots, \rho_{\kappa_{Life}}$ for the MUD D in $Life_n$ with inputs according to external interface (5.5) on p.87 such that the attacking participants $\tilde{I} ||| \tilde{O}_0$ and $\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_n$ each obtain the respective given views on D

$$(View_0^{\tilde{I}}, View_0^{\tilde{O}}, View_1^{\tilde{V}_1}, View_2^{\tilde{V}_2}, \dots, View_n^{\tilde{V}_n}) \quad (5.6)$$

7) This index n may be subject to further restrictions due to certain properties of the credential mechanism. For example, a k -overshoot detecting or preventing coin credential mechanism is shown by an honest MUD no more often than k times. Hence, we need to consider $Life_n$ and $life_n$ only for $n \leq k$.

5 CREDENTIAL SCHEMES

if the MUD uses any $\rho \in \{\rho_1, \dots, \rho_{\kappa_{Life}}\}$ as its internal choice.

(2b) Statement for $life_n$: For each

- attacking issuer \tilde{I} of $life_n$,
- n -tuple of not necessarily distinct attacking verifiers $\tilde{V}_1, \dots, \tilde{V}_n$,
- $(n+1)$ -tuple of not necessarily distinct observers $\tilde{O}_0, \tilde{O}_1, \dots, \tilde{O}_n$ under the leaking observer assumption,
- issuing key pair $(rk_t, pk_t) \in [genKey(prek, t)]$,
- input $(\psi_0, \psi_{\tilde{O}_i}, \chi_0, v_0)$ of MUD D and input $(\psi_{\tilde{O}_i}, w_i, \{\ominus_i\})$ of observer \tilde{O}_i in $life_n$ (for $i = 0, \dots, n$) such that
 - (i) the intro pseudonym $\psi_{\tilde{O}_i}$ is made by witness w_i ,
 - (ii) the intro pseudonym $\psi_{\tilde{O}_i}$, the intermediate pseudonym ψ_0 and the witness v_0 are related as follows: $\psi_0 = make'(\psi_{\tilde{O}_i}, v_0)$,
 - (iii) the signature χ_0 is valid for the intermediate pseudonym ψ_0 with respect to pk_t ,
- $(n+1)$ -tuple (e_0, e_1, \dots, e_n) of successful executions of $life_n$, where e_i is an execution of

$$life_n([pk_t], [\tilde{I}[rk_t], D[\psi_0, \psi_{\tilde{O}_i}, \chi_0, v_0], \tilde{O}_i[\psi_{\tilde{O}_i}, w, \{\ominus_i\}], \tilde{V}_1[], \tilde{V}_2[], \dots, \tilde{V}_n[]]) , \quad (5.7)$$

and the respective views of $\tilde{I}, \tilde{V}_1, \dots, \tilde{V}_n$ and $\tilde{O}_0, \dots, \tilde{O}_n$ on D are listed in Table 5-6 on p.88.

Attacker	e_0	e_1	...	e_n
Issuer \tilde{I}	$view_0^{\tilde{I}}$	$view_1^{\tilde{I}}$...	$view_n^{\tilde{I}}$
Verifier \tilde{V}_1	$view_0^{\tilde{V}_1}$	$view_1^{\tilde{V}_1}$...	$view_n^{\tilde{V}_1}$
\vdots	\vdots	\vdots	\ddots	\vdots
Verifier \tilde{V}_n	$view_0^{\tilde{V}_n}$	$view_1^{\tilde{V}_n}$...	$view_n^{\tilde{V}_n}$
Observers $\tilde{O}_0, \dots, \tilde{O}_n$	$view_0^{\tilde{O}_0}$	$view_1^{\tilde{O}_1}$...	$view_n^{\tilde{O}_n}$

TABLE 5-6 Views on MUD D resulting from n executions of $life_n$

There are exactly κ_{life} internal choices $\rho_1, \dots, \rho_{\kappa_{life}}$ for the MUD D in $life_n$ with inputs according to (5.7) on p.88 such that the attacking participants $\tilde{I} ||| \tilde{O}_0$ and $\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_n$ each obtain the respective given views on D

$$(view_0^{\tilde{I}}, view_0^{\tilde{O}_0}, view_1^{\tilde{V}_1}, view_2^{\tilde{V}_2}, \dots, view_n^{\tilde{V}_n}) \quad (5.8)$$

if D uses any $\rho \in \{\rho_1, \dots, \rho_{\kappa_{life}}\}$ as its internal choice. ◆

Remarks: Issue-wise unlinkability can be defined in a similar fashion. Since the views of different verifiers are not required to be unlinkable, only two executions of $Life_n$ and $life_n$ have to be considered, and the tuples of views (5.6) on p.88 and (5.8) on p.88 had to be replaced respectively by

$(View_{\tilde{I}_0}, View_{\tilde{O}}, View_{\tilde{V}_1}, View_{\tilde{V}_2}, \dots, View_{\tilde{V}_n})$ and $(view_{\tilde{I}_0}, view_{\tilde{O}}, view_{\tilde{V}_1}, view_{\tilde{V}_2}, \dots, view_{\tilde{V}_n})$.

A complete definition also needs to address protocol $show^*$, but is omitted here.

We shortly reflect the subtleties of defining fail-safe unlinkability (leaking observer assumption) and non-fail-safe unlinkability (shielded observer assumption).

Under the leaking observer assumption (fail-safe unlinkability), observers could be equipped for example with (undocumented) radio receivers in order to time stamp each transaction and keep a log file about them (out-of-band inflow). The radio receivers might pick up broadcast signals from satellites or cell phone networks, etc., which the issuers and verifiers can receive as well. After releasing the log file to the issuers and verifiers, most of their views on the host of the leaking observer could be linked pretty accurately by corresponding time-stamps.

Since out-of-band inflow works independently of the actual *issue* and *show* protocols, it cannot be ruled out by a definition about *issue* and *show* anyway. Therefore, allowing the chance of out-of-band inflow cannot be regarded as a flaw of the above definition. In practice, additional countermeasures need to be taken against out-of-band inflow, for example, MUDs could shield their observers from electromagnetic radiation and the processes of designing, manufacturing, and retiring of observers could be reviewed by independent security test laboratories.

Under the shielded observer assumption (non-fail-safe unlinkability), out-of-band inflow may occur, but out-of-band outflow is precluded. Still, in practice, there is a chance of in-band outflow because observers could use individual abortion characteristics in order to leak information to issuers and verifiers. Consider a scenario where a MUD needs its observer in each execution of *issue* and *show*. Some devious observer might refuse each request coming in from its host MUD the first time, but respond correctly to each repeated request. Thus the MUD must always abort the first attempt of getting a credential issued, and so must it abort every first attempt of showing a credential. Certainly an annoying observer from the holder's perspective. The issuer and verifier, however, might detect this characteristic protocol abortion and conclude that they have interacted with the same observer and therefore with the same MUD and holder. Other observers might run different abortion characteristics. A credential scheme that allows this class of attacks may well satisfy the proposed notion of unlinkability, because these attacks make use of invalid views, i.e., views on protocols that are aborted prematurely, which are ignored by the definition. This appears justified because any abortion of the observer (whether systematic or not) is detectable by the MUD. Thus the holder may decide at any time that an observer is no longer trustworthy and shall be replaced.

5.3.5 Interrelations between Integrity and Privacy Definitions

The integrity requirements (Definition 5.2 on p.83 through 5.5) and privacy requirements (Definition 5.6 on p.87) are orthogonal to a large extent. An open question is whether a K overshower of a show-wise unlinkable scheme can be identified, when $K > 1$. This appears difficult to achieve because unlinkability requires that MUDs can show each credential K times under different target pseudonyms, which makes all shows, i.e., the authorized and the unauthorized, "indistinguishable" to the verifiers, so that no verifier (coalition) can keep track of the number of shows.

If such schemes exist at all, the complexity of brute-force identifying a K overshower of a show-wise unlinkable scheme is exponential in K . Since all shows belonging to the same coin credential are unlinkable by the verifier, he can run *extract* on every $(K + 1)$ -subset out of the whole pool of, say, n transcripts obtained, and see whether they reveal a witness or not. This amounts to

$$\binom{n}{K+1}/2 = O(n^K) \quad (5.9)$$

checks on average. This is probably prohibitively expensive even for small values of K . However, there may be show-wise unlinkable credential schemes that have a smarter procedure of identifying the attacker than exhaustive search.

In contrast, identifying a K overshower in an issue-wise unlinkable scheme can be quite efficient, for example, if all shows of one credential are for the same target pseudonym. Then it takes only one application of *extract* in addition to finding the $K + 1$ linked transcripts. For example, the coin credential scheme by Brands [34,35] uses this approach.

5.4 Personal Credential Schemes

First, a definition is given that singles out a special case of credential schemes (Section 5.2.2 on p.74) as personal credential schemes. The following efficient cryptographic mechanism is joint work with Birgit Pfitzmann. It builds on the Chaum-Pedersen(3) Signature Mechanism (Section 4.2.2 on p.55) and ElGamal encryption [99].

5.4.1 Definition

Definition 5.7 Personal Credential Scheme

A personal credential scheme with security parameter k and credential types T is a type-unforgeable, transfer preventing and show-wise fail-safe unlinkable credential scheme with credential types T and recognition characteristic $(FAR, FRR) = (0, 0)$.⁸ The domain of transcripts and the operation *extract* are not defined for a personal credential scheme. ◆

5.4.2 The Cryptographic Mechanism in a Nutshell

In the following Mechanism 5.8 on p.92, mobile user devices are equipped with biometric observers (Section 5.1.5 on p.69). The main idea is simple: In order to issue a credential of type t to a MUD, the issuer passes it an encrypted signing key rk_t , which can be decrypted only by the observer of that MUD. The encryption mechanism is based on standard ElGamal encryption [99] and is done such that during the *issue* operation:

- the MUD can make sure that the issuer encrypts nothing but the private signing key rk_t , but
- the MUD does not get any information about the private issuing key other than its type.

By using its observer, the MUD can then show credentials of type t many times. Before a *show* operation, the MUD can choose a target pseudonym ϕ , and its observer provides a blind signature for ϕ if

8) Weaker personal credential schemes can be defined by allowing FAR and FRR positive values.

it recognizes the biometric identity of its holder successfully. Then the MUD passes ϕ and the signature to the verifier.

Each target pseudonym ϕ of a MUD serves in turn as a new encryption key, and the corresponding signature χ serves as a public key certificate for ϕ in the traditional sense. Hence, showing a credential provides the verifier with a certified encryption key ϕ that the verifier can use for issuing a new credential to the MUD. Target pseudonyms are chosen jointly by a MUD and its observer in such a way that

- the MUD can get credentials only for properly chosen target pseudonyms, and
- the verifiers do not get any information about the respective observers' contributions to these target pseudonyms.

We are going to use an extended ElGamal encryption mechanism for issuing credentials. Consider a discrete log setting $(p, q) \in dls_k$ and two generators g_1, g_2 of G_q . The verifier chooses a decryption key $(x_1, x_2) \in \mathbb{Z}_q^2$ and computes the corresponding public encryption key $y \leftarrow g_1^{x_1} g_2^{x_2} \bmod p$. A sender encrypts a message $m \in G_q$ by choosing an exponent $\alpha \in \mathbb{Z}_q^*$ and computing the ciphertext as follows:

$$\begin{aligned} (c_1, c_2) &\leftarrow (g_1^\alpha \bmod p, g_2^\alpha \bmod p) \\ c_3 &\leftarrow m \cdot y^\alpha \bmod p \end{aligned} \quad (5.10)$$

After sending the ciphertext (c_1, c_2, c_3) to the recipient, the plaintext is recovered by the following calculation

$$m = \frac{c_3}{c_1^{x_1} c_2^{x_2}} \bmod p \quad (5.11)$$

Standard ElGamal uses only a single value decryption key (x_1) , a single generator (g_1) , and a ciphertext of only two components, such as (c_1, c_3) . The extended ElGamal encryption mechanism above allows the sender to use a public encryption key y that is represented with respect to two generators, namely g_1, g_2 . This is important in the following Personal Credential Mechanism, because the public encryption keys will be pseudonyms that are represented with respect to two generators such that they can also serve as messages to be signed by the Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55.

5.4.3 Use of Pseudonyms, Signatures, Witnesses and Co-Witnesses

Throughout the life cycle of a MUD with its observer, a sequence of pseudonyms and corresponding witnesses and co-witnesses unfolds. The MUD keeps the co-witnesses, while the observer keeps the witnesses. During operation *intro*, the observer generates a witness w and the MUD generates a co-witness v (see Definition 5.1 on p.76). Together, they compute the source pseudonym ψ matching w and v . During operation *issue*, this source pseudonym and its witness and co-witness are used for second purposes, namely encryption and decryption, as explained in the following Table 5-7 on p.92

After a credential has been successfully issued to a MUD for target pseudonym ψ , the MUD can derive a fresh pseudonym ϕ by using a new co-witness v' , while its observer re-uses the witness w it

5 CREDENTIAL SCHEMES

Operation	Parameter	Purpose and use
<i>intro</i>	Source pseudonym ψ	Fresh source pseudonym for the MUD.
	Signature χ_O for ψ	Will serve as a public key certificate for ψ in <i>issue</i> .
	MUD's co-witness v for source pseudonym ψ	w and v serve as secrets to identify the MUD with its observer against the issuer during <i>intro</i> .
	Observer's witness w for intro pseudonym ψ_O and for source pseudonym ψ at the same time.	
<i>issue</i>	Source pseudonym ψ	used by the issuer as El-Gamal encryption key to encrypt signing key rk_t .
	MUD's co-witness v for source pseudonym ψ	used by the MUD to transform ciphertext of the issuer into ciphertext for its observer.
	Observer's witness w for source pseudonym ψ	used by the observer as El-Gamal decryption key to recover rk_t .
	MUD's intermediate pseudonym ψ' matches MUD's co-witness v' and observer's witness w	used by the MUD towards its observer.
	Observer's output witness w'	contains the decrypted signing key rk_t .

TABLE 5-7 Use of Pseudonyms and Witnesses During *Intro* and *Issue*

has generated for the initial source pseudonym ψ . The MUD and its observer use their respective co-witness and witness in the same way as during operation *intro* (see Table 5-8 on p.92).

Operation	Parameter	Purpose and use
<i>show</i>	Intermediate pseudonym ψ'	used by the MUD toward its observer.
	Fresh target pseudonym ϕ	used by the MUD toward the verifier.
	Signature χ'' for ϕ	serves as a public key certificate for ϕ .
	MUD's co-witness v'' for target pseudonym ψ	w and v serve as secrets to identify the MUD with its observer against the issuer during <i>intro</i> .
	Observer's witness w for intro pseudonym ψ_O and for target pseudonym ϕ .	
	Observer's input witness w'	used by observer to produce blind signature for target pseudonym ϕ .

TABLE 5-8 Use of Pseudonyms and Witnesses During *Show*

By showing a credential for target pseudonym ϕ , the MUD has implicitly introduced a new source pseudonym to the verifier. Now, the verifier can turn around and become an issuer. The verifier can re-use the target pseudonym ϕ as new source pseudonym, and issue a credential of some other type t' to the MUD upon request. In doing so, the issuer now uses $\psi = \phi$ as an El-Gamal encryption key according to Table 5-7 on p.92 and the observer still uses the same witness w as its decryption key to recover the new signing key $rk_{t'}$.

5.4.4 Personal Credential Mechanism

Mechanism 5.8 Personal Credential Mechanism (PC)

Let $(genPrekey^{(CP)}, genKey^{(CP)}, signBlind, verify)$ be the Chaum-Pedersen(3) Signature Mechanism of Section 4.2.2 on p.55. The PC Mechanism for credential types $T = \{t_1, \dots, t_m\}$ is as follows.

GENERATE PREKEY $((P, p), (p, q), g, g_1, g_2, h) \leftarrow \text{genPrekey}(k)$

Pick a double discrete log setting $((P, p), (p, q)) \in_R \text{ddls}$ (Definition 3.19 on p.29) uniformly at random from $\text{ddls}_{k, \ell}$ for some appropriate $\ell \leq k$. Choose three generators g, g_1, g_2 of G_q as well as a generator h of G_p .

DOMAINS

- The signing key domains are $RK_{\text{prek}} = G_q \setminus \{1\}$ and the verifying key domains are $PK_{\text{prek}} = G_p \times G_q$. When a public key is certified by an issuer to represent some type $t \in T$, then we write the respective keys as rk_t and pk_t , respectively. Issuers neither certify different public keys nor different private keys to represent the same type of credential.
- The pseudonym domains are $\Psi_{\text{prek}} = M^{(\text{CP})} = G_q$. They are at the same time the message domains of the Chaum-Pedersen(3) Signature Mechanism.
- The pseudonym witness domains are $W_{\text{prek}} = \mathbb{Z}_q^2 \setminus \{(0, 0)\}$, the blinder domains are $\Omega_{\text{prek}}^{(\text{CP})} = \mathbb{Z}_q^*$ and the co-witness domains are $V_{\text{prek}} = \Omega_{\text{prek}}^{(\text{CP})} \times W_{\text{prek}}$. We write co-witnesses as $v = (\omega, u)$.
- The making functions are inherited from $CEG(2)$: $\text{make}(w) = g_1^{w_1} g_2^{w_2} \bmod p$.
- The co-making functions are inherited from $ECEG(2)$: $\text{make}'(\psi, (\omega, u)) = \psi^\omega g_1^{u_1} g_2^{u_2} \bmod p$.
- The credential domains are $C_{\text{prek}} = \Sigma_{\text{prek}}^{(\text{CP})}$. They are also the signature domains of the Chaum-Pedersen(3) Signature Mechanism.
- The optional transcript domains: TR_{prek} are not defined.

OPERATIONS

Generate Key

$$(rk_t, pk_t) \leftarrow \text{genKey}(\text{prek}, t)$$

<i>genKey</i>	Issuer
(1)	Choose $x \in_R \mathbb{Z}_q^*$,
(2)	$rk_t \leftarrow g^x \bmod p$ { x will not be needed afterwards} ,
(3)	$pk_t = (\underline{pk}_t, \overline{pk}_t) \leftarrow (h^{rk_t} \bmod P, g^{rk_t \bmod q} \bmod p)$.

FIGURE 5-7 Generating an Issuer Key Pair

For easier reference, the first and second component of pk_t are denoted $\underline{pk}_t, \overline{pk}_t$, respectively.

Verify

This algorithm is inherited from the Chaum-Pedersen(3) Signature Mechanism in Section 4.2.2 on p.55.

PersObs

The observer generates its native key pair and, if it is biometric, it writes the biometric template ☺ provided as input to its composition tape.

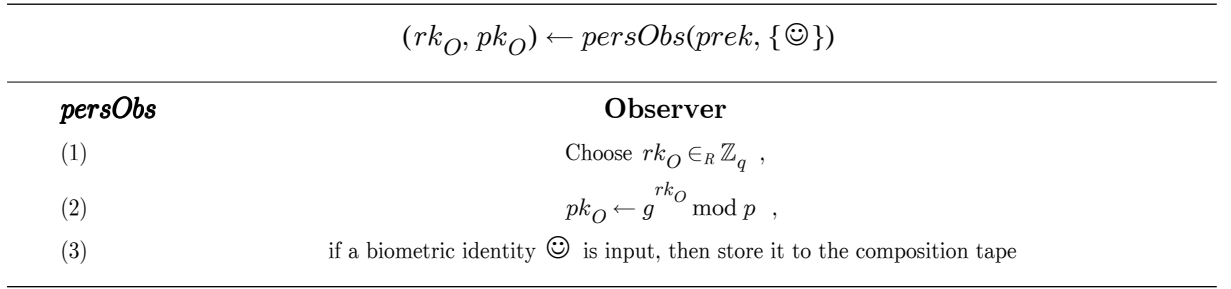


FIGURE 5-8 Personalizing an Observer

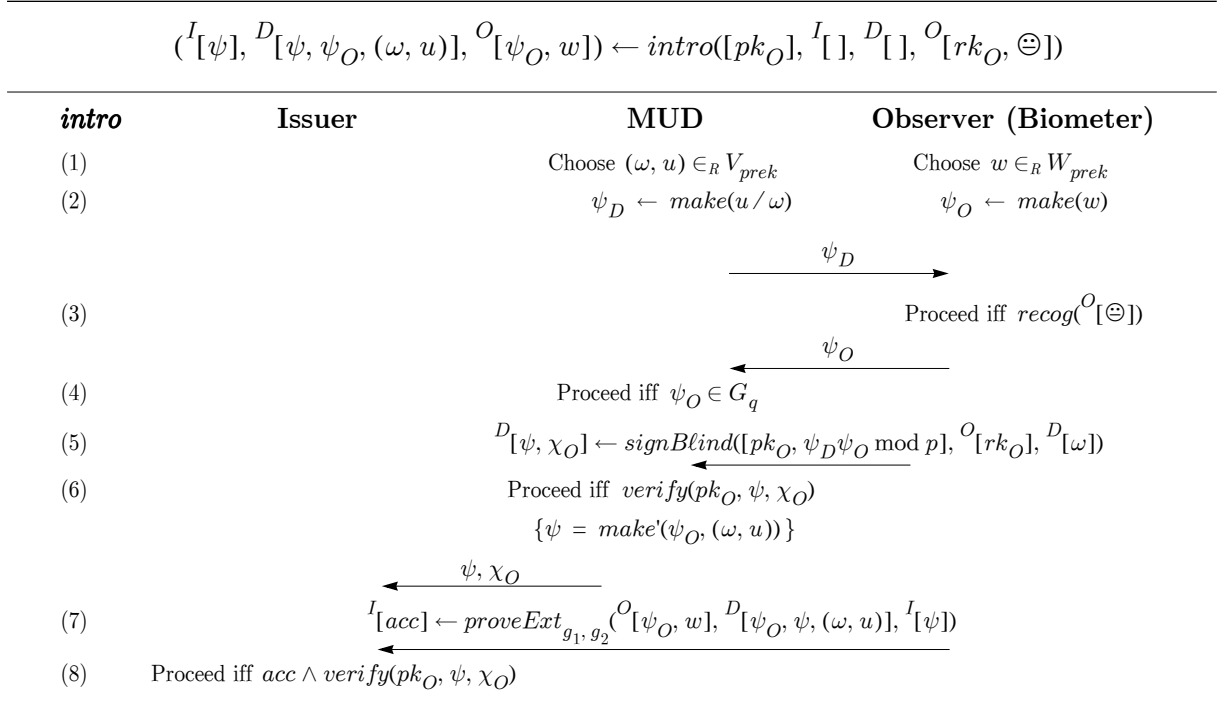
Introduce Pseudonym Explicitly


FIGURE 5-9 Introducing a Pseudonym

The observer takes its native signing key rk_O and the biometric identity \ominus of the actual user as input. Common input is the observer's native verifying key $pk_O = g^{rk_O} \bmod p$. The MUD D chooses a co-witness $(\omega, u) \in_R V_{prek}$ uniformly at random, and the observer chooses a witness $w \in_R W_{prek}$ uniformly at random (step (1)). The MUD makes its partial pseudonym ψ_D from the co-witness (ω, u) , and the observer O makes its partial pseudonym ψ_O from the witness w (step (2)). D sends its partial pseudonym ψ_D to O . If O can recognize the actual biometric identity \ominus through its biometric sensor, then it returns its partial pseudonym ψ_O to the MUD in step (3). If the MUD finds the observer's intro pseudonym ψ_O to live in G_q (step (4)), then the MUD executes member protocol $signBlind$ with the observer in order to obtain a blind signature χ_O from the observer for the source pseudonym ψ in step (5). If the MUD has received a valid signature (step (6)) it sends the source pseudonym together with the signature χ_O to the issuer. In step (7), the MUD and the observer con-

vince the issuer by an extended proof of knowledge that they together know a witness for the source pseudonym ψ . Finally the issuer verifies the signature against the native verifying key pk_O in step (8).

After the protocol is successfully terminated, the observer writes the pair (ψ_O, w) into a local list to be used later to run the protocols *issue* and *show*. In protocol *issue*, the observer's witness w will serve as a decryption key, and the partial pseudonym ψ_O as the corresponding encryption key. ψ_O is blinded by the MUD so to become ψ , and the signature χ_O serves as a public key certificate for ψ .

Issue Credential

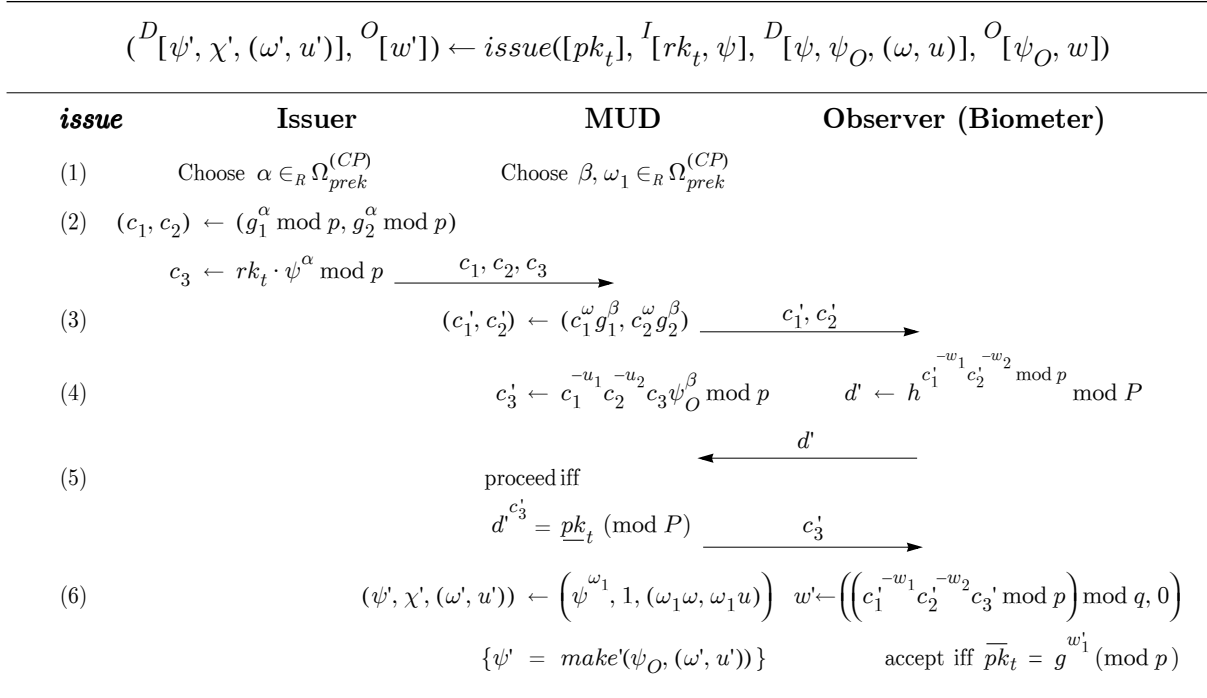


FIGURE 5-10 Issuing a Personal Credential

If a MUD D wants to get a type t credential issued, then D looks up a pseudonym ψ_O that it has used before in order to introduce a pseudonym ψ . D tells the issuer to use pseudonym ψ and its observer to use ψ_O . In addition, D tells the issuer and its observer which type t of credential it requests. The observer looks up from its own local lists the witness w it has used when the intro pseudonym ψ_O was introduced. The issuer checks if the source pseudonym ψ has been introduced to the issuer before; either explicitly by *intro* or implicitly by *show*.

The issuer and the MUD each make a respective random choice α and β , in step (1). In addition, the MUD chooses a blinder $\omega_1 \in_R \Omega_{prek}^{(CP)}$ in order to prepare its intermediate pseudonym ψ' . The issuer computes in step (2) three values c_1, c_2 and c_3 similar to standard ElGamal encryption. The MUD transforms (c_1, c_2) into (c'_1, c'_2) and sends it to the observer in step (3). Then the MUD transforms c_3 into c'_3 , while the observer computes a response d' in step (4) and returns it to the MUD. In step (5), the MUD verifies if the issuer has indeed encrypted the discrete logarithm rk_t of \underline{pk}_t with respect to h in step (2). Only if this verification succeeds, the issuer sends c'_3 to the observer. Now, the

observer can recover the encrypted signing key $rk_t \bmod q$ and verify if it is the discrete logarithm of \overline{pk}_t with respect to g . Note that the encrypted signing key rk_t is returned as a witness w' , and therefore it must be wrapped into a pair such as $w' = (rk_t, 0)$ (step (6)). The MUD sets its interim pseudonym ψ' and respective co-witness (ω', u') by using the blinder ω_1 chosen in step (1). The MUD sets the output credential χ' to 1, which will no longer be used in the following.

After the protocol has successfully terminated, the observer writes the private key rk_t into its local list of private signing keys. If the observer has a different private key on file for the announced type t , then it overwrites the previous private key by the new one.

Show Credential For New Target Pseudonym

If a MUD wants to show a type t credential received earlier, then D looks up the pseudonym ψ_O that it has used when the credential was issued and tells its observer to use ψ_O as well and which type t of credential to show. The observer looks up from its own local lists the witness w it has used when the intro pseudonym ψ_O was introduced and the private signing key rk_t of type t . (The witness w is uniquely determined by ψ_O and is not included in the list of inputs of the observer.)

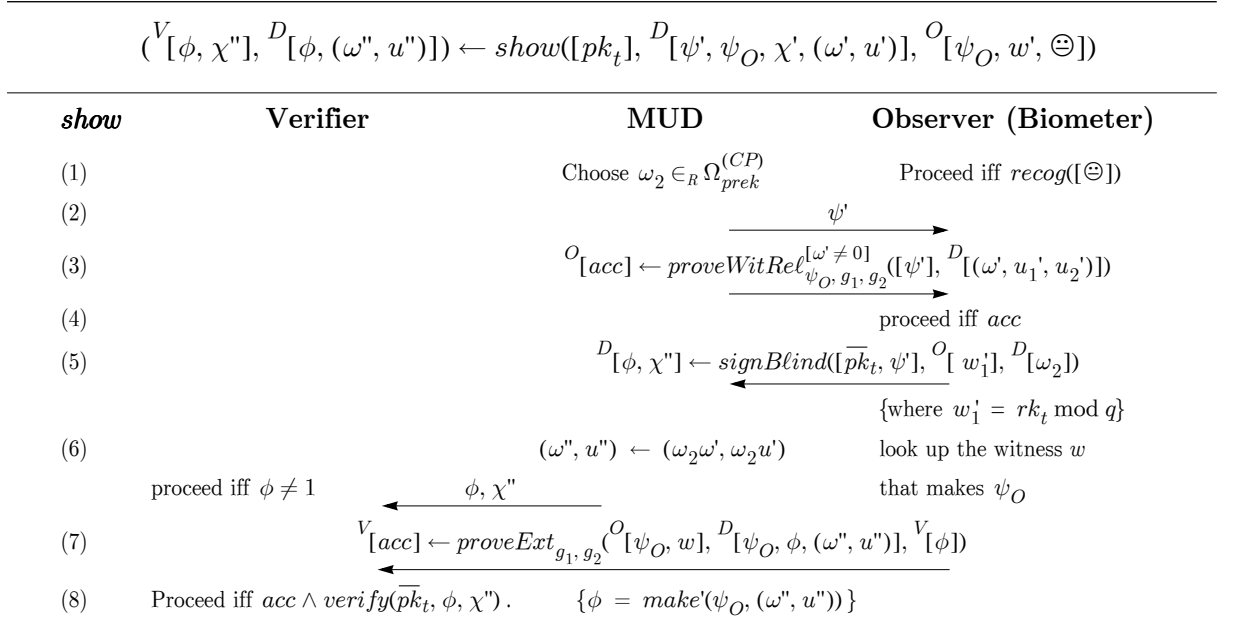


FIGURE 5–11 Showing a Personal Credential for a new Target Pseudonym

The MUD chooses a blinder $\omega_2 \in_R \Omega_{prek}^{(CP)}$ uniformly at random and the observer O proceeds only if it recognizes the actual biometric identity \ominus in step (1). The MUD sends its intermediate pseudonym $\psi' = make'(\psi_O, (\omega', u'))$ to the observer in step (2) and convinces the observer in step (3) that the MUD knows a representation, namely (ω', u') , of ψ' such that $\omega' \neq 0$ by using the protocol $proveWitRel$ of the Brands(2) Mechanism 4.5 on p.46, with respect to the 3 generators ψ_O, g_1, g_2 . In step (4) the observer aborts if it is not convinced by the MUD in step (3). Otherwise, the observer signs ψ' in step (5) using $signBlind$ of the Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55 such that the MUD ends up with a blind signature χ'' for the target pseudonym $\phi = \psi'^{\omega} \bmod p$. Next,

the MUD computes the co-witness (ω'', u'') that co-makes ϕ with respect to ψ_O in step (6) and sends the target pseudonym ϕ and the signature χ'' to the verifier. Preparing the next step, the observer looks up the witness w that makes pseudonym ψ_O . In step (7), the MUD and the observer convince the issuer by an extended proof of knowledge that they together know a witness for the target pseudonym ψ . Finally, the verifier checks the signature χ'' for ϕ with respect to \overline{pk}_t in step (8).

Show Credential for Re-used Target Pseudonym

If a MUD wants to show a credential for a target pseudonym ϕ that has been introduced before, then D looks up the pseudonym ψ_O that it has used to introduce ϕ and to get the credential issued in the first place, and tells its observer to use ψ_O as well.

$$(V[\phi, \chi''], D[\phi]) \leftarrow show^*([pk_t], D[\psi', \psi_O, \chi', v', v''], O[\psi_O, w', \{\ominus\}])$$

<i>show</i>	Verifier	MUD	Observer (Biometer)
(1)		$\omega_2 \leftarrow \frac{\omega''}{w'}$	Proceed iff $recog(\{\ominus\})$
(2)	:	:	:
(3)	$\xrightarrow{\psi'}$ {all further steps of all participants as in protocol <i>show</i> }		

FIGURE 5-12 Showing a Personal Credential for a re-used Target Pseudonym

The protocol *show** differs from protocol *show* only in step (1) of the MUD:

- In step (1) of *show**, the MUD computes the blinder ω_2 from the given co-witnesses v' and v'' rather than choosing ω_2 at random as in step (1) of *show*.

Consequently, step (6) of *show* can be omitted because the co-witness $v'' = (\omega'', u'')$ is already given as an input and is related to co-witness v' and ω_2 as after step (6) of *show*.

Recognize

Before an observer is personalized, it does not recognize any biometric identity, i.e., $recog(\ominus)$ returns *FALSE*. After an observer has been personalized (*persObs*) to a biometric identity \ominus , the observer recognizes this biometric identity without error, i.e., $(FAR, FRR) = (0, 0)$. In other words., whenever $recog(\ominus)$ is called it returns TRUE if and only if the actual biometric identity \ominus sensed and the biometric template \ominus in its memory originate from the same human individual. ♦

Remarks: (1) Note that after running protocol *issue*, the observer ends up with a signing key component rk_t such that $w' = rk_t \bmod q$. Neither the MUD nor its observer have a need to re-use the resulting credential χ' later on. Thus the output parameter χ' is set to 1 just to be defined. (In contrast, the coin credential scheme of Section 5.5 on p.110 will make use of the resulting credential χ' , but not of the resulting output w' of the observer.) Observe that the effectiveness requirement (2) equation (5.2) on p.80 of Definition 5.1 on p.76 does not care about the credential χ' that the MUD outputs after protocol *show*. It only requires that the intermediate pseudonym satisfies $\psi' = make'(\psi_O, (w', u'))$. This intermediate pseudonym and its partial witnesses (ω', u') and w can be taken over from a past execution of *issue*, or they can be chosen from scratch. This will give us

enough flexibility to combine the personal and coin credentials via equal pseudonyms. See Section 5.7 on p.121 and later Section 8 on p.145.

(2) This construction essentially solves an open problem posed by Chen [76,74]. She has asked for a personal certificate scheme based upon the SDL-Assumption (Assumption 3.21 on p.30) or the SR-Assumption (Assumption 3.21 on p.30).

(3) Issuing a credential in the PC Mechanism is essentially to encrypt a secret pre-image of a publicly known one-way function. Hence, the MUD can make sure that the issuer sends nothing else but the uniquely determined pre-image. This is similar—but not identical—to another cryptographic primitive called *publicly verifiable encryption* introduced by Asokan, Shoup and Waidner [5] Section 4. In publicly verifiable encryption, there is an operation V by which anyone can verify whether the encrypted message satisfies a given publicly known predicate. In *issue*, only the MUD hosting the receiving observer can verify whether an encrypted message satisfies the given predicate because only this MUD obtains the additional message d' from the receiving observer in step (4) of *issue*. Since this primitive blends properties of publicly verifiable encryption and proxy encryption according to Blaze, Bleumer, and Strauss [20], it could be named *verifiable proxy encryption*. We will not single out a formal definition of verifiable proxy encryption in this work.

(4) Interesting variations of the PC Mechanism are obtained if the observer's biometric sensor is replaced or accompanied by other types of sensors. For example, locally restricted credentials can be obtained if the observer has a GPS receiver and compares the location where a credential is shown with the location where it was issued. Home-bound or out-bound credentials are achieved if the observer enforces a certain upper bound or lower bound on the distance between both locations. Other modifications or extensions are possible.

Security Suggestion 5.1 PC Mechanism

In the random oracle model, the PC Mechanism is a personal credential scheme:

- (i) Type unforgeability holds under the SR Assumption 3.21 on p.30.
- (ii) Transfer prevention holds under the SR Assumption 3.21 on p.30 and some reasonable assumptions about the El-Gamal encryption scheme and the double discrete log setting.
- (iii) Show-wise fail-safe unlinkability holds under some reasonable heuristic assumptions. ◆

Security Considerations

EFFECTIVENESS

We check the conditions of Definition 5.1 on p.76 (Effectiveness) in turn. Let $prek \in [genPrekey(\mathbb{N})]$ be a correctly generated prekey. Let the observer O be personalized for the holder \ominus , and (rk_O, pk_O) be the native key pair of O . Let (rk_t, pk_t) ($t \in T$) be an issuer's key pair of credential type t .

- (1) Let the issuer, MUD and observer run protocol

$$(I[\psi], D[\psi, \psi_O, (\omega, u)], O[\psi_O, w]) \leftarrow intro([pk_O], I[], D[], O[rk_O, \ominus]) .$$

It is obvious from step (2) in Figure 5–9 on p.94 that the observer's witness w makes the pseudonym $\psi_O = makee(w)$. Furthermore, the source pseudonym returned by protocol *signBlind* in

step (5) in Figure 5–9 on p.94 is $\psi = (\psi_D \psi_O)^\omega \pmod p$ according to step (3) of Figure 4–5 on p.56. Since $\psi_D = \text{make}(u/\omega)$ according to step (2) in Figure 5–9 on p.94, we get the equation

$$\psi = (\text{make}(u/\omega)\psi_O)^\omega = \text{make}(u/\omega)^\omega \psi_O^\omega = \text{make}(u)\psi_O^\omega = \text{make}'(\psi_O, (\omega, u)) \quad . \quad (5.12)$$

For the first rewriting we have used the definition of ψ according to step (2) and step (4) of Figure 5–9 on p.94. The third rewriting uses the homomorphic property of *make* (Lemma 4.1 on p.44), and the last rewriting uses the definition of the co-making function *make'*. Thus, the MUD's co-witness makes the source pseudonym ψ with respect to the intro pseudonym ψ_O . Note that the co-witness (ω, u) chosen in step (1) of *intro* (Figure 5–9 on p.94) is an element of $W_{\text{prek}} = (\mathbb{Z}_q^*)^2$.

(2) Let the above issuer, the MUD and the observer run protocol

$$({}^D[\psi', \chi', (\omega', u')], {}^O[w']) \leftarrow \text{issue}([pk_t], I[rk_t, \psi], {}^D[\psi, \psi_O, (\omega, u)], {}^O[\psi_O, w]) \quad ,$$

where each participant uses its output from *intro* as input.

Firstly, the intermediate pseudonym ψ' has a respective co-witness (ω', u') such that

$$\frac{\omega'}{\omega} = \frac{u'}{u} \pmod q \quad \text{and} \quad (\omega', u') \in W_{\text{prek}} \quad (5.13)$$

because the co-witness (ω', u') is computed to be $(\omega_1 \omega, \omega_1 u)$ for some $\omega_1 \in \mathbb{Z}_q^*$ in step (6) of protocol *issue* (Figure 5–10 on p.95).

Secondly, the co-witness (ω', u') co-makes the intermediate pseudonym ψ' with respect to ψ_O , which can be seen as follows:

$$\psi' = \psi^{\omega_1} = (\text{make}'(\psi_O, (\omega, u)))^{\omega_1} = \text{make}'(\psi_O, (\omega_1 \omega, \omega_1 u)) = \text{make}'(\psi_O, (\omega', u')) \quad . \quad (5.14)$$

For the first and last rewriting we have used the definition of ψ' and (ω', u') according to step (6) of Figure 5–10 on p.95. The second rewriting follows from the pre-condition Figure 5–11 on p.96, and the third rewriting uses Lemma 4.2 on p.48.

Furthermore, the observer ends up with the signing key rk_t of type t :

$$\begin{aligned} & c_1^{-w_1} c_2^{-w_2} c_3' \\ = & c_1^{-\alpha w_1} g_1^{-\beta w_1} c_2^{-\alpha w_2} g_2^{-\beta w_2} c_1^{-\alpha u_1} c_2^{-\alpha u_2} c_3' \psi_O^\beta && \text{step (3), step (4) of } \textit{issue} \\ = & g_1^{-\alpha \omega w_1} g_1^{-\beta w_1} g_2^{-\alpha \omega w_2} g_2^{-\beta w_2} g_1^{-\alpha u_1} g_2^{-\alpha u_2} rk_t^\alpha \psi_O^\alpha g_1^{\beta w_1} g_2^{\beta w_2} && \text{step (2) of } \textit{issue}, \text{ step (2) of } \textit{intro} \\ = & \text{make}(u + \omega w)^{-\alpha} rk_t^\alpha \psi^\alpha && \text{simplify and apply def. of } \textit{make}, \\ = & \text{make}'(\psi_O, (\omega, u)) rk_t^\alpha \psi^\alpha = rk_t \pmod p \quad . && \text{def. of } \textit{make}', \text{ pre-condition } \psi = \text{make}'(\psi_O, (\omega, u)). \end{aligned}$$

5 CREDENTIAL SCHEMES

Hence, it follows that $w' = c_1^{-w_1} c_2^{-w_2} c_3' \bmod p = rk_t \bmod p \pmod{q}$ according to algorithm *genKey* (Figure 5–7 on p.93). Next, the verifying step (5) of *issue* is checked by inserting for d' the value computed and transferred by the observer in step (4) of *issue*:

$$d'c_2' = h^{c_1^{-w_1} c_2^{-w_2} c_3' \bmod p} = h^{rk_t} = \underline{pk}_t \pmod{P} .$$

(3) Let the above MUD and its observer run protocol *show* with a verifier V as follows:

$$(V[\phi, \chi''], D[\phi, (\omega'', u'')]) \leftarrow show([\underline{pk}_t], D[\psi', \psi_O, \chi', (\omega', u')], O[\psi_O, w', \ominus]) .$$

Firstly, the target pseudonym ϕ has a respective co-witness (ω'', u'') such that

$$\frac{\omega''}{\omega'} = \frac{u''}{u'} \bmod q \quad \text{and} \quad (\omega'', u'') \in W_{prek} \quad (5.15)$$

because the co-witness (ω'', u'') is computed to be $(\omega_2 \omega', \omega_2 u')$ for some $\omega_2 \in \mathbb{Z}_q^*$ in step (5) of protocol *show* (Figure 5–11 on p.96). From (5.13) on p.99 in paragraph (2) on p.99 and (5.15) on p.100 in paragraph (3) on p.100 above follows by induction over repeated executions of *issues* and *show* that whenever the MUD introduces a pseudonym ϕ implicitly by using protocol *show*, where the MUD takes shared input ψ_O , the respective co-witness (ω'', u'') satisfies:

$$\frac{\omega''}{\omega} = \frac{u''}{u} \bmod q .$$

Secondly, the co-witness (ω'', u'') co-makes the target pseudonym with respect to pseudonym ψ_O :

$$\phi' = \psi'^{\omega_2} = (make'(\psi_O, (\omega', u')))^{\omega_2} = make'(\psi_O, (\omega_2 \omega', \omega_2 u')) = make'(\psi_O, (\omega'', u'')) .$$

For the first and last rewriting we have used the definition of ϕ and (ω'', u'') according to step (5) of Figure 5–11 on p.96. The second rewriting follows from the precondition (5.14) on p.99, and the third rewriting follows from Lemma 4.2 on p.48. This also ascertains that the extended proof of knowledge in step (7) of Figure 5–11 on p.96 convinces the verifier V .

Thirdly, the observer uses the type t issuing key $w' = rk_t$ it has received according to paragraph (2) on p.99 above in order to produce the resulting credential χ'' . It is a valid blind signature of the Chaum-Pedersen(3) Signature Mechanism for the target pseudonym ϕ with respect to \overline{pk}_t (Security Suggestion 4.6 on p.57) according to step (5) of Figure 5–11 on p.96. Hence, the verifier's check in step (8) of Figure 5–11 on p.96 holds.

- (4) The arguments from (3) on p.100 above apply to protocol *show** in place of protocol *show* because the target pseudonym ϕ is defined in the same way and so is the co-witness $(\omega'', u'') = (\omega_2 \omega', \omega_2 u')$. Since in step (1) of *show**, ω_2 is computed to be $\omega'' / \omega' \bmod q$ in (Figure 5–12 on p.97), it follows from (5.15) on p.100 in paragraph (3) on p.100 above that $\omega_2 = u'' / u' \bmod q$ and thus $(\omega'', u'') = (\omega_2 \omega', \omega_2 u')$.
- (5) By definition of the algorithm *recog*, observers recognize no biometric identities before they are personalized to a biometric template. Afterwards they recognize biometric identities without error.

TYPE UNFORGEABILITY

Consider an attacker coalition of polynomial-time Turing machines, i.e., a MUD \tilde{D} , $L \in \mathbb{N}$ observers $\tilde{O}_1, \dots, \tilde{O}_L$ under the broken observer assumption and a verifier \tilde{V} , that has never been issued a type t credential, but may have verified a number of type t credentials in a gathering stage before the actual attack. In order to show a type t credential, the attacker coalition must provide the following to an honest verifier:

- (i) a target pseudonym ϕ , and a Chaum-Pedersen(3) signature χ valid for ϕ with respect to \overline{pk}_t (after step (6) of *show* in Figure 5–11 on p.96 or *show** in Figure 5–12 on p.97) and
- (ii) an interactive proof-of-knowledge of a witness of ϕ (in step (7) of *show* in Figure 5–11 on p.96 or *show** in Figure 5–12 on p.97).

Since the attacker coalition is presumed to have not been issued a type t credential in the past and to have no access to an honest observer that has been used to get a type t credential issued, the attacker coalition has neither access to the private issuing key $rk_t \bmod q$ itself nor to an honest observer that keeps $rk_t \bmod q$ inside.

Assume the attacker coalition achieves (i) on p.101, then the target pseudonym ϕ must be one of those that the attacker \tilde{V} has received during the gathering stage. Were ϕ not one of those target pseudonyms, then the attacker coalition had come up with a Chaum-Pedersen(3) signature χ valid for a new message ϕ with respect to \overline{pk}_t although the attacker coalition neither had access to the respective signing key $rk_t \bmod q$ itself nor to a signer who had access to $rk_t \bmod q$. This is a contradiction to the restrictiveness of the Chaum-Pedersen(3) signature scheme (Security Suggestion 4.6 on p.57), where we need to look at the case of the recipient obtaining signatures for $n = 0$ messages before he must come up with a message and valid signature of its own (see Definition 4.7 on p.53).

Assume next that the attacker coalition achieves (ii) on p.101. Here in step (7) of protocol *show* or *show**, the attacker coalition proves knowledge of a witness of the target pseudonym ϕ to an honest verifier by using protocol *proveExt* of the *ECEG(2)* Mechanism 4.6 on p.47. Because all observers $\tilde{O}_1, \dots, \tilde{O}_L$ are assumed to be broken and controlled by \tilde{D} , we may focus only on the subprotocol of *proveExt* that is visible to the honest verifier, i.e., the subprotocol between the verifier and the co-prover. Each subprotocol instance of *proveExt* is equivalent to the respective protocol instance of *prove* of the *CEG(2)* Mechanism 4.4 on p.43 according to the extendedness property (Security Suggestion 4.4 on p.48) of *ECEG(2)*. The validity property of this protocol *prove* of *CEG(2)* (Proposition 4.1 on p.44) implies that the attacker coalition “knew” a witness of the target pseudonym ϕ under the one-way function *make* before the attacker coalition entered into step (7). The only time in the past where the attacker coalition may have learned information about a witness of ϕ was during the gathering stage, where an honest MUD showed a credential of type t for the target pseudonym ϕ . More particularly, in step (7) of an execution of protocol *show* or *show**, where \tilde{V} acted as the verifier. That in turn is a contradiction to the witness indistinguishability of protocol *prove* of *CEG(2)* according to (Proposition 4.2 on p.45).

TRANSFER PREVENTION

Consider any polynomial-time attacking MUD \tilde{D} and any number $L \in \mathbb{N}$ of observers O_1, O_2, \dots, O_L under the intact observer assumption. The observers have been personalized to

respective biometric identities $\odot_1, \odot_2, \dots, \odot_L$, and each of these and only these observers have been used by some MUDs including \tilde{D} to acquire a credential of type t . Afterwards, the attacking MUD \tilde{D} is given access to all the intact observers O_1, O_2, \dots, O_L , but not to their respective biometric identities $\odot_1, \odot_2, \dots, \odot_L$. In order to show a type t credential, \tilde{D} must provide to an honest verifier:

- (i) a target pseudonym ϕ , and a Chaum-Pedersen(3) signature χ valid for ϕ with respect to \overline{pk}_t (after step (6) of *show* in Figure 5–11 on p.96 or *show** in Figure 5–12 on p.97) and
- (ii) an interactive proof-of-knowledge of a witness of ϕ (in step (7) of *show* in Figure 5–11 on p.96 or *show** in Figure 5–12 on p.97).

In doing so, \tilde{D} may use

- (1) one or more of the intact observers O_1, O_2, \dots, O_L , or
- (2) one or more other observers that have not been used in order get a credential of type t issued, or
- (3) no observer at all.

Using one or more of the intact observers O_1, O_2, \dots, O_L according to approach (1) on p.102 would not help \tilde{D} because each of them would abort the protocol *show* in Figure 5–11 on p.96 or *show** in Figure 5–12 on p.97 already in step (1) because each of them has a perfect recognition characteristic and \tilde{D} has no access to their respective biometric identities.

Approach (2) on p.102 would not gain \tilde{D} any more information about credentials of type t than approach (3) on p.102 because none of the observers in approach (2) on p.102 has been involved in getting credentials of type t issued. These other observers might store issuing keys of types other than t , but since these issuing keys have been chosen independently of rk_t , they release no information about rk_t . Hence, the probability that \tilde{D} learns any information about rk_t is no better than pure guessing and therefore negligible.

Observe that \tilde{D} without having the proper biometric identities available, cannot coerce the intact observers to execute any other protocol than *issue* because it is the only protocol where the observer does not ask for its correct biometric identity in one of the first steps before sending any information to \tilde{D} . Thus the strongest attacking scenario for \tilde{D} is (A) to use one or more of the intact observers O_1, O_2, \dots, O_L in order to get type t credentials issued, (B) to wait until one or more of the authorized persons use \tilde{D} with the respective intact observer(s) in order to show credentials of type t and then (C) to use the information learned during executions of (A) and (B) in order (C) to show a type t -credential to an honest verifier without using any observer (approach (3) on p.102). We denote this final successful execution of *show* or *show** as *show*[!].

(A) When an issuer issues a type t credential, the issuer encrypts a signing key $rk_t \bmod q$ of type t in step (2) of *issue* using the extended ElGamal encryption explained in equation (5.10) on p.91. Note that \tilde{D} can attack this extended ElGamal encryption only passively, because \tilde{D} can only select plaintexts from a collection of different unknown plaintexts, e.g., $rk_t \bmod q$, by requesting credentials of various types $t \in T$. However, \tilde{D} would not know the selected plaintext, let alone be able to compose a plaintext on its own. Extracting any information about the plaintext from an extended ElGamal encrypted ciphertext by a passive attack is considered to be infeasible under some reasonable assumptions. (See Tsionis and Yung [224] Theorem 1.) However, \tilde{D} learns more during *issue* than just an

ElGamal encrypted plaintext $rk_t \bmod q$ because the observer in step (4) of *issue* is willing to evaluate the function

$$d_{w_1, w_2}(c'_1, c'_2) = h^{c'_1{}^{w_1} c'_2{}^{w_2} \bmod p} \bmod P, \quad \text{for arguments } (c'_1, c'_2) \in G_q^2 \text{ chosen by } \tilde{D}, \quad (5.16)$$

where the primes p , P , and the generator h of G_p are publicly known parts of the prekey, and (w_1, w_2) is the secret witness of the observer involved in the issuing. Learning the exponent $c'_1{}^{w_1} c'_2{}^{w_2} \bmod p$ would be sufficient for \tilde{D} to efficiently compute the private signing key $rk_t = (c'_1{}^{w_1} c'_2{}^{w_2} c'_3 \bmod p) \bmod q$ because c'_3 is also known to \tilde{D} .

Although we cannot reduce this problem to a known complexity theoretic assumption, we have not found a way in which the attacker could infer substantial information about $rk_t \bmod q$ by requesting the observer to evaluate the function $d_{w_1, w_2}(c'_1, c'_2)$ (equation (5.16) on p.103) for chosen arguments (c'_1, c'_2) polynomially many times. It appears neither possible for \tilde{D} to learn information about the observer's witness (w_1, w_2) itself, nor to combine two or more responses by the observer into a new response for a known argument because the function $d_{w_1, w_2}(c'_1, c'_2)$ has no obvious homomorphic properties. For example, the following equation holds for no obvious operation \circ that is independent of (w_1, w_2) and computable in polynomial time:

$$d_{w_1, w_2}(\alpha_1, \alpha_2) d_{w_1, w_2}(\beta_1, \beta_2) = d_{w_1, w_2}(\alpha_1 \circ \beta_1, \alpha_2 \circ \beta_2) .$$

(B) In step (5) of protocol *show* or *show**, the observer uses the signing key $rk_t \bmod q$ only in order to provide a Chaum-Pedersen(3) signature to \tilde{D} . Since the Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55 is restrictive (Security Suggestion 4.6 on p.57), the recipient \tilde{D} (in step (5) of protocol *show* or *show**) has only a negligible chance of figuring out the signing key $rk_t \bmod q$. See the remark following Definition 4.7 on p.53.

It appears plausible that the overall probability that \tilde{D} figures out the issuing key $rk_t \bmod q$ throughout a polynomial composition of steps (A) and (B) is bounded above by the sum of the probabilities of figuring it out in every single step (A) or (B). Since the probability in each single step is negligible and the number of steps is bounded above by a polynomial in k , the overall probability that \tilde{D} figures out the issuing key is still negligible.

(C) We are left to consider an attacker \tilde{D} that executes *show*[!] without an observer and without knowing an issuing key $rk_t \bmod q$ but succeeds to convince the honest verifier by providing:

- (i) a target pseudonym ϕ , and a Chaum-Pedersen(3) signature χ'' valid for ϕ with respect to \overline{pk}_t (after step (6) of *show*[!] in Figure 5–11 on p.96) and
- (ii) an interactive proof-of-knowledge of a witness of ϕ (in step (7) of *show*[!] in Figure 5–11 on p.96).

Assume that \tilde{D} achieves (i) on p.102, then the target pseudonym ϕ must be one of those that \tilde{D} has sent to the honest verifier during an execution of step (A) above. Were ϕ not one of those target pseudonyms then \tilde{D} had come up with a Chaum-Pedersen(3) signature χ'' valid for at least one more message ϕ with respect to \overline{pk}_t than the respective intact observer had provided itself. This would violate the one-timeness of the Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55 (Security Suggestion 4.6 on p.57).

Assume next that \tilde{D} achieves (ii) on p.102. Here in step (7) of *show*[!], \tilde{D} proves knowledge of a witness of the target pseudonym ϕ to an honest verifier by using protocol *proveExt* of the *ECEG(2)* Mechanism 4.6 on p.47. Because \tilde{D} works without an observer, we need to focus on the subprotocol of *proveExt* that is visible to the honest verifier, i.e., the subprotocol between the verifier and the co-prover of *proveExt* in Figure 4–4 on p.48. Each subprotocol instance of *proveExt* is equivalent to the respective protocol instance of *prove* of the *CEG(2)* Mechanism 4.4 on p.43 according to the *extend- edness* property (Security Suggestion 4.4 on p.48) of *ECEG(2)*. The *validity* property of protocol *prove* of *CEG(2)* (Proposition 4.1 on p.44) implies that \tilde{D} “knew” a witness of the target pseudonym ϕ under the one-way function *make* before \tilde{D} entered into step (7) of *show*[!]. The only time in the past where \tilde{D} may have learned a witness of ϕ was during executions of step (B) above, where \tilde{D} used an intact observer O to show a credential of type t for the target pseudonym ϕ (Figure 5–11 on p.96).

(C1) Assume the witness of ϕ that \tilde{D} had learned in that step (B) with respect to g_1, g_2 was (y_1, y_2) .

(C2) In that step (B), \tilde{D} proved to the intact observer O in protocol *proveWitRel* of step (3) of protocol *show* or *show** (Figure 5–11 on p.96) that \tilde{D} knows a witness (ω', u_1', u_2') with respect to ψ_O, g_1, g_2 , where $\omega' \neq 0$ (see Proposition 4.3 on p.47).

(C3) Later on in that step (B), \tilde{D} proved to the honest verifier in protocol *proveExt* of step (7) of protocol *show* or *show** (Figure 5–11 on p.96) that \tilde{D} knows a witness (ω'', u_1'', u_2'') such that

$$\phi = \text{make}'(\psi_O, (\omega'', (u_1'', u_2''))) = \psi_O^{\omega''} g_1^{u_1''} g_2^{u_2''} \pmod{p} .$$

(See Security Suggestion 4.4 on p.48 for *ECEG(2)*).

(C4) In that step (B), \tilde{D} provided to the honest verifier in step (5) of protocol *show* or *show** (Figure 5–11 on p.96) a Chaum-Pedersen(3) signature χ'' for ϕ with respect to ψ_O, g_1, g_2 . From the restrictiveness property of Chaum-Pedersen(3) follows that some ω_2 exists that relates the witnesses of (C2) and (C3) as follows $(\omega'', u_1'', u_2'') = (\omega_2 \omega', \omega_2 u_1', \omega_2 u_2')$. Furthermore, $\omega_2 \neq 0$ because otherwise the condition $\phi = 1$ of step (6) of protocol *show* or *show** were violated.

We conclude that \tilde{D} can compute $\omega_2 = \omega'' / \omega' \pmod{q}$ because it can be derived from the witnesses that \tilde{D} has proved to know in (C2) and (C3). Note that $\omega' \neq 0$ according to (C2).

From here, \tilde{D} can compute a witness of ψ_O with respect to g_1, g_2 as follows. From the assumption (C1) and the presumption (C3), \tilde{D} gets two representations of the target pseudonym ϕ :

$$\phi = g_1^{y_1} g_2^{y_2} = \psi_O^{\omega''} g_1^{u_1''} g_2^{u_2''} \pmod{p} .$$

The second equality is rewritten as follows:

$$g_1^{\frac{y_1 - u_1''}{\omega''}} g_2^{\frac{y_2 - u_2''}{\omega''}} = \psi_O \pmod{p} ,$$

where $\omega'' = \omega_2 \omega' \neq 0$ because neither ω_2 nor ω' disappear according to (C4) and (C2), respectively.

That \tilde{D} can compute a witness of ψ_O is a contradiction to the fact that ψ_O has been chosen uniformly at random by the intact observer O during protocol *intro* and that the intact observer O has used its witness (w_1, w_2) of ψ_O only in its communication with \tilde{D} during protocol *proveExt* of *ECEG(2)* in protocols *intro*, *show* and *show**, and in its response to \tilde{D} in step (4) of protocol *issue*. The subprotocol instance of *proveExt* between the intact observer O and \tilde{D} is equivalent to the respective protocol instance of *prove* of the *CEG(2)* Mechanism 4.4 on p.43 according to the *extend- edness* property (Security Suggestion 4.4 on p.48) of *ECEG(2)*. Therefore, protocol *proveExt* releases no information about the witness (w_1, w_2) of O to \tilde{D} because protocol *prove* is witness indistinguishable. Furthermore, we have argued earlier in step (A) that the intact observer's response to \tilde{D} in step (4) of protocol *issue* would only leave \tilde{D} a negligible chance of figuring out the witness (w_1, w_2) .

SHOW-WISE FAIL-SAFE UNLINKABILITY

We show that for all $n \in \mathbb{N}$, the honest MUD in both $Life_n$ and $life_n$ blocks any in-band inflow to the observer and any in-band outflow from the observer. Consider $2n + 3$ participants of protocol $Life_n$ for any $n \in \mathbb{N}$ (see Section 5.3.4 on p.84), namely an attacking issuer \tilde{I} of type t credentials, an honest MUD D , $n + 1$ attacking observers $\tilde{O}_0, \dots, \tilde{O}_n$ and n attacking credential verifiers $\tilde{V}_1, \dots, \tilde{V}_n$. Consider $n + 1$ successful executions E_0, \dots, E_n of $Life_n$, where each execution E_i ($i \in [0, n]$) is run by \tilde{I} , D , \tilde{O}_i and $\tilde{V}_1, \dots, \tilde{V}_n$. Now consider the resulting views⁹ of all attacking participants on the honest MUD D according to Table 5-5 on p.87 of Definition 5.6 on p.87.

The issuer participates in execution E_0 of $Life_n$ during *intro* and *issue*. Thus the issuer's view $View_0^{\tilde{I}}$ contains an output of *signBlind* (step (5) of *intro*), a verifier's view of *proveExt* (step (7) of *intro*) and an issuer's view of *issue* (step (2) of *issue*). This is summarized in the first two rows of the following Table 5-9 on p.105. The views of attackers \tilde{O}_0 and $\tilde{V}_1, \dots, \tilde{V}_n$ are listed in the following rows. The last column lists all internal choices of the MUD in respective member protocols of $Life_n$.

Attacker	Member prot. of $Life_n$	Attacker's view (including shared inputs) contains	Internal Choice of MUD
issuer \tilde{I}	<i>intro</i> (Figure 5-9)	observer's native verifying key $pk_{\tilde{O}}$, output $\psi, \chi_{\tilde{O}}$ of <i>signBlind</i> (step (5)), verifier's view of <i>proveExt</i> (step (7))	(see below under <i>intro</i> for observers)
$(View_0^{\tilde{I}})$	<i>issue</i> (Figure 5-10)	issuer's view in <i>issue</i> (step (2))	(see below under <i>issue</i> for observers)

TABLE 5-9 Attackers and their views on the MUD in $Life_n$

9) Of the views of \tilde{I} , \tilde{O}_i and \tilde{V}_i on D we are only interested in the messages exchanged with D , not in the internal choices of \tilde{I} and \tilde{O} (see Definition 3.11 on p.21). So in order to relax the view notation, we do not display the internal choices of \tilde{I} , \tilde{O}_i and \tilde{V}_i in the remaining argument.

Attacker	Member prot. of $Life_n$	Attacker's view (including shared inputs) contains	Internal Choice of MUD
	<i>intro</i> (Figure 5–9)	$\psi_D, \psi_{\tilde{O}}, \Theta_i$ (step (2) and (3)), signer's view of <i>signBlind</i> (step (5)), prover's view of <i>proveExt</i> (step (7))	$(\omega, u) \in_R V_{prek}$ in <i>signBlind</i> (one execution) in <i>proveExt</i> (one execution)
observer \tilde{O}_0 ($View_0^{\tilde{O}_0}$)	<i>issue</i> (Figure 5–10)	source pseudonym ψ , observer's view in <i>issue</i> (step (3), (4),(5))	$\beta, \omega_1 \in_R \Omega_{prek}^{(CP)}$
	<i>show</i> (Figure 5–11)	intermediate pseudonym ψ' verifier's view of <i>proveWitRel</i> (step (3)), Θ_i , (step (4)), signer's view of <i>signBlind</i> (step (5)), prover's view of <i>proveExt</i> (step (7))	$\omega_{2,1}, \dots, \omega_{2,n} \in_R \Omega_{prek}^{(CP)}$ in <i>proveWitRel</i> in <i>signBlind</i> in <i>proveExt</i>
verifier $\tilde{V}_1, \dots, \tilde{V}_n$ ($View_i^{\tilde{V}_i}$)	<i>show</i> (Figure 5–11)	type t verifying key pk_t , output (ϕ, χ') of <i>signBlind</i> (step (5)), verifier's view of <i>proveExt</i> (step (7))	(see above under <i>show</i> for observers)

 TABLE 5-9 Attackers and their views on the MUD in $Life_n$

“ $\leq q$ ”: We show that any such $(n+2)$ -tuple of valid views $(view_0^{\tilde{I}}, view_0^{\tilde{O}_0}, view_1^{\tilde{V}_1}, \dots, view_n^{\tilde{V}_n})$ originating from $n+1$ executions of $Life_n$ according to Definition 5.6 on p.87 matches with at most q values of the internal choice of the MUD in $Life_n$. D 's internal choice is detailed in the last column of Table 5-9 on p.105.

- (i) The issuer's view in *intro* contains the respective observer's native verifying key $pk_{\tilde{O}_i}$, which uniquely determines a particular native signing key $rk_{\tilde{O}_i}$ of some observer \tilde{O}_i , $i \in [0, n]$.

The issuer's and the observer's views in *intro* together determine the MUD's co-witness (ω, u) up to q possible values as follows: $\omega = \log_{\psi_D} \psi_{\tilde{O}_i}$ (from step (5)) and $u \in make^{-1}(\psi_D^\omega)$ (from step (2)). They also determine a unique internal choice of the MUD in member protocol *signBlind* with signing key $rk_{\tilde{O}_i}$ (step (5)), and, for each possible co-witness (ω, u) , they determine a unique internal choice of the MUD in member protocol *proveExt* (step (7)). This follows from blindness of *signBlind* (Security Suggestion 4.6 on p.57) and from co-unlinkability of *proveExt* (Security Suggestion 4.4 on p.48), respectively.

- (ii) The issuer's and the observer's views in *issue* uniquely determine the choices β, ω_1 of the MUD in step (1) of *issue* as follows: $\beta = \log_{g_1}(c_1' c_1^{-\omega} \bmod p)$ (from step (3)) and $\omega_1 = \log_{\psi} \psi'$ (from step (6)).

- (iii) For each possible co-witness (ω, u) of the MUD, the verifier's view and the observer's view in each instance of *show* uniquely determine the internal choice s_1, s_2, s_3 of the MUD in member protocol *proveWitRel*. Note that for each co-witness $(\omega, (u_1, u_2))$, the given challenge c and the MUD's response messages r_1, \dots, r_3 in *proveWitRel* in Figure 4–3 on p.47 establish 3 linear equations, one for each of the three variables s_1, s_2, s_3 according to step (4) of protocol *proveWitRel*. Thus, for each co-witness (ω, u) of the MUD, there is exactly one internal choice $\vec{s} = (s_1, s_2, s_3)$ of the MUD in step (1) of *proveWitRel* (Figure 4–3 on p.47) such that the MUD produces the given views to the verifier and observer.

Furthermore in member protocol *signBlind* in step (5) of *show*, the verifier's view and the observer's view determine unique blinders $\omega_{2,i} = \log_{\psi'} \phi_i$ for $i = 1, \dots, n$ (which are chosen in

step (1) of *show*), and a unique internal choice of the MUD in *signBlind* (Security Suggestion 4.6 on p.57: blindness). In member protocol *proveExt* in step (7) of *show*, for each possible co-witness (ω, u) of the MUD, the verifier's view and the observer's view determine a unique internal choice of the MUD (Security Suggestion 4.4 on p.48: co-unlinkability).

" $\geq q$ ": We are left to show that each of the q internal choices identified above in fact results in the $(2n + 2)$ -tuple of given views. The only messages of the MUD that have not yet been taken into account are c_2, c_2', c_3, c_3', d' of respective step (3) and step (4) of member protocol *issue* of *Life_n*. We need to consider any two valid views (c_1, c_2, c_3) of an issuer on the MUD and (c_1', c_2', c_3', d') of an observer on the MUD. We first identify the valid views by using two heuristic arguments:

- For the issuer's and the observer's views to be valid, the observer must be able to recover the secret rk_t from the output messages (c_1', c_2', c_3') that the issuer has encrypted into the input messages (c_1, c_2, c_3) . Otherwise, the observer would not be able to help the MUD later on to show a credential of type t , and hence, a protocol *show* would be unsuccessful. In other words, decrypting the output (c_1', c_2', c_3') must yield the same result as decrypting the input (c_1, c_2, c_3) :

$$c_1^{-(\omega u_1 + w_1)} c_2^{-(\omega u_2 + w_2)} c_3 = c_1'^{-w_1} c_2'^{-w_2} c_3' \pmod{p} . \quad (5.17)$$

- Secondly, we assume that the issuer's view c_1, c_2, c_3 must satisfy the condition

$$\log_{g_1} c_1 = \log_{g_2} c_2 \quad (5.18)$$

in order for the observer being able to decrypt the plaintext rk_t . Note that an attacking issuer and an attacking observer could initially agree on any encryption mechanism and any encryption/decryption key pair of which the MUD may have no idea. However, the way in which the MUD transforms the issuer's messages c_1, c_2, c_3 into the observer's messages c_1', c_2', c_3' during protocol *issue* appears as if only the extended ElGamal like encryption mechanism makes sense to use, because otherwise the observer will not be able to decrypt the plaintext from messages c_1', c_2', c_3' . Thus, we also assume that the observer must receive an extended ElGamal ciphertext, and therefore a valid view (c_1', c_2', c_3', d') of the observer satisfies the condition

$$\log_{g_1} c_1' = \log_{g_2} c_2' . \quad (5.19)$$

- Thirdly, it is clear that d' is uniquely determined by c_3' according to step (5) of member protocol *issue* of *Life_n*, namely: $d' = pk^{1/c_3'} \pmod{p}$. Any other d' would not contribute to a valid view of an observer on the MUD.

Thus, the remaining messages c_2' and c_3' must be chosen exactly as prescribed by respective step (3) and step (4) of member protocol *issue*: In order to address message c_2' , we rewrite the expression $c_2^\omega g_2^\beta \pmod{p}$ by expressing it as a power of g_2 :

$$\begin{aligned} c_2^\omega g_2^\beta &= (g_2^\delta)^\omega g_2^\beta && \text{let } \delta = \log_{g_2} c_2 \\ &= g_2^{\omega\delta + \beta} \end{aligned}$$

5 CREDENTIAL SCHEMES

From step (3) of *issue* we have $c_1' = c_1^\omega g_1^\beta \pmod{p}$, which is $g_1^{\omega\delta + \beta}$ because of condition (5.18) on p.107. From condition (5.19) on p.107 we find that the above expression $g_2^{\omega\delta + \beta}$ can be rewritten as

$$= c_2' \pmod{p} . \quad (5.20)$$

In order to address message c_3' , we rewrite the expression $c_1^{-u_1} c_2^{-u_2} c_3 \psi_O^\beta$ for each of the identified pairs $u = (u_1, u_2)$ as follows:

$$\begin{aligned} c_1^{-u_1} c_2^{-u_2} c_3 \psi_O^\beta &= c_1^{-(u_1 + \omega w_1)} c_2^{-(u_2 + \omega w_2)} c_3 \psi_O^\beta c_1^{\omega w_1} c_2^{\omega w_2} && \text{expand expression} \\ &= c_1'^{-w_1} c_2'^{-w_2} c_3 \psi_O^\beta c_1^{\omega w_1} c_2^{\omega w_2} && \text{by condition (5.17) on p.107} \\ &= (c_1^{-\omega w_1} g_1^{-\beta w_1}) (c_2^{-\omega w_2} g_2^{-\beta w_2}) c_3 \psi_O^\beta c_1^{\omega w_1} c_2^{\omega w_2} && \text{insert } c_1', c_2' \text{ from step (3) of } issue \\ &= (c_1^{-\omega w_1} g_1^{-\beta w_1}) (c_2^{-\omega w_2} g_2^{-\beta w_2}) c_3' (g_1^{\beta w_1} g_2^{\beta w_2}) c_1^{\omega w_1} c_2^{\omega w_2} && \psi_O = g_1^{w_1} g_2^{w_2} \pmod{p} \\ &= c_3' \pmod{p} . && (5.21) \end{aligned}$$

We find that each $(2n + 2)$ -tuple of views of an issuer, $n + 1$ observers and n credential verifiers of $Life_n$ allows the MUD exactly q different values for the MUD's internal choice in $Life_n$.

Next, consider $2n + 3$ participants of protocol $life_n$ for any $n \in \mathbb{N}$ (see Section 5.3.4 on p.84), namely an attacking issuer \tilde{I} of type t credentials, an honest MUD D , $n + 1$ attacking observers $\tilde{O}_0, \dots, \tilde{O}_n$ and n attacking credential verifiers $\tilde{V}_1, \dots, \tilde{V}_n$. Consider $n + 1$ successful executions e_0, \dots, e_n of $life_n$, where each execution e_i ($i \in [0, n]$) is run by \tilde{I} , D , \tilde{O}_i and $\tilde{V}_1, \dots, \tilde{V}_n$. Now consider the resulting views⁹ of all attacking participants on the honest MUD D according to Table 5-6 on p.88 of Definition 5.6 on p.87.

The only difference between the protocols $life_n$ and $Life_n$ is that in $life_n$ the MUD introduces the source pseudonym implicitly by having the issuer verify some credential. Thus, the issuer's view and the observer's view in member protocol *intro* of $Life_n$ must be replaced by their respective views in the first instance of member protocol *show* in $life_n$. The resulting views of the attackers on the honest MUD are summarized in Table 5-10 on p.108. The fourth column lists the internal choice of the MUD in each member protocol of $life_n$.

Attacker	member prot. of $life_n$	Attacker's view (including shared inputs) contains	Internal Choice of MUD
issuer \tilde{I} ($View_0^{\tilde{I}}$)	<i>show</i> (Figure 5-11)	type t_0 ^a verifying key pk_{t_0} , output ϕ_0, χ_0 of <i>signBlind</i> (step (5)), verifier's view of <i>proveExt</i> (step (7))	(see below under <i>show</i> for observers)
	<i>issue</i> (Figure 5-10)	issuer's view in <i>issue</i> (step (2))	(see below under <i>issue</i> for observers)

TABLE 5-10 Attackers and their views on the MUD in $life_n$

Attacker	member prot. of $life_n$	Attacker's view (including shared inputs) contains	Internal Choice of MUD
observer O_0 ($View_0^{\tilde{O}_0}$)	<i>show</i> (Figure 5–11)	intermediate pseudonym ψ'_0 (step (2)), verifier's view of <i>proveWitRel</i> (step (3)), \ominus_i (step (4)), signer's view of <i>signBlind</i> (step (5)), prover's view of <i>proveExt</i> (step (7))	$\omega_{2,0} \in_R \Omega_{prek}^{(CP)}$ (one value of ω_2 in <i>proveWitRel</i> (one execution) in <i>signBlind</i> (one execution) in <i>proveExt</i> (one execution)
	<i>issue</i> (Figure 5–10)	source pseudonym $\psi = \phi_0$, ^b observer's view in <i>issue</i> (step (3),(4),(5))	$\beta, \omega_1 \in_R \Omega_{prek}^{(CP)}$
	<i>show</i> (Figure 5–11)	intermediate pseudonym ψ' (step (2)) verifier's view of <i>proveWitRel</i> (step (3)), \ominus_i (step (4)), signer's view of <i>signBlind</i> (step (5)), prover's view of <i>proveExt</i> (step (7))	$\omega_{2,1}, \dots, \omega_{2,n} \in_R \Omega_{prek}^{(CP)}$ (n values of ω_2 in <i>proveWitRel</i> (n executions) in <i>signBlind</i> (n executions) in <i>proveExt</i> (n executions)
verifier $\tilde{V}_1, \dots, \tilde{V}_n$ ($View_i^{\tilde{V}_i}$)	<i>show</i> (Figure 5–11)	type t verifying key pk_t , output ϕ, χ of <i>signBlind</i> (step (5)), verifier's view of <i>proveExt</i> (step (7))	(see above under <i>show</i> for observers)

 TABLE 5-10 Attackers and their views on the MUD in $life_n$

- Since there are multiple instances of protocol *show* in protocol $life_n$, we will use an index zero for the parameters of the first instance of *show*, which precedes protocol *issue*, and a respective index $i = 1, \dots, n$ for the parameters of each instance of *show* following protocol *issue*.
- Since the target pseudonym introduced by the first instance of *show* also serves as a source pseudonym during the following instance of *issue*, the two pseudonyms are equal: $\psi = \phi_0$.

“ ≤ 1 ”: We show that any such $(n+2)$ -tuple of valid views $(view_0^{\tilde{I}}, view_0^{\tilde{O}_0}, view_1^{\tilde{V}_1}, \dots, view_n^{\tilde{V}_n})$ originating from $n+1$ executions of $Life_n$ according to Definition 5.6 on p.87 matches with at most one internal choice of the MUD in $life_n$, which is detailed in the last column of Table 5-10 on p.108.

- The issuer's and the observer's views in *show* together uniquely determine the MUD's internal choice $\vec{s} = (s_1, s_2, s_3)$ in *proveWitRel* (see the argument for protocol $Life_n$ and a unique value $\omega_{2,0} = \log_{\psi_0} \phi_0$). They also determine a unique internal choice of the MUD in member protocol *signBlind* (step (5) of *show*), and they determine a unique internal choice of the MUD in member protocol *proveExt* (step (7)). This follows from blindness of *signBlind* (Security Suggestion 4.6 on p.57) and from co-unlinkability of *proveExt* (Security Suggestion 4.4 on p.48), respectively.
- The issuer's and the observer's views in *issue* uniquely determine the choices β, ω_1 of the MUD in step (1) of *issue* as follows: $\beta = \log_{g_1} (c_1' c_1^{-\omega} \bmod p)$ (from step (3)) and $\omega_1 = \log_{\psi} \psi'$ (from step (6)).
- The verifier's view and the observer's view in each instance of *show* uniquely determine the internal choice of the MUD in member protocol *proveWitRel*. Furthermore in member protocol *signBlind* in step (5) of *show*, the verifier's view and the observer's view determine a unique value $\omega_{2,i} = \log_{\psi} \phi_i$ (from step (5)), and, for each possible co-witness (ω, u) , they determine a unique internal choice of the MUD in member protocol *proveExt* (step (7)). This follows again from blindness of *signBlind* (Security Suggestion 4.6 on p.57) and from co-unlinkability (Security Suggestion 4.4 on p.48), respectively.

“ ≥ 1 ”: We are left to show that the q internal choices identified above in fact result in the $(2n + 2)$ -tuple of given views. The only messages of the MUD that have not yet been taken into account are c_2, c_2', c_3, c_3', d' of protocol *issue*. The uniquely determined internal choice for (β, ω_1) of *life_n* that was identified above makes the MUD produce c_2' and c_3' as well because the rewritings (5.20) on p.108 and (5.21) on p.108 apply to this case as well. \square

5.5 Coin Credential Schemes

First, we give a definition of coin credential schemes as a special case of credential schemes (Section 5.2.2 on p.74) and afterwards present a construction based on an e-cash mechanism by Brands [34]. This construction is compatible with the personal credential mechanism of Mechanism 5.8 on p.92 in the sense that source pseudonyms are taken from the same domain and are represented in the same way.

5.5.1 Definition

Definition 5.9 Coin Credential Scheme

A coin credential scheme with credential types T is a type-unforgeable, K -overshow detecting and/or preventing and show-wise or issue-wise fail-safe unlinkable credential scheme with credential types T and recognition characteristic $(FAR, FRR) = (1, 0)$.¹⁰ The domain of biometric identities and the operation *persObs* are not defined for a coin credential scheme. \blacklozenge

The operations *issue* and *show* of coin schemes are sometimes called “withdraw” and “pay” in the literature. The former is motivated by the fact that it is usually the customers who actively withdraw money from their accounts at a bank, and not the banks who “push” money into a customer’s electronic wallet. The term “pay” refers to the fact that coin credentials are consumed by showing them. Nevertheless, we avoid distractions by renaming operations and keep to the names introduced in Definition 5.1 on p.76.

5.5.2 Coin Credential Mechanism

In the following coin credential Mechanism 5.10 on p.111, mobile user devices are equipped with non-biometric observers, so as to prevent overshooting. An efficient coin credential mechanism is presented where the MUD and observer represent source pseudonyms in the same way as in the PC Mechanism 5.8 on p.92 as follows

$$\psi = g_1^{u_1 + \omega w_1} g_2^{u_2 + \omega w_2},$$

where (p, q) is a discrete log setting (Definition 3.17 on p.28), g_1, g_2 are constant generators of G_q , the MUD holds the co-witness $(\omega, u) = (\omega, (u_1, u_2))$ and its observer holds the witness $w = (w_1, w_2)$. This is a generalization of Brands’ original proposal [34,35] where all co-witnesses have a constant component $\omega = 1$. The following coin credential mechanism enables the MUD to obtain a coin credential

10) A recognition characteristic of $(1,0)$ means that all biometric identities are accepted, or in other words, biometric identities are completely ignored.

- for the same source pseudonym for which the MUD is issued a personal credential of the PC Mechanism 5.8 on p.92, or
- for the same target pseudonym for which the MUD has shown a personal credential before.

The dual case of showing a coin credential for a target pseudonym for which the MUD has shown a personal credential before will be considered in Section 5.7 on p.121.

The main reason for essentially repeating Brands proposal here is to prepare the construction of coin group credentials in Section 7.2 on p.143, which is a major extension to Brands proposal.

Mechanism 5.10 Coin Credential Mechanism (CC)

Let $(genPrekey^{(CP)}, genKey, signBlind, verify)$ be the operations of the Chaum-Pedersen(3) Signature Mechanism and $(genPrekey^{(ECEG(3))}, make, proveExt)$ be the extended proof-of-knowledge scheme ECEG(3), whose families of candidates, witnesses, and making functions equal the families of messages, witnesses and making functions of the Chaum-Pedersen(3) Signature Mechanism. Then, the Coin Credential Mechanism for credential types $T = \{t_1, \dots, t_m\}$ is constructed as follows.

GENERATE PREKEY $(p, q, g, g_1, g_2, g_3, hash) \leftarrow genPrekey(k)$

Pick a discrete log setting (p, q) uniformly at random from dls_k (Definition 3.17 on p.28). Then pick uniformly at random four generators $g, g_1, g_2, g_3 \in_R G_q$ and select a hash function that maps binary strings to elements in \mathbb{Z}_q^* . The generators g, g_1, g_2, g_3 and the hash function are also used for the Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55.

DOMAINS

- The signing key domains are $RK_{prek} = RK_{prek}^{(CP)}$, and the verifying key domains are $PK_{prek} = PK^{(CP)}$, which contain signing and verifying keys of the Chaum-Pedersen(3) Signature Mechanism, respectively (Mechanism 4.8 on p.55).
- The pseudonym domains are $\Psi_{prek} = M_{prek}^{(CP)} \cup (M_{prek}^{(CP)})^2$. Source pseudonyms and intro pseudonyms are taken from $M_{prek}^{(CP)}$, whereas target pseudonyms are taken from $(M_{prek}^{(CP)})^2$.
- The pseudonym witness domains are $W_{prek} = W_{prek}^{(ECEG(3))}$. The subset of witnesses with third component zero, i.e., $w = (w_1, w_2, 0)$, is denoted as $W_{prek}^{(ECEG(2+1))}$. The blinder domains are $\Omega_{prek}^{(CP)} = \mathbb{Z}_q^*$.
- The co-witness domains are $V_{prek} = V_{prek}^{(ECEG(3))} = \Omega_{prek}^{(CP)} \times W_{prek}^{(ECEG(3))}$. The subset of co-witnesses $v = (\omega, u)$ with $u \in W_{prek}^{(ECEG(2+1))}$ is denoted as $V_{prek}^{(ECEG(2+1))}$.
- The making functions are inherited from $CEG(3)$: $make(w) = \prod_{i=1}^3 g_i^{w_i} \bmod p$.
- The co-making functions are from $ECEG(3)$: $make'(\psi, (\omega, u)) = \psi^\omega \prod_{i=1}^3 g_i^{w_i} \bmod p$.
- The credential domains are $C_{prek} = \Sigma_{prek}^{(CP)}$. They are also the signature domains of the Chaum-Pedersen(3) Signature Mechanism.
- The transcript domains are $TR_{prek} = W_{prek}^{(ECEG(3))} \times \Omega_{prek}^{(CP)}$, i.e., the product of respective witness and blinder domains of the Chaum-Pedersen(3) Signature Mechanism.
- The optional domains \mathcal{B}_{prek} of biometric identities are undefined.

OPERATIONS

The operations *genKey* and *verify* are inherited from the Chaum-Pedersen(3) Signature Mechanism with the convention that $genKey(preK, t) = (genKey^{(CP)}(preK), t)$. The remaining four operations (*intro*, *issue*, *show*, and *extract*) are defined below.

Introduce Pseudonym

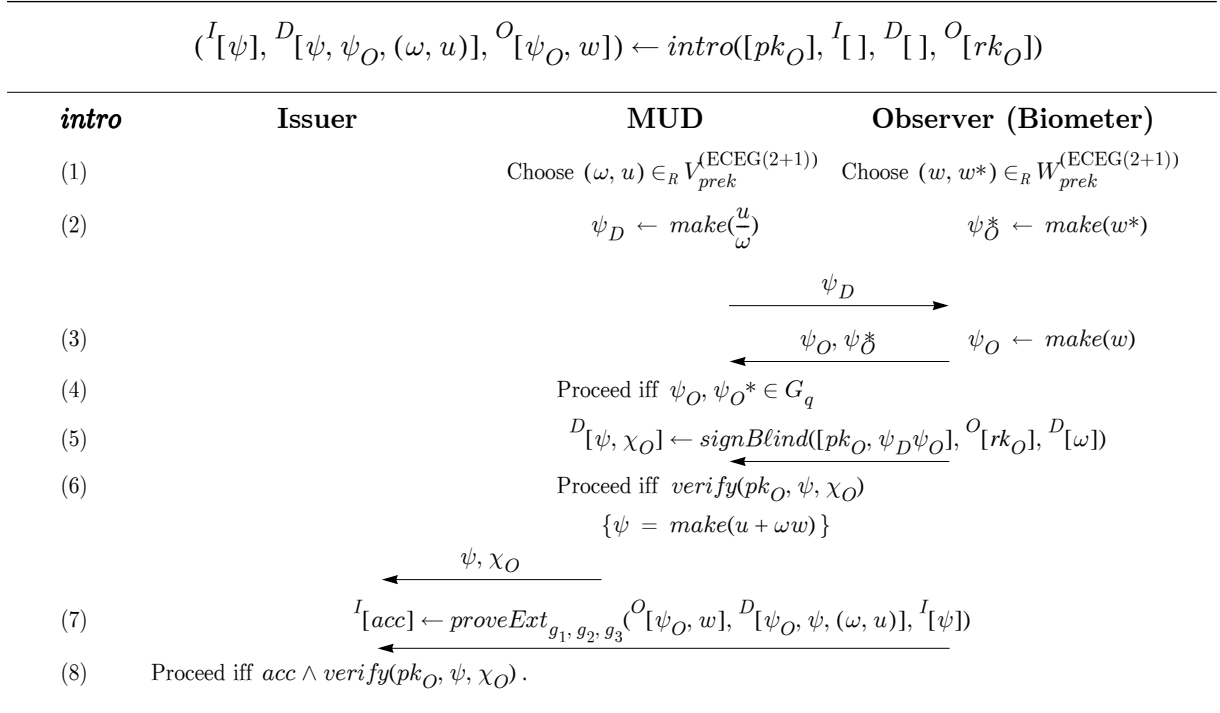


FIGURE 5-13 Introducing a Pseudonym

This protocol is essentially the same as for the PC Mechanism (Figure 5-9 on p.94) except that no biometric verification is performed, and the observer chooses an auxiliary witness $w^* = (w_1, w_2, 0) \in_R W^{(ECEG(2+1))}$ uniformly at random. The auxiliary witness w^* makes the auxiliary pseudonym ψ_O^* (step (2)). After successful execution of the protocol, the MUD writes the auxiliary intro pseudonym ψ_O^* to its composition tape, and the observer writes the corresponding auxiliary witness w^* to its composition tape.

Issue Credential

The MUD chooses three blinders $\omega_1, \omega_2, \omega_3$ uniformly at random (step (1)) and computes the values φ and $\bar{\varphi}$ by taking its observers auxiliary pseudonym into account (step (2)). Afterwards, *D* blinds the factor φ (step (3)) and prepares the intermediate pseudonym component $\underline{\psi}'$ (step (4)). Now, *D* receives from *I* a blind signature for the intermediate pseudonym component $\bar{\psi}'$. *D* also authenticates the intermediate pseudonym component $\underline{\psi}'$ using the extended protocol *signBlind* in step (5) (see Section 4.2.2 on p.55). The MUD accepts if the signature returned is valid for both intermediate pseudonym components $\bar{\psi}'$ and $\underline{\psi}'$ (step (6)). Then *D* defines its intermediate pseudonym ψ' as the

$$({}^D[\psi', \chi', (\omega', u')], {}^O[w']) \leftarrow \text{issue}([pk_t], I[rk_t, \psi], D[\psi, \psi_O, (\omega, u)], O[\psi_O, w])$$

<i>issue</i>	Issuer	MUD	Observer
(1)		Choose $\omega_1, \omega_2, \omega_3 \in_R \Omega_{prek}^{(CP)}$	
(2)		$\varphi \leftarrow \psi \cdot \psi_O^{\omega_2/\omega_1} \cdot g_3^{\omega_3/\omega_1} \bmod p$ $\bar{\varphi} \leftarrow \psi_O^{-\omega_2/\omega_1} \cdot g_3^{1-\omega_3/\omega_1} \bmod p$	
(3)	$\varphi \leftarrow \psi \cdot g_3 \bmod p$	$\varphi \leftarrow \varphi \bar{\varphi} \bmod p$	$\{\varphi = \text{make}(u + \omega w + (0, 0, 1))\}$
(4)		$\underline{\psi}' \leftarrow \varphi^{\omega_1} \bmod p$	
(5)	${}^D[\bar{\psi}', \chi'] \leftarrow \text{signBlind}([pk_t, \varphi], I[rk_t], D[\{\underline{\psi}'\}, \omega_1])$		
(6)		Proceed iff $\text{verify}(pk_t, \bar{\psi}', \{\underline{\psi}'\}, \chi')$	
(7)		$\psi' \leftarrow (\bar{\psi}', \underline{\psi}')$	
(8)		$(\omega', u') \leftarrow (\omega_1 \omega, \omega_1(u + (0, 0, 1)))$ $\{\bar{\psi}' = \text{make}(u' + \omega' w')\}$	$w' \leftarrow w$

FIGURE 5-14 Issuing a Coin Credential

pair $(\bar{\psi}', \underline{\psi}')$. Finally, D and O update their respective co-witnesses and witnesses, and D writes the three blinders $\omega_1, \omega_2, \omega_3$ to its composition tape (step (8)).

Show Credential

D passes the credential χ' for the pseudonym ψ' to the verifier in step (1), and V verifies it using the extended verification of Figure 4-5 on p.56 in step (2). After successful verification, V chooses a challenge c at random (step (3)) and sends it to D . D looks up the values ω_1, ω_2 from its composition tape and transforms c into another challenge c' according to step (4). Then D forwards the challenge c' to O . O makes sure that this is the first time it helps its host D to show a coin credential by using witness w' . Then O responds by looking up the auxiliary witness w^* from its composition tape and evaluating two linear polynomials $w_1 + Xw_1^*$ and $w_2 + Xw_2^*$ at $X = c'$ according to step (5). Note that the third components are zero, i.e., $w_3 = w_3^* = 0$. D returns the resulting triple t' back to D . D looks up the auxiliary intro pseudonym ψ_O^* from its composition tape and checks, if the response t' is correct (step (6)). Then D evaluates two linear polynomials t_1 and t_2 at t_1' and t_2' and adds a proper control value t_3 , which depends on the yet unused value ω_3 on D 's composition tape (step (7)). Next, D returns the result t back to V . If V can successfully verify t (step (8)), it sets the target pseudonym ϕ and credential χ'' and returns the transcript τ consisting of the challenge c and D 's response t (step (9)).

Extract

On input two different transcripts for the same pseudonym ϕ , *extract* returns a witness w for ϕ . ♦

Showing a coin credential according to Figure 5-15 on p.114 differs from showing a personal credential according to Figure 5-11 on p.96 in a number of ways. However, both protocols have a similar struc-

$$(\overset{V}{\llbracket} \phi, \chi'', \tau \rrbracket, \overset{D}{\llbracket} \phi, (\omega'', u'') \rrbracket) \leftarrow \text{show}(\llbracket pk_t \rrbracket, \overset{D}{\llbracket} \psi', \psi_O, \chi', (\omega', u') \rrbracket, \overset{O}{\llbracket} \psi_O, w' \rrbracket)$$

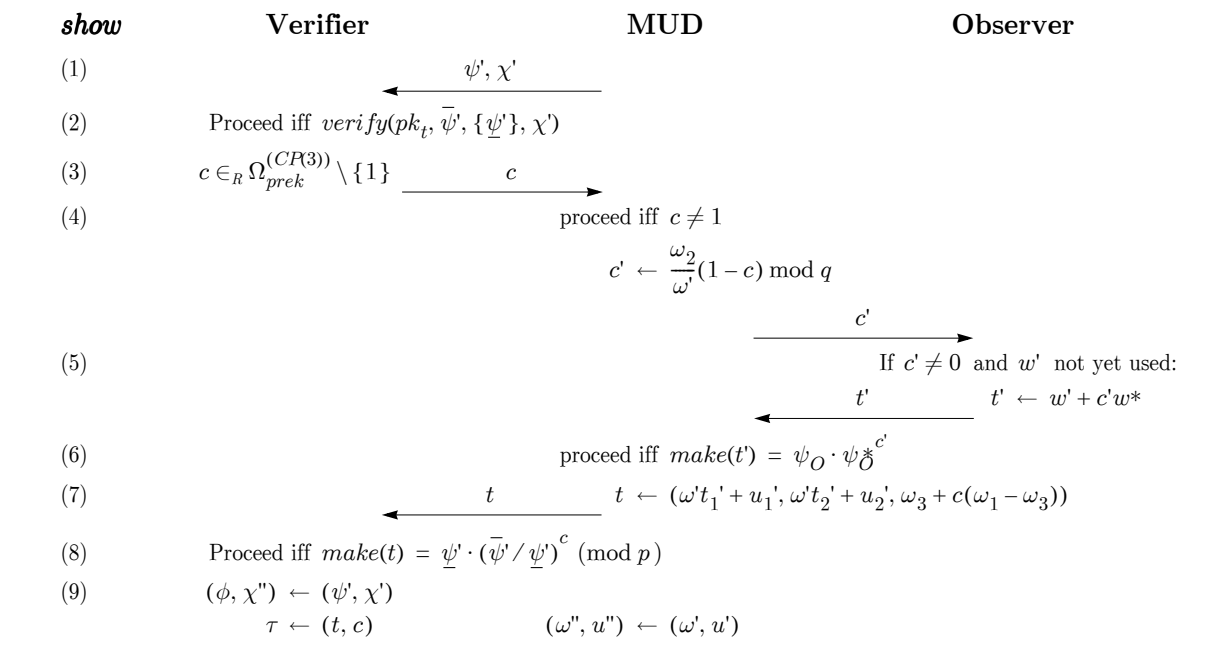


FIGURE 5-15 Showing a Coin Credential

$$w \leftarrow \text{extract}(\{\tau_1, \tau_2, \dots\})$$

extract

- (1) Let $\sigma = (s_1, s_2, s_3, b)$ and $\tau = (t_1, t_2, t_3, c)$ be two different transcripts in the argument set $\{\tau_1, \tau_2, \dots\}$ that have originated from two shows for the same target pseudonym. If the argument contains no two such transcript, then return \perp and stop. Otherwise,
- (2) return $w = (w_1, w_2) = \left(\frac{s_1(c-1) - t_1(b-1)}{s_3(c-1) - t_3(b-1)}, \frac{s_2(c-1) - t_2(b-1)}{s_3(c-1) - t_3(b-1)} \right)$.

FIGURE 5-16 Extracting a Witness

ture. In order to prevent replay, they both provide a proof of knowledge to the verifier that the MUD and its observer know a witness of the target pseudonym (or one its components). In protocol *show* of the PC Mechanism 5.8 on p.92, the MUD and its observer use an extended proof of knowledge according to the *ECEG(2)* Mechanism, while in protocol *show* of the CC Mechanism 5.10 on p.111 the proof of knowledge is integrated into the interactive protocol that guarantees the double spender detection.

Remarks: It is worth noting that the CC Mechanism 5.10 on p.111 is secure against holder framing by computationally unlimited verifiers. If a verifier claims that a coin credential has been spent twice, although in fact it is spent for the first time, a court would ask the verifier to submit the witness he has found for the holder's source pseudonym. A computationally unlimited attacker can come up with a witness for any given source pseudonym, because he can compute representations of pseudonyms.

However, since there are $q^{\ell-1}$ possible witnesses for each pseudonym (ℓ the number of generators), even a computationally unlimited attacker has only but negligible chance of submitting exactly the holder's witness. So the court can detect this kind of attacker by comparing the true witness of the accused holder with the alleged witness submitted by the attacker (see [34] p.43).

A more efficient approach is obtained by following a remark of Brands [34] Section 11: He has conjectured that his original offline e-cash scheme also works with simpler witnesses from \mathbb{Z}_q^2 such that source pseudonyms have witnesses $(w_1, 0)$ and intermediate and target pseudonyms have witnesses (w_1', w_2') . See also the second remark following Security Suggestion 4.6 on p.57. A drawback of Brands e-cash with such simpler witnesses is that it is only computationally secure against holder framing by the bank.

Brands has proposed a number of interesting extensions to his scheme, which are worth to mention. It is straightforward to encode different coin values or different currencies into the types of credentials. Assuming a binary encoding, i.e., different types for powers of 2 (1, 2, 4, ...), and coin values from 1 to 2^k , the average value requires $k/2$ coins. Brands has shown, that the scheme can be extended so as to require only one single coin for each value between 0 and 2^k [34] p.42.

Brands has also shown how the CC Mechanism 5.10 on p.111 can be extended to give an issue-wise unlinkable K -overshow detecting (and preventing) coin scheme. The idea is to split the target pseudonym's witness into $K + 1$ parts and to share these among $K + 1$ instead of two responses during the show protocol. This can be achieved by using polynomials of degree K (instead of 1) in the verifier's challenge c . For more details see Brands [34] p.43.

Security Suggestion 5.2 CC Mechanism

In the random oracle model, the CC Mechanism is a 1-overshow preventing and detecting coin credential scheme under the following assumptions:

- (i) Type unforgeability holds under Security Suggestion 4.6 on p.57 that the Blind Chaum-Pedersen(3) Signature Mechanism 4.8 on p.55 is restrictive and further heuristic arguments by Brands in [34] Proposition 12.
- (ii) 1-overshow prevention holds under the intact observer assumption using a perfect biometric recognition characteristic and under the SR-Assumption 3.21 on p.30.
- (iii) 1-overshow detection holds under the SDL Assumption 3.20 on p.29, the SDH Assumption 3.23 on p.30 and further heuristic arguments of Brands for his e-cash scheme [34,35].
- (iv) Show-wise fail-safe unlinkability holds under some heuristic argument of Brands [34,35]. ◆

Security Considerations

Mechanism 5.10 on p.111 differs slightly from the original proposal of Brands in two points: the protocol *intro* for introducing new source pseudonyms explicitly and the way how a MUD and its observer split their joint witness of a source pseudonym into a co-witnesses and a witness. Where Brands proposal uses equal witness and co-witness domains, i.e., $W = V = \mathbb{Z}_q^3$ with vector addition modulo q , we use the domains and addition of ECEG(3).

EFFECTIVENESS

Effectiveness condition (1) on p.79 of Definition 5.1 on p.76 can be shown similarly as for the PC Mechanism 5.8 on p.92 because protocol $intro^{(CC)}$ differs from protocol $intro^{(PC)}$ only by not involving biometric recognition and using an auxiliary witness component w^* and a corresponding auxiliary intro pseudonyms component ψ^* , which are chosen and kept by the observer.

Effectiveness condition (2) on p.79 of Definition 5.1 on p.76 is immediate from effectiveness of the Chaum-Pedersen(3) Signature Mechanism because the MUD outputs exactly the pseudonym and signature that was produced by $signBlind$. This also guarantees that the verifier proceeds in step (2) of $show$.

Effectiveness condition (3) on p.80 of Definition 5.1 on p.76 is also satisfied: We first verify that protocol $show$ runs successfully by checking the conditions in step (2), step (6) and step (8). V 's condition in step (2) is satisfied because the credential χ'' has been computed as a Chaum-Pedersen(3) signature for the target pseudonym $\psi' = (\bar{\psi}', \underline{\psi}')$ using the signing key rk_t of type t . D 's condition in step (6) of $show$ is verified as follows:

$$make(t') = make(w' + c'w^*) = make(w) \cdot make(c'w^*) = \psi_O \cdot \psi_O^*{}^{c'}$$

V 's condition in step (8) of $show$ is verified as follows:

$$\begin{aligned} \underline{\psi}' \cdot (\bar{\psi}' / \underline{\psi}')^c &= \varphi^{\omega_1} \cdot \bar{\varphi}^{-\omega_1 c} && \text{issue step (4), (5), (3)} \\ &= \psi^{\omega_1} \psi_O^*{}^{\omega_2} \cdot g^{\omega_3} \cdot \psi_O^*{}^{-\omega_2 c} \omega_1^{c-\omega_3 c} g_3 && \text{step (2)} \\ &= \psi^{\omega_1} \psi_O^*{}^{\omega_2(1-c)} \cdot g_3^{\omega_3 + c(\omega_1 - \omega_3)} && \text{simplify} \\ &= make(\omega_1(\omega w + u)) make(\omega' c' w^*) \cdot g_3^{\omega_3 + c(\omega_1 - \omega_3)} && \text{show step (4), (7)} \\ &= make(\omega' w' + u) make(\omega' c' w^*) \cdot g_3^{\omega_3 + c(\omega_1 - \omega_3)} && \text{issue step (8)} \\ &= make(\omega'(w' + c'w^*) + u + (0, 0, \omega_3 + c(\omega_1 - \omega_3))) && \text{simplify} \\ &= make(\omega' t' + u + (0, 0, \omega_3 + c(\omega_1 - \omega_3))) && \text{show step (5)} \\ &= make(t) \pmod{p} . && \text{show step (7)} \end{aligned}$$

The only extra condition to settle is that the co-witness (ω', u') co-makes the intermediate pseudonym ψ' with respect to the intro pseudonym ψ_O :

$$make'(\psi_O, (\omega', u')) = make'(\psi_O, (\omega\omega_1, u\omega_1)) = (make'(\psi_O, (\omega, u)))^{\omega_1} = \varphi^{\omega_1} = \bar{\psi}' .$$

The optional effectiveness conditions (4) on p.80 and (5) on p.80 do not apply to the CC Mechanism 5.10 on p.111.

TYPE UNFORGEABILITY

Consider an attacker coalition $\tilde{D}, \tilde{O}_1, \dots, \tilde{O}_L, \tilde{V}$ that has never been issued a type t credential, but may have verified a number of type t credentials.

Step A In order to show a type t credential, the attacker coalition must provide to an honest verifier

- (i) a target pseudonym ϕ , and a Chaum-Pedersen(3) signature χ'' valid for ϕ with respect to \overline{pk}_t after step (1) of *show* in Figure 5–15 on p.114 and
- (ii) and a valid witness t of ϕ in step (4) through step (8) of *show* in Figure 5–15 on p.114.

Since the attacker coalition is presumed to have not been issued a type t credential in the past and to have no access to an honest observer that has been used to get a type t credential issued in the past, the attacker coalition can achieve (i) and (ii) above either by

- (B) choosing a target pseudonym ϕ such that it knows a witness for it in order to succeed in (ii) and forging a Chaum-Pedersen(3) signature for ϕ in order to succeed in (i), or
- (C) replaying a target pseudonym ϕ and a valid signature χ' in order to succeed in (i) and somehow try to succeed (ii) afterwards.

We consider each alternative in turn:

Step B The attacker coalition aims to construct the target pseudonym $\phi = (\overline{\psi}', \underline{\psi}')$ such that it knows witnesses for both $\overline{\psi}'$ and $\underline{\psi}'$, each with respect to generators g_1, g_2, g_3 , and to construct a valid Chaum-Pedersen(3) signature for ϕ with respect to $pk_t = g^{rk_t}$. Since the generators g_1, g_2, g_3 and g are all chosen uniformly at random and independently of each other, Brands [34] Proposition 12, 2. applies, saying that there really is no way for a computational attacker (even if he had access to the honest signer) to construct a signature for a message $\phi = (\overline{\psi}', \underline{\psi}')$, while the attacker knows a witness of $\overline{\psi}'$ with respect to g_1, g_2, g_3 .

Step C The attacker coalition has only a negligible chance of convincing an honest verifier that it knows a witness of the target pseudonym $\phi = (\overline{\psi}', \underline{\psi}')$ that has been captured and replayed from an honest MUD with an honest observer. Assume to the contrary that the attacker has sent the target pseudonym $\phi = (\underline{\psi}', \overline{\psi}')$ to the verifier in step (1) of Figure 5–15 on p.114 and afterwards were capable to respond convincing witnesses $s = (s_1, s_2, s_3)$ and $t = (t_1, t_2, t_3)$ for two respective challenges $b, c \in \mathbb{Z}_q^* \setminus \{1\}$, $b \neq c$, which satisfy the condition in step (8), i.e.

$$make(s) = \underline{\psi}' \cdot (\overline{\psi}' / \underline{\psi}')^b \pmod{p} \quad \text{and} \quad make(t) = \underline{\psi}' \cdot (\overline{\psi}' / \underline{\psi}')^c \pmod{p} . \quad (5.22)$$

Take the first equation to the power of $1 - c$, the second equation to the power of $b - 1$ and consider the product of the resulting equations:

$$make(s)^{1-c} make(t)^{b-1} = \overline{\psi}'^{1-c} \underline{\psi}'^{b-1} = \overline{\psi}'^{b-c} \pmod{p} .$$

We take the $(b - c)$ -th root of this equation and use the homomorphic properties of function *make* according to Lemma 4.1 on p.44 to find a witness of the intermediate pseudonym component $\overline{\psi}'$:

$$make\left(\frac{(1-c)s + (b-1)t}{b-c}\right) = \overline{\psi}' \pmod{p} .$$

Obviously, the witness $((1 - c)s + (b - 1)t) / (b - c)$ is well-defined and can be computed by an attacker in polynomial time. This contradicts the Subgroup Representation Assumption 3.21 on p.30.

1-OVERSHOW PREVENTION

Brands has shown in [35] Proposition 16 that, under the intact observer assumption, an attacking MUD \tilde{D} can show a coin only by asking its observer to use its respective witness shares w', w^* in step (5) of Figure 5–15 on p.114. Could \tilde{D} come up with w', w^* itself, it had found a representation of ψ_O in *intro* (Figure 5–13 on p.112), which contradicts the SR-Assumption 3.21 on p.30. The honest observer hosted by \tilde{D} therefore prevents \tilde{D} from showing any coin more than once by refusing to use any of its witness shares in more than one execution of *show* (see the condition “ w' not yet used” in step (5) of Figure 5–15 on p.114).

1-OVERSHOW DETECTION

Brands has shown in [35] Proposition 18 that, under the broken observer assumption, an attacking MUD \tilde{D} will be identified by its co-witness (ω, u) and the witness w of its observer if it shows any of its coin credentials more than once. The same arguments carry over to the CC Mechanism 5.10 on p.111 above because it differs from Brands original proposal only in the way how the witness of a source or target pseudonym is split up between the MUD and its observer. Under the broken observer assumption, however, a MUD and its observer are essentially one participant, such that the MUD knows the complete witnesses of all its source and target pseudonyms anyway, which leaves the particular splitting into witnesses and co-witnesses irrelevant.

SHOW-WISE FAIL-SAFE UNLINKABILITY

Brands has argued in [35] that his e-cash mechanism is 1-show-wise fail-safe unlinkable (in our terminology). We show the same property holds for the CC Mechanism 5.10 on p.111. Since the CC Mechanism 5.10 on p.111 uses essentially different witnesses for target pseudonyms than for source pseudonyms, it is not possible to re-use target pseudonyms as source pseudonyms and therefore, only protocol $Life_n$ is defined according to Definition 5.6 on p.87, whereas protocol $life_n$ is not. Since each coin is to be shown only once, we need to consider protocol $Life_1$ only.

We show that the honest MUD in $Life_1$ blocks any in-band inflow to the observer and any in-band outflow from the observer. Consider $2n + 3$ participants of protocol $Life_1$ (see Section 5.3.4 on p.84), namely an attacking issuer \tilde{I} of type t credentials, an honest MUD D , 2 attacking observers \tilde{O}_0, \tilde{O}_1 and one attacking credential verifiers \tilde{V}_1 . Consider 2 successful executions E_0, E_1 of $Life_1$, where each execution E_i ($i \in [0, 1]$) is run by $\tilde{I}, D, \tilde{O}_i$ and \tilde{V}_1 . Now consider the resulting views¹¹ of all attacking participants on the honest MUD D according to Table 5-5 on p.87 of Definition 5.6 on p.87.

The issuer participates in execution E_0 of $Life_1$ during *intro* and *issue*. Thus the issuer’s view $View_0^{\tilde{I}}$ contains an output of *signBlind* (step (5) of *intro*), a verifier’s view of *proveExt* (step (7) of *intro*) and the signer’s view of *signBlind* (step (5) of *issue*). This is summarized in the first two rows of the following Table 5-11 on p.119. The views of attackers \tilde{O}_0 and \tilde{V}_1 are listed in the following rows. The last column lists all internal choices of the MUD in respective member protocols of $Life_1$.

11) Of the views of \tilde{I}, \tilde{O}_i and \tilde{V}_i on D we are only interested in the messages exchanged with D , not in the internal choices of \tilde{I} and \tilde{O} (see Definition 3.11 on p.21). So in order to relax the view notation, we do not display the internal choices of \tilde{I}, \tilde{O}_i and \tilde{V}_i in the remaining argument.

Attacker	member prot. of $Life_1$	Attacker's view (including shared inputs) contains	Internal Choice of MUD
issuer \tilde{I} ($View_{\tilde{I}}^0$)	<i>intro</i> (Figure 5-13)	observer's native verifying key pk_O , output ψ, χ_O of <i>signBlind</i> (step (5)), verifier's view of <i>proveExt</i> (step (7))	(see below under <i>intro</i> for observers)
	<i>issue</i> (Figure 5-14)	signer's view in <i>signBlind</i> (step (5))	(see below under <i>issue</i> for observers)
observer O_0 ($View_{O_0}^0$)	<i>intro</i> (Figure 5-13)	$\psi_D, \psi_O, \psi_{O^*}$ (step (2) and (3)), signer's view of <i>signBlind</i> (step (5)), prover's view of <i>proveExt</i> (step (7))	$(\omega, u) \in_R V_{prek}$ in <i>signBlind</i> (step (5)) in <i>proveExt</i> (step (7))
	<i>issue</i> (Figure 5-14)	—	$\omega_1, \omega_2, \omega_3 \in_R \Omega_{prek}^{(CP)}$ in <i>signBlind</i> (step (5))
	<i>show</i> (Figure 5-15)	c' (step (4)) t' (step (8))	none
verifier \tilde{V}_1 ($View_{\tilde{V}_1}^i$)	<i>show</i> (Figure 5-15)	verifying key pk_t , (ψ', χ') (step (1)), c (step (3)) and t (step (7))	(see above under <i>show</i> for observers)

TABLE 5-11 Attackers and their views on the MUD in $Life_1$

“ ≤ 1 ”: We show that any such 3-tuple of valid views $(view_{\tilde{I}}^0, view_{O_0}^0, view_{\tilde{V}_1}^i)$ originating from 2 executions of $Life_1$ according to Definition 5.6 on p.87 matches with at most one value of the internal choice of the MUD in $Life_1$, which is detailed in the last column of Table 5-11 on p.119.

- (i) The issuer's and the observer's views in *intro* together determine the MUD's co-witness (ω, u) up to q possible values as follows: $\omega = \log_{\psi_D \psi_O} \psi$ (from step (5)) and $u \in make^{-1}(\psi_D^\omega)$ (from step (2)). They also determine a unique internal choice of the MUD in member protocol *signBlind* (step (5)), and, for each possible co-witness (ω, u) , they determine a unique internal choice of the MUD in member protocol *proveExt* (step (7)). This follows from blindness of *signBlind* (Security Suggestion 4.6 on p.57) and from co-unlinkability of *proveExt* (Security Suggestion 4.4 on p.48), respectively.
- (ii) For each possible co-witness (ω, u) , the issuer's view in *issue* and the verifier's view in *show* uniquely determine the internal choices $\omega_1, \omega_2, \omega_3 \in \Omega_{prek}^{(CP)}$ of the MUD in step (1) of *issue* as follows:
 - $\omega_1 = \log_{\varphi} \bar{\psi}'$ is determined uniquely according to protocol *issue* step (5) by the issuer's view of *issue*. For each possible co-witness (ω, u) of the source pseudonym ψ , ω_1 determines the MUD's co-witness $(\omega', u') = (\omega_1 \omega, \omega_1(u + (0, 0, 1)))$ of the intermediate pseudonym ψ' according to protocol *issue* step (8).
 - The value $\omega_2 = \frac{\omega' c'}{1-c} \bmod q$ ($c \neq 1$) is determined uniquely according to protocol *show* step (4) by the views of the verifier and the observer in protocol *show*.
 - The value $\omega_3 = (t_3 - c\omega_1) / (1-c) \bmod q$ is determined uniquely according to protocol *show* step (7) by the views of verifier and observer in protocol *show*.
- (iii) Finally, the component u' of the MUD's co-witness $v' = (\omega', u')$ of the intermediate pseudonym ψ' is uniquely determined as $u'_1 = t_1 - \omega' t'_1$ according to protocol *show* step (7). This determines the component $u = (u_1, u_2, 0) = \omega_1^{-1}(u'_1, u'_2, u'_3) - (0, 0, 1)$ of the MUD's co-witness

5 CREDENTIAL SCHEMES

$v = (\omega, u)$ of the source pseudonym ψ according to protocol *issue* step (8). If this value u is such that $make(u) = \psi_D^\omega$ (see step (2) of *issue*), then we have determined exactly one choice of the MUD, otherwise none.

“ ≥ 1 ”: We are left to show that in fact: $make(u) = \psi_D^\omega$ according to (iii) above.

We start by expanding the components u_1, u_2 of u . For $i = 1, 2$ we find:

$$\begin{aligned}
 u_i &= \omega_1^{-1} u_i' && \text{rewrite } u_i \text{ (issue step (8))} \\
 &= \omega_1^{-1} (t_i - \omega' t_i') && \text{rewrite } u_i' \text{ (show step (7))} \\
 &= \omega_1^{-1} (t_i - \omega_1 \omega t_i') && \text{rewrite } \omega' \text{ (issue step (8))} \\
 &= \omega_1^{-1} t_i - \omega (w_i + c' w_i^*) && \text{rewrite } t_i' \text{ (show step (5))} \quad (5.23)
 \end{aligned}$$

We follow a heuristic argument of Brands [34] that the observer must have chosen w and w^* in protocol *show* step (5) such that they are respective witnesses of ψ_O and ψ_O^* . Otherwise, the honest MUD D would not be convinced in protocol *show* step (8), therefore would abort the protocol, and thus produce an invalid observer's view. We continue rewriting (5.23):

$$\begin{aligned}
 &= \frac{t_i}{\omega_1} - \omega \left(w_i + \frac{\omega_2}{\omega'} (1 - c) w_i^* \right) && \text{rewrite } c' \text{ (show step (4))} \\
 &= \frac{t_i}{\omega_1} - \omega w_i - \frac{\omega_2}{\omega_1} (1 - c) w_i^* \pmod{q} . && \text{rewrite } \omega' \text{ again} \quad (5.24)
 \end{aligned}$$

Next, we rewrite the expression $make((t_1, t_2, 0))$:

$$\begin{aligned}
 make((t_1, t_2, 0)) &= \underline{\psi}^{1-c} \overline{\psi}^c g_3^{-t_3} && \text{according to show step (8)} \\
 &= \underline{\varphi}^{\omega_1(1-c)} \overline{\varphi}^{c\omega_1} g_3^{-t_3} && \text{rewrite } \underline{\psi}' \text{ (issue step (4))} \\
 &&& \text{and } \overline{\psi}' \text{ (issue step (5), (2))} \\
 &= (\underline{\psi} \overline{\psi} \psi_O^*)^{\omega_2/\omega_1} (\underline{\omega}_3/\overline{\omega}_1)^{\omega_1(1-c)} (\underline{\psi} g_3)^{\omega_1 c} g_3^{-t_3} && \text{rewrite } \underline{\varphi} \text{ (issue step (2)),} \\
 &&& \text{and } \overline{\varphi} \text{ (issue step (2), (3))} \\
 &= \underline{\psi}^{\omega_1} \overline{\psi}^{\omega_2(1-c)} . && \text{simplify.} \quad (5.25)
 \end{aligned}$$

Finally, we evaluate the expression $make(u)$:

$$\begin{aligned}
 make((u_1, u_2, 0)) &= make\left(\frac{(t_1, t_2, 0)}{\omega_1} - \omega w - \frac{\omega_2}{\omega_1} (1 - c) w^*\right) && \text{use result (5.24) on p.120} \\
 &= make(t_1, t_2, 0)^{1/\omega_1} make(w)^{-\omega} make(w^*)^{-(\omega_2/\omega_1)(1-c)} && \text{Lemma 4.1 on p.44} \\
 &= (\underline{\psi}^{\omega_1} \overline{\psi}^{\omega_2(1-c)})^{1/\omega_1} \underline{\psi}_O^{-\omega} \overline{\psi}_O^{-(\omega_2/\omega_1)(1-c)} && \text{use result (5.25) on p.120,} \\
 &&& \text{and issue step (2), (3)} \\
 &= \underline{\psi} \overline{\psi} \psi_O^*^{(\omega_2/\omega_1)(1-c)} \underline{\psi}_O^{-\omega} \overline{\psi}_O^{-(\omega_2/\omega_1)(1-c)} && \text{simplify}
 \end{aligned}$$

$$\begin{aligned}
&= \psi \psi_O^{-\omega} && \text{cancel out } \psi_O^{-(\omega_2/\omega_1)(1-c)} \\
&= \psi_D^\omega . && \text{intro step (5)}
\end{aligned}$$

5.6 Bond Credential Schemes

Bond credential schemes can be defined as credential schemes that combine the holder authorization rules of personal and coin credential schemes.

5.6.1 Definition

Definition 5.11 Bond Credential Scheme

A bond credential scheme with credential types T is a type-unforgeable, transfer preventing, K -overshow preventing or K -overshow detecting and show-wise unlinkable credential scheme with credential types T and recognition characteristic $(FAR, FRR) = (1, 0)$. ♦

5.6.2 Cryptographic Mechanism

A 1-overshow preventing bond credential scheme is easily constructed by modifying the CC Mechanism 5.10 on p.111 slightly. Just insert the biometric recognition operations of the PC Mechanism into the observer's part of the protocols *intro* and *show*.

5.7 Conjunctions of Personal and Coin Credentials

Real life is full of examples where individuals need to show 2 or more different types of legitimations in order to obtain another legitimation. For example, to get on an international flight one must present an air ticket and an ID document. Such AND combinations of different credentials are easily implemented if holders can be asked to present their MUDs physically at the verifier's site. The verifier uses physical and organizational means to ensure he is shown credentials by the same MUD and observer.

However, if a holder wants to combine 2 or more of her credentials remotely, the verifier needs to verify each credential separately, and he must be convinced that the credentials are shown by the same holder. The latter can be achieved for personal credentials and overshow preventing coin credentials by showing them for the same target pseudonym. This convinces a verifier because personal credentials and transfer preventing coin credential mechanisms both require an observer and this observer keeps the witness of an intro pseudonym that is required to show either type of credential.

In order to prepare certain applications in Section 8 on p.145, we only consider how to show the particular credentials of the PC Mechanism and of the CC Mechanism for equal target pseudonyms. We first observe that the PC Mechanism 5.8 on p.92 and the CC Mechanism 5.10 on p.111 observe the same domain families for their source pseudonyms, which are both represented with respect to two generators $g_1^{(PC)}, g_2^{(PC)}$ and $g_1^{(CC)}, g_2^{(CC)}$, respectively. In contrast, the domain families of target pseudonyms of the CC Mechanism are pairs $(\bar{\phi}, \underline{\phi})$, where each component is represented with respect to three generators. CC target pseudonyms are more complex than PC target pseudonyms in order to achieve the capability of double spending detection. Since both components of a CC target pseudonym

5 CREDENTIAL SCHEMES

can be verified to belong together (see the verifying operation of Mechanism 5.10 on p.111), it suffices to consider a modified PC Mechanism

- with target pseudonyms taken from the same domains as either component of a CC target pseudonym and
- that allows to represent its target pseudonyms with respect to 3 generators.

The price of this approach is that there is no obvious way to re-use these target pseudonyms as source pseudonyms of either the PC Mechanism or the CC Mechanism, but we will not need that feature in the following.

Next, we show how to modify the PC Mechanism such that the protocol $show^*$ (Figure 5–12 on p.97) allows to show PC credentials for the leading component $\bar{\phi}$ of a CC target pseudonym $(\bar{\phi}, \underline{\phi})$ introduced implicitly via a previous execution of protocol $show^{(CC)}$ (Figure 5–15 on p.114). The leading component $\bar{\phi}$ can be represented by the MUD D with respect to the intro pseudonym ψ_O and three generators $(g_1, g_2, g_3) = (g_1^{(CC)}, g_2^{(CC)}, g_3^{(CC)})$ as follows:

$$\bar{\phi} = \psi_O^{\omega\omega_1} g_1^{\omega_1 u_1} g_2^{\omega_1 u_2} g_3^{\omega_1} \pmod{p} . \quad (5.26)$$

The modified PC Mechanism is set up by sharing the three generators g_1, g_2, g_3 and the respective discrete log setting with the CC Mechanism, instead of using two independent generators $g_1^{(PC)}, g_2^{(PC)}$ of an independent discrete log setting. In addition, the following modifications are made to Definition 5.7 on p.90:

- The Chaum-Pedersen(3) Signature Mechanism is used in place of the Chaum-Pedersen(2) Signature Mechanism.
- The *ECEG*(3) Mechanism is used in place of the *ECEG*(2) Mechanism.

The MUD D in protocol $show^*$ in Figure 5–12 on p.97 takes the input co-witness $v'' = (\omega\omega_1, \omega_1 u_1, \omega_1 u_2, \omega_1)$ for a given CC target pseudonym $(\bar{\phi}, \underline{\phi})$ such that

$$\bar{\phi} = \psi_O^{\omega\omega_1} g_1^{\omega_1 u_1} g_2^{\omega_1 u_2} g_3^{\omega_1} = g_1^{\omega_1(u_1 + \omega w_1)} g_2^{\omega_1(u_2 + \omega w_2)} g_3^{\omega_1} \pmod{p} . \quad (5.27)$$

Accordingly, the proof of knowledge ($proveWitRel$) in step (3) of $show^*$ is taken from *Brands*(4) (see Mechanism 4.5 on p.46) and uses the 4 generators ψ_O, g_1, g_2, g_3 . The extended proof of knowledge ($proveExt$) in step (7) of $show^*$ is taken from *ECEG*(3) (see Mechanism 4.6 on p.47) and uses the 3 generators g_1, g_2, g_3 .

6

Group Signature Schemes

In this section, a modular refinement of blind signature schemes (Section 4 on p.33) is provided and thereby a modular refinement of all credential schemes of Section 5 on p.59 is prepared. A *group oriented signature scheme* or *group signatures scheme* allows each member of a group to sign messages in behalf of the group without revealing one's identity. However, in case of a dispute later on, the manager or center of the group has enough evidence to identify and prove who has produced the disputed signature.

Our program is as follows. In Section 6.1 on p.123, we review the existing work on group oriented signatures. In Section 6.2 on p.125, the new achievements are summarized. In Section 6.3 on p.126 formal definitions are given for blind group signature schemes. In Section 6.4 on p.129 a practical blind group signature mechanism is presented.

6.1 Overview of Existing Literature

Since Boyd [33] presented a first paper on the subject in 1986, group-oriented digital signing has attracted increasing interest. We distinguish two basic classes of group-oriented signatures, tabulate their important requirements and give a short overview of the existing literature.

6.1.1 Options and Features

Threshold Signatures allow signers to form *signing groups* and sign in co-operation, so as to enforce some dual control over what can be signed. The special case where all signers of a group must co-operate was first introduced by Boyd [33] as *multisignatures*.

Group Signatures allow signers to stay anonymous within an *anonymizing group* of potential signers. An additional option is if, in case of disputes, the signer can later be *re-identified*, and if so, whether re-identification requires interaction with the signer or not. An important security requirement arising for re-identifiable group signatures is *unframeability*, i.e., security of group members against being falsely accused of having signed a message. Typical degrees of unframeability are *computational*, i.e., it is hard to compute a signature for which a non-signer is held responsible, and *uncondi-*

tional, i.e., up to a negligible probability of guessing, it is impossible even for a computationally unlimited attacker to produce a signature for which a non-signer is held responsible.

The security requirements specific to threshold and group signatures are as follows:

Anonymity of Signer(s) can be either computational or unconditional depending on whether signers remain anonymous against polynomial-time attackers or even against computationally unlimited attackers (possibly with a small error probability). In the following, we focus on computational signer anonymity, which appears to have efficient implementations and is thus more relevant in practice as long as certain complexity theoretic assumptions hold. In threshold signature schemes, it is an option to keep the actual subset of signers anonymous to the verifiers. In group signature schemes the signers are kept anonymous to the verifiers by definition.

Management of Group of Signers

Centralized/Decentralized: A group signature scheme is called *centralized*, if joining, revoking and—if available—re-identifying group members, i.e., group management, is done by some group center, which registers each group member in advance. If no group managing center exists, the group signature scheme is called *decentralized*.

Static/Dynamic: A group signature scheme is called *static* if the public group key, which is needed in order to verify signatures of any member of the group, needs to be rebuilt every time a new member joins the group or a former member leaves the group. Otherwise, a group signature scheme is called *dynamic*.

Granularity of Access Structure: The set of authorized signing groups can be specified for example by group size, or by enumeration of all members of all signing groups¹.

6.1.2 Existing Work

Threshold-signatures were first proposed by Boyd and were further developed by Desmedt and Frankel [91,111], Croft and Harris [81] and by Ohta, Okamoto [170,171]. There seems to be growing consensus to call this class “threshold signatures”. For example, see Desmedt’s overview [90]. More work on combinations of threshold and anonymity requirements is found in the work of Cramer, Damgård and Schoenmakers [79].

The concept and the first four constructions of static group signatures are due to Chaum and van Heijst [70]. Three of those constructions give computational anonymity and the fourth gives unconditional anonymity to the signer. Decentralized group management is not considered in their paper. Chen and Pedersen [77] proposed two static group signature schemes with centralized or decentralized group management; one with computational, the other with unconditional anonymity and both with computational unframeability. In their constructions, the group center can be replaced by a threshold primitive so that any k out of n members of the signer’s group can identify a signer. Their basic idea to achieve re-identification is to have the signer choose two private keys, one of which she keeps by herself, whereas the other is submitted to her group center. Group members then sign messages by producing two signatures, one with respect to each of their two private keys. If some part of the signa-

1) This is actually a generalization of threshold signature schemes.

ture is deterministic, then the group center, who holds one private key of each signer, can later re-sign messages of any disputed signature with the submitted private keys of each group member and see which one fits (the deterministic component of) the disputed signature. For obvious reasons, this technique is called *double signing*.

Camenisch later proposed a group signature with centralized or decentralized group management, computational anonymity and unconditional unframeability [47]. Recently, Camenisch and Stadler [51] have proposed the first practical dynamic group signature scheme with public group keys and signatures of constant size (relative to the group size). A blind version for their scheme was presented by Lysyanskaya and Ramzan in [149] and another by Nguyen, Mu and Varadharajan [167]. Ateniese and Tsudik [6,7] have proposed two group signature schemes of similar efficiency, but Traoré [223] has shown that both of them are vulnerable to attacker coalitions who produce group signatures for which no-one in the coalition is held responsible. Traoré [223] himself has proposed an even more efficient group signature scheme, which is based on different but more ad-hoc complexity-theoretic assumptions. Apparently, there is room left for new group signature schemes with better trade-offs between (provable) security and efficiency.

6.2 New Achievements

A formal definition of blind group signatures with static group management is given. A blind group signature mechanism is presented that is based on a group signature scheme by Chen and Pedersen ([77] Section 4) and follows Bleumer [21]. The new mechanism is as efficient as the original by Chen and Pedersen in terms of signer effort and communication. The basic idea of Chen and Pedersen to make signers identifiable is to require two signatures for each message with respect to different keys (double signing). This leads to a protocol where almost all operations are executed twice on different data. We carry out this “double signing” by only one execution of the original protocol instead of two. This improvement also applies to the blind group signature mechanism to be presented.

The original CPGS Mechanism ([77] Section 4) uses public group keys and signatures of size linear in the size of the anonymizing group. Likewise, the computation effort is linear in the size of the anonymizing group. In contrast, the group signature scheme by Camenisch and Stadler [51] uses public keys and signatures of significant but constant size (relative to the size of the anonymizing group). The break even point between CPGS and Camenisch-Stadler, below which CPGS is more efficient, is an anonymizing group size of about 100 members. It is about 200 members for the improved CPGS version. This appears to be sufficient for many practical purposes.

The CPGS Mechanism allows only static group management, i.e., every time a member joins or leaves the group, the public (and private) group key needs to be updated. Another promising approach is the blind group signature mechanism with dynamic group management proposed by Lysyanskaya and Ramzan in [149], which is based on work of Camenisch, Stadler [51]. Moreover, the length of their public keys and signatures is independent of the group size. However, its security rests on more non-standard complexity theoretic assumptions than the one given here and it has not been analyzed with respect to one-timeness and restrictiveness yet.

6.3 Blind Static Group Signature Schemes

In order to refine blind signature schemes into blind static group signature schemes, we basically split the key generating operation and add an operation by which signers can be identified. Key generation is split into an operation that generates individual keys of a group member and one that takes individual keys and generates the group keys. The former is used by group members, whereas the latter is used by group centers. In case a group signature is disputed, the group center can identify the originator by the operation *identify*. The splitting of key generation serves the sole purpose of explicitly distinguishing between attacking group members and attacking group centers. The definition abstracts from the problem of updating group keys when new members join the group or previous members leave it. More work is needed to take these practically important features into account.

The presence of an operation *identify* introduces two new threats, namely framing by attacking group centers and truly anonymous signatures of attacking group members. Framing is the false accusation by a collusion of attacking group members and the group center that another group member has produced a certain signature. Truly anonymous signatures occur if one or more group members manufacture a signature for which none of them is later identified. The case where another group member is identified falls under framing, but it can also happen that no group member at all is identified.

Definition 6.1 Blind Static Group Signature Schemes

A one-time restrictive blind group signature scheme of group size n is a one-time restrictive blind signature scheme (Definition 4.7 on p.53) with the following additions.

DOMAINS:

In addition to the domain families set out in Definition 4.7 on p.53, a one-time restrictive blind static group signature scheme has the following 2 additional domain families PI , RG whose respective members are:

- The *public individual key* domains PI_{prek} and
- the *private group key* domains RG_{prek} .

The private keys of a (blind) static group signature scheme are those generated and held by individual group members, while the public keys are those by which anyone can verify all signatures of all signers of a particular group. Public keys will be referred to more specifically as *public group keys*.

In order to ease notation, we do not display the domain index $prek$ in the following.

OPERATIONS:

In addition to the operations set out in Definition 4.7 on p.53, a one-time restrictive blind group signature scheme has a set $\{genIKey, genGKey, identify, verifyId\}$ of 4 additional operations satisfying 4 additional requirements: group effectiveness, signer anonymity, unframeability and coalition resistance.

To enhance readability, we call the signing operation of a group signature scheme *gSignBlind* instead of *signBlind*.

Generate Individual Keys

$$(rk_i, pi_i) \leftarrow genIKey(prekey) \quad , \text{ where } i \in [1, n]$$

A probabilistic operation that takes a prekey $prek$ and returns a pair (rk_i, pi_i) of a *private individual key* and a corresponding *public individual key*, respectively.

Generate Group Keys $(rg, pk) \leftarrow genGKey(prekey, (pi_1, \dots, pi_n))$

A probabilistic operation that takes a prekey $prek$ and an n -tuple of public individual keys $pi_1, \dots, pi_n \in PI$, and outputs a pair $(rg, pk) = RG \times PK$ of a *private group key* rg and a *public group key* pk , which we also call the *public key* (of the group of individuals who have contributed their public individual keys to $genGKey$).

The operation $genKey$ of a one-time restrictive blind signature scheme is then defined as follows:

$$(rk, pk) \leftarrow genKey(prekey), \text{ where } rk = (rk_1, \dots, rk_n) \text{ and}$$

$$(rg, pk) \leftarrow genGKey(prekey, (pi_1, \dots, pi_n)) \text{ and for } i = 1 \dots n: (rk_i, pi_i) \leftarrow genIKey(prekey) \quad .$$

Identify $pi \leftarrow identify((rg, pk), m, \sigma)$

A deterministic operation that takes a pair $(rg, pk) \in RG \times PK$ of a private group key and a public key, a message $m \in M$ and a signature $\sigma \in \Sigma$. The result is a public individual key $pi \in PI$ or it is undefined, denoted \perp .

Verify Identity $accept \leftarrow verifyId((pi, pk), m, \{m^*\}, \sigma)$

A deterministic operation that takes a public individual and a public key $(pi, pk) \in PI \times PK$, a message $m \in M$, an optional message $m^* \in M$ and a signature $\sigma \in \Sigma$ and returns a Boolean result. If the result is *TRUE* then we say that the group member with public individual key pi is held *responsible* for group signature σ (on message m).

In addition to effectiveness, one-timeness, restrictiveness and blindness according to Definition 4.7 on p.53, a static blind one-time restrictive group signature scheme satisfies the following additional requirements.

GROUP EFFECTIVENESS

- (i) If Alice is registered by her public individual key pi_i and later produces a signature σ for message m , then she will be identified by pi_i as the originator of m .

More precisely: Let $prek \in [genPrekey(k)]$ be a prekey, $(rk_i, pi_i) \in [genKey(prekey)]$ be Alice's individual key pair, i some index in the range $[1, n]$, $pi_j \in PI$ be arbitrary individual keys for $j = 1, \dots, i-1, i+1, \dots, n$, and $(rg, pk) \leftarrow [genGKey(prekey, (pi_1, \dots, pi_n))]$ be the corresponding group key pair. If Alice produces a signature for message(s) $m, \{m^*\} \in M$ while Bob uses some blinder $\omega \in \Omega$:

$${}^B[m', \sigma'] \leftarrow gSignBlind([pk, m], {}^A[rk_i], {}^B[\{m^*\}, \omega]) \quad ,$$

and if the signature is later disputed, then Alice is identified as the signer of $m, \{m^*\}$:

6 GROUP SIGNATURE SCHEMES

$$pi_i \leftarrow \text{identify}((rg, pk), m, \sigma) \quad \text{and} \quad \text{verifyId}((pi_i, pk), m, \{m^*\}, \sigma) = \text{TRUE} .$$

SIGNER ANONYMITY (COMPUTATIONAL)

The following definition roughly follows ideas of Chen and Pedersen [77] Section 5.

For all $n \in \mathbb{N}$, all probabilistic polynomial-time Turing machines \tilde{B} and each positive constant $c > 0$, all sufficiently large k , the probability of the following event is $< \frac{1}{n} + c^{-k}$. After choosing a prekey $prek \leftarrow \text{genPrekey}(k)$ and n individual key pairs $(rk_i, pi_i) \leftarrow \text{genIKey}(prek)$ for $i \in [1, n]$, and a corresponding group key pair $(rg, pk) \leftarrow \text{genGKey}(prek, (pi_1, \dots, pi_n))$, the attacker \tilde{B} takes as input the values $n, k, prek$ and the public (group) key pk , and repeats the following two steps a number of times that is bounded above by a polynomial in k :

- (i) \tilde{B} chooses a message $m_j \in M$, an optional second message $m_j^* \in M$, a group member index i_j , and requests a group signature for $(m, \{m^*\})$ from group member A_{i_j} .
- (ii) Group member A_{i_j} engages in protocol $g\text{SignBlind}$ with the attacker \tilde{B} on respective inputs: $\tilde{B}[\bullet, \bullet] \leftarrow g\text{SignBlind}([pk, m_j], [rk_{i_j}, \tilde{B}[\{\bullet\}, \bullet]])$. Since \tilde{B} is considered an active attacker in this protocol, his private input and output is at his own discretion.

Finally, \tilde{B} is given a pair or triple $(m, \{m^*\}, \sigma)$ that is chosen uniformly at random from the set of all pairs or triples that are valid with respect to the public group key pk . The attacker outputs a group member index i such that

$$\text{verifyId}((pi_i, pk), m, \{m^*\}, \sigma) = \text{TRUE} .$$

Remarks: The above definition of anonymity allows the attacker to take on the role of Bob in the blind signature protocol $g\text{SignBlind}$ and execute it with different signers Alice of his choice. Obviously, if in each execution the participant Bob were honest and the attacker would only get to see the blinded signatures from respective honest Bob after each execution, then he would have no better probability of success than $1/n + c^{-k}$. Otherwise, the attacker in the definition could simply use honest Bob's in order to increase his probability of success beyond the limit of $1/n + c^{-k}$.

UNFRAMEABILITY

A polynomial-time attacker who knows the public individual keys of all group members and the private individual keys of all members but Alice cannot produce a signature for which Alice is held responsible.

Let $prek \in [\text{genPrekey}(k)]$ be a prekey, $(rk_i, pi_i) \leftarrow \text{genIKey}(prek)$ be Alice's pair of a private key and a public individual key. The polynomial-time attacker takes Alice's public individual key pi_i and then computes some public key $pk \in PK$. After obtaining a polynomial size set M_0 of messages (or pairs of messages and optional messages) and corresponding signatures from executions of $g\text{SignBlind}$ with honest Alice using her private key rk_i , the attacker has only a negligible chance of producing a signature σ for a new message (or pair of message and optional message) $(m, \{m^*\}) \notin M_0$ for which Alice is held responsible, i.e.,

$$\text{verifyId}((pi_i, pk), m, \{m^*\}, \sigma) = \text{TRUE} .$$

COALITION RESISTANCE

Any (polynomial-size) coalition of polynomial-time attackers cannot produce signatures for which not at least one of them is held responsible.

Let $prek \in [genPrekey(k)]$ be a prekey, $n \in \mathbb{N}$ be an anonymizing group size, and $\ell \in [0, n]$ be the size of an attacker coalition. The attacker coalition submits any ℓ public individual keys $pi_j \in PI$ for $j = 1 \dots \ell$. The remaining individual keys for $j = \ell + 1 \dots n$ are chosen by honest group members, and the private group key and public key are computed accordingly: $(rg, pk) \in [genGKey(prek, \{pi_1, \dots, pi_n\})]$. The attacker coalition then has only a negligible chance of producing a message m and a signature σ valid for m with respect to pk such that the attacker coalition is not held responsible for m , i.e.,

$$identify((rg, pk), m, \sigma) \notin \{pi_1, \dots, pi_\ell\} \quad \text{and}$$

$$\text{for all } j \in [1, \ell]: verifyId((pi_j, pk), m, \{m^*\}, \sigma) = FALSE \quad . \quad \blacklozenge$$

Remarks: In practice, the generation of private group keys and public keys can be done by a trusted group center or by a multi-party computation of the group members. In order to prevent framing, group members need to sign their public individual keys when they submit them. In case the group center identifies a group member later on, it can prove so only by providing the individual key of the identified member, the signature for this individual key, the public key of the group, the message and the signature that has lead to identification.

The above definition could be generalized from 1-out-of- n group blind signature schemes to (t -out-of- n) group blind signature schemes. These schemes assure that at least t -out-of- n members of a group have co-operated in order to produce a signature. In this case, signing is a $(t+1)$ -party operation. More work on combinations of threshold and anonymity requirements was done by Cramer, Damgård and Schoenmakers [79].

6.4 Cryptographic Mechanism

We motivate the following construction by revisiting an interactive proof-of-knowledge protocol by Berry Schoenmakers that allows to prove knowledge of one-out-of- n witnesses without revealing which [210]. Consider (\mathbb{Z}_p, G_q) a discrete log setting, g a generator of G_q . The candidate is an n tuple (h_1, h_2, \dots, h_n) of values in G_q and the corresponding witnesses are $x_i = \log_g h_i$. Schoenmakers' protocol allows a prover who knows at least one of the witnesses x_i to prove exactly that, but not which x_i , the prover knows. See Figure 6–1 on p.130. It has been proven by Schoenmakers et al in [210,79] that this establishes a witness indistinguishable proof of knowledge of x_i satisfying $h_i = g^{x_i} \bmod p$ for some $i \in [1, n]$. The key design idea is as follows: After the prover has sent his a_j components in step (2), he must provide matching components r_j for all j , i.e., components that satisfy the verifier's condition in step (6). Since only the sum of all d_j 's is fixed by the random challenge c (step (6)), he may choose the d_j 's for all but one j before he gets to know the challenge c in step (3). By choosing d_j in advance, it is possible to form a_j in such a way (step (2)) that the prover need not know x_j in order to find a matching r_j (step (5)).

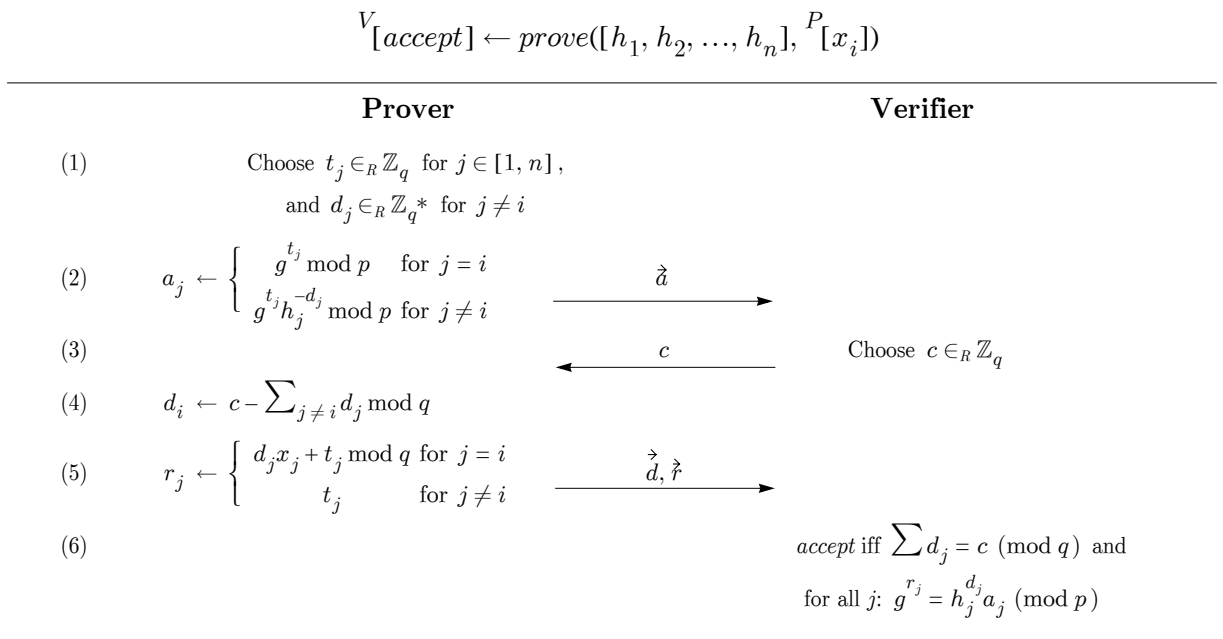


FIGURE 6-1 Proving Knowledge of one-out-of- n Witnesses

Chen and Pedersen [77] have proposed a group signature protocol based on Schoenmakers' protocol above. They require signers to sign each message twice with two different private signing keys. one is kept private and guarantees unforgeability, the other is disclosed to the group center such that the group center can later identify the originator of any given group signature. The following mechanism turns the group signature protocol of Chen and Pedersen into a blind one. The techniques are similar to those used for the Blind Chaum-Pedersen(ℓ) Signature Mechanism 4.8 on p.55.

Mechanism 6.2 Blind Chen-Pedersen Group Signature(ℓ, n) Mechanism (Blind CPGS(ℓ, n))

The parameter ℓ determines the number of generators with respect to which witnesses represent messages, and the parameter n determines the size of the group to be managed.

Generate Prekey $(p, q, g, g_1, \dots, g_\ell, \text{hash}) \leftarrow \text{genPrekey}(k)$

Pick a discrete log setting \mathbb{Z}_p, G_q uniformly at random from dls_k (Definition 3.17 (Discrete Log Framework and Settings)). Then pick $\ell + 1$ generators g, g_1, \dots, g_ℓ of G_q (Observation 3.18 (Generators in Discrete Log Settings)). Furthermore, $\text{hash} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ is a hash function that returns elements of \mathbb{Z}_q^* .

DOMAINS

- The domains of private keys and public individual keys are $RK_{prek} = \mathbb{Z}_q^2$ and $PI_{prek} = G_q \times \mathbb{Z}_q$, respectively.
- The domains of private group keys and public (group) keys are $RG_{prek} = \mathbb{Z}_q^n$, and $PK_{prek} = G_q^n$, respectively.
- The message and signature domains are

$$M_{prek} = G_q \text{ and } \Sigma_{prek} = G_q^2 \times G_q^{2n} \times G_q^{2n} \times \mathbb{Z}_q^n \times \mathbb{Z}_q^{2n} ,$$

i.e., 5-tuples whose first component is in G_q^2 and the other 4 components are n -dimensional vectors with components in G_q^2 , G_q^2 , \mathbb{Z}_q and \mathbb{Z}_q^2 , respectively. For example, we denote a message m and a signature as $\sigma = (z, \vec{a}, \vec{b}, \vec{d}, \vec{r})$.

- The blinder domains are $\Omega = \mathbb{Z}_q^*$, and the witness domains are $W = \mathbb{Z}_q^\ell \setminus \{(0, \dots, 0)\}$, each equal to those of the Blind Chaum-Pedersen(ℓ) Signature Mechanism 4.8 on p.55. Likewise, the making functions are as defined in equation (4.10) on p.56, i.e.,

$$make(w) = \prod_{i=1}^{\ell} g_i^{w_i} \pmod{p} \in M .$$

- The witness equivalence relations are defined as in equation (4.11) on p.56, i.e.,

$$(v \equiv w) \Leftrightarrow (\forall \lambda, \mu \in [1, \ell] : v_\lambda w_\mu = v_\mu w_\lambda \pmod{q}) .$$

OPERATIONS

Generate Individual Keys $(rk_i, pi) \leftarrow genIKey(p, q, g, hash)$

Each group member $i \in [1, n]$ generates a private individual key $rk_i = (x_{i1}, x_{i2}) \in \mathbb{Z}_q^2$ uniformly at random. Only someone who has both private key components, can later produce a valid group signature. A group member Alice keeps the first component x_{i1} private such that no other group member, nor the group center nor any outsider can fabricate group signatures for which Alice will be held responsible. The second component x_{i2} will be released to the group center, so that the group center can later identify the group member for any given group signature. Each group member computes its public individual key $pi_i = (pi_{i1}, pi_{i2}) = (g^{x_{i1}} \pmod{p}, x_{i2})$ and sends it to the group center to get registered.

Generate Group Keys

$$(rg, pk) \leftarrow genGKey((p, q, g, hash), (pi_1, \dots, pi_n)) .$$

The private group key is the n -tuple $rg = (pi_{12}, \dots, pi_{n2}) = (x_{12}, \dots, x_{n2})$. The public (group) key is the n -tuple $pk = (pk_1, \dots, pk_n)$, where

$$pk_i = (pk_{i1}, pk_{i2}) = (pi_{i1}, g^{x_{i2}} \pmod{p}) = (g^{x_{i1}} \pmod{p}, g^{x_{i2}} \pmod{p}) \text{ for } i = 1 \dots n .$$

The group center publishes the public group key and the index $i = 1 \dots n$ of each group member. When group member Bob receives the public group key and its group index i , Bob verifies if the public group key component pk_i matches the individual public key that Bob has submitted to the group center in the first place. Otherwise, the group key generation must be repeated. In practice, the group center will also certify the published group key and the indices of all group members. These are standardized techniques and will not be considered in the following.

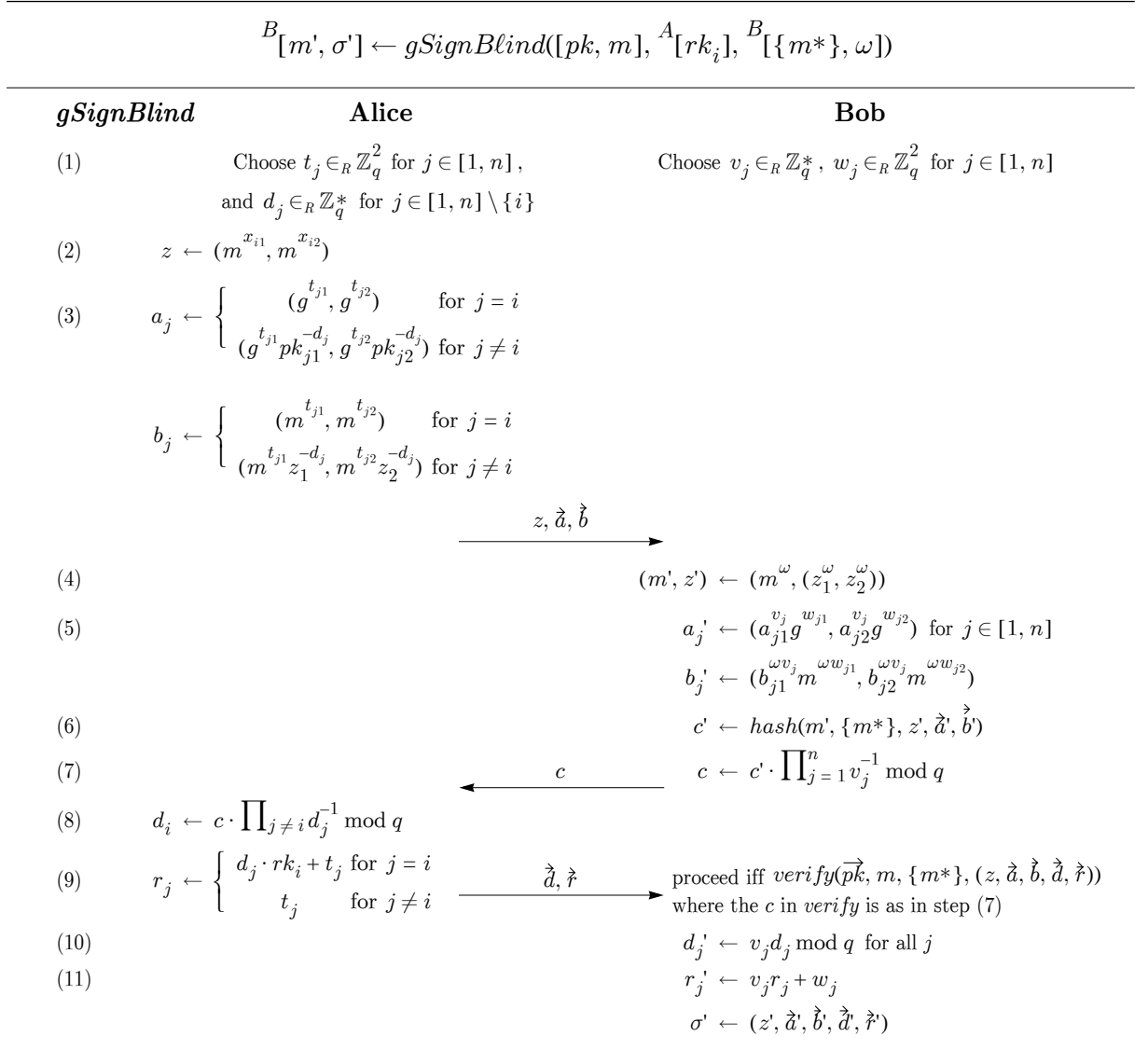
gSignBlind


FIGURE 6-2 Producing a Chen-Pedersen Group Signature

Alice takes as input her private key $rk_i = (x_{i1}, x_{i2})$. Basically, Alice executes two Chen-Pedersen signing protocols in parallel, one using x_{i1} as her private signing key, the other using x_{i2} . The values $z_1, (a_{j1}, b_{j2})$ corresponding to the former, whereas $z_2, (a_{j2}, b_{j2})$ corresponding to the latter. Observe that both member protocols take the same challenge c from Bob in step (7), so the values $d_j, (r_{j1}, r_{j2})$ are shared by both sub-protocols. Bob makes internal random choices in step (1) in order to blind Alice's messages z, \vec{a}, \vec{b} obtained in step (3). He then takes the hash value of the blinded results (m', z', a', b') (step (6)), and prepares his challenge in step (7). After step (9) Bob checks if Alice has produced a valid signature for m with respect to pk (see predicate $verify$ below), where the challenge c in the verification equation is the one computed in step (7). In step (10) and

step (11), Bob finally blinds Alice's responses d_j in order to compute the missing signature component \vec{r} .

If the optional input message m^* is used, we call this protocol the *extended signing protocol* (cf. Mechanism 4.8 on p.55) of the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130.

Verify

Bob verifies Alice's signature $\sigma = (z, \vec{a}, \vec{b}, \vec{d}, \vec{r})$ for message m as follows:

$$\begin{aligned} \text{Verify}(pk, m, \{m^*\}, \sigma) &\stackrel{\text{Def}}{=} (c = \prod_{j=1}^n d_j \bmod q) \wedge (\forall j \in [1, n] : g^{r_{j1}} = pk_{j1}^{d_j} a_{j1} \wedge \\ &g^{r_{j2}} = pk_{j2}^{d_j} a_{j2} \wedge m^{r_{j1}} = z_1^{d_j} b_{j1} \wedge m^{r_{j2}} = z_2^{d_j} b_{j2}) \\ &\text{where } c = \text{hash}(m, \{m^*\}, z, \vec{a}, \vec{b}) . \end{aligned} \quad (6.1)$$

Identify

$$pi_i \leftarrow \text{identify}(\vec{rg}, pk, m, \sigma)$$

identify

Group Center

(1) Check for which $i \in [1, n]$ the following equation holds

$$m^{x_{i2}} \stackrel{?}{=} z_2 \pmod{p} \quad (6.2)$$

(2) If such an index i is found return pi_i , otherwise return \perp .

FIGURE 6-3 Identifying the Signer

A group center on input a private group key $\vec{rg} = (x_{12}, \dots, x_{n2})$, corresponding public key pk , a message $m \in M$ and a signature σ acceptable for m identifies the originator of σ (step (1)) and outputs a public individual key $pi \in PI \cup \{\perp\}$, if a match is found (step (2)).

Verify Identity

$$\text{verifyId}(pi_i, pk, m, \{m^*\}, \sigma) = (m^{x_{i2}} = z_2 \pmod{p}) \wedge \text{verify}(pk, m, \{m^*\}, \sigma) . \quad (6.3)$$

Verifies the predicate (6.2) on p.133. ◆

Security Suggestion 6.1 Blind CPGS(ℓ, n) Mechanism

For all $\ell \in \mathbb{N} \setminus \{1\}$, $n \in \mathbb{N}$ the Blind CPGS(ℓ, n) Signature Mechanism 6.2 on p.130 is a blind static group signature scheme according to Definition 6.1 on p.126 under the following assumptions:

- (i) One-timeness and restrictiveness hold if they hold for the Blind Chaum-Pedersen(ℓ) Signature Mechanism 6.2 on p.130 (Security Suggestion 4.6 on p.57).
- (ii) Computational signer anonymity holds under some heuristic arguments made by Chen and Pedersen in [77] Section 5.
- (iii) Blindness, unframeability and coalition resistance hold under the SDL Assumption 3.20 on p.29. ◆

Security Considerations

Chen and Pedersen have suggested a special case of this scheme, which does not lead to a blind, but to an ordinary group signature scheme (see section 4 in [77]). They suggest the recipient always chooses $w_j = 0$ and $v_j = 1$, for all $j \in [1, n]$, and they split the challenge c into summands d_j instead of into factors.² In their special case, the recipient's part reduces to computing the hash value c' , which equals c because the product in step (7) collapses to 1.

EFFECTIVENESS AND GROUP EFFECTIVENESS

We have to check that Bob's output message and signature satisfy the verification predicate. It is immediate from step (7), step (8), and step (10) that the first equation holds:

$$c' = c \prod_{j=1}^n v_j = \prod_{j=1}^n d_j v_j = \prod_{j=1}^n d_j' \pmod{q}$$

Next we look at the four remaining equations. Since protocol *gSignBlind* is essentially two parallel executions of Schoenmakers' (Figure 6–1 on p.130), we only check the first equation and the third equation because the second and fourth are verified similarly. It is helpful to check these equations separately for $j = i$ and for $j \neq i$:

$$\begin{aligned} \text{For } j = i: \quad pk_{j1}^{d_j'} a_{j1}' &= (g^{x_{i1}})^{d_j'} a_{j1}^{v_j} g^{w_{j1}} && \text{def. of } pk_{j1}, \text{ step (5)} \\ &= (g^{x_{i1}})^{v_j d_j'} (g^{t_{j1}})^{v_j} g^{w_{j1}} && \text{step (3), step (10)} \\ &= g^{v_j(d_j x_{j1} + t_{j1}) + w_{j1}} \\ &= g^{v_j r_{j1} + w_{j1}} && \text{step (9)} \\ &= g^{r_{j1}'} \pmod{p} . && \text{step (11)} \end{aligned}$$

$$\begin{aligned} \text{For } j \neq i: \quad pk_{j1}^{d_j'} a_{j1}' &= pk_{j1}^{d_j'} a_{j1}^{v_j} g^{w_{j1}} && \text{def. of } pk_{j1}, \text{ step (5)} \\ &= pk_{j1}^{v_j d_j'} (g^{t_{j1}} pk_{j1}^{-d_j'})^{v_j} g^{w_{j1}} && \text{step (3), step (10)} \\ &= g^{v_j t_{j1} + w_{j1}} \\ &= g^{v_j r_{j1} + w_{j1}} && \text{step (9)} \\ &= g^{r_{j1}'} \pmod{p} . && \text{step (11)} \end{aligned}$$

2) They require that the choices d_j for $j \neq i$ and the challenge c be taken from \mathbb{Z}_q^* , so they in fact use the same domains as in *gSignBlind*.

$$\begin{aligned}
\text{For } j = i : \quad z_1^{d_j'} b_{j1}' &= (z_1^\omega)^{d_j'} b_{j1}^{\omega v_j} m^{\omega w_{j1}} && \text{step (4), step (5)} \\
&= ((m^{x_1}^\omega)^{v_j d_j} (m_1^{t_{j1}})^{\omega v_j} m_1^{\omega w_{j1}}) && \text{step (10), step (2)} \\
&= m^{\omega(v_j(d_j x_{j1} + t_{j1}) + w_{j1})} \\
&= m^{\omega(v_j r_{j1} + w_{j1})} && \text{step (9)} \\
&= m^{r_{j1}'} \pmod{p} . && \text{step (4), step (11)}
\end{aligned}$$

$$\begin{aligned}
\text{For } j \neq i : \quad z_1^{d_j'} b_{j1}' &= (z_1^\omega)^{d_j'} b_{j1}^{\omega v_j} m^{\omega w_{j1}} && \text{step (4), step (5)} \\
&= (z_1^\omega)^{v_j d_j} (m_1^{t_{j1}} z_1^{-d_j})^{\omega v_j} m^{\omega w_{j1}} && \text{step (10), step (3)} \\
&= m^{\omega(v_j t_{j1} + w_{j1})} \\
&= m^{\omega(v_j r_{j1} + w_{j1})} && \text{step (9)} \\
&= m^{r_{j1}'} \pmod{p} . && \text{step (4), step (11)}
\end{aligned}$$

ONE-TIMENESS AND RESTRICTIVENESS

We first consider the signer's messages indexed $j \neq i$, where i is the member index of the signer in his group. Were the challenges d_j given to the signer after he has computed the values a_{j1}, b_{j1} after step (3) of *gSignBlind*, then the signer had only a negligible chance of computing the values a_{j1}, b_{j1}, r_{j1} such that the verifying condition $pk_{j1}^{d_j} a_{j1} = g^{r_{j1}} \pmod{p}$ is satisfied. Otherwise he had broken the validity of the CEG(1) Mechanism 4.4 on p.43. Thus Schoenmakers has argued that for each $j \neq i$, the signer needs to choose the value d_j before he sends the values a_j, b_j to the verifier.

Once the signer has sent all values $z, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n, b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_n$ to the verifier, he is therefore committed to the corresponding values $d_1, \dots, d_{i-1}, d_{i+1}, \dots, d_n$ as well. When the verifier sends the challenge $c \in \mathbb{Z}_q^*$ after step (7), the signer is then also committed to the value $d_i = c \prod_{j=1}^n d_j^{-1}$. Since the signer knows the private individual key rk_i , he is not committed to d_i any sooner than after receiving the challenge c from the verifier. Since the signer has no information about c by the time he is committed to the values $d_1, \dots, d_{i-1}, d_{i+1}, \dots, d_n$, the value d_i is determined by the verifier uniformly at random (by means of the random choices v_j).

Next we consider the signer's messages indexed $j = i$. We have seen above, that the signer is forced to provide exactly two signatures of the Blind Chaum-Pedersen(ℓ) Signature Mechanism 4.8 on p.55 for the same challenge d_i , one of them valid for m with respect to the public key pk_{i1} , the other valid for m with respect to the public key pk_{i2} .

Thus, if for any $\ell \in \mathbb{N} \setminus \{1\}$, $n \in \mathbb{N}$, the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 were not restrictive or not one-time, then the respective attack yields a CPGS(ℓ, n) signature $(\vec{z}, \vec{a}, \vec{b}, \vec{d}, \vec{r})$ for some message m , and thus—again by counting only messages indexed i —two signatures of the Blind Chen-Pedersen(ℓ) Signature Mechanism 4.8 on p.55 for message m .

BLINDNESS

We show that for each output $(m', \vec{z}', \vec{a}', \vec{b}', \vec{c}', \vec{d}', \vec{r}')$ of Bob valid with respect to \vec{pk} and each valid view $view_{\tilde{A}} = (\rho, m, \vec{z}, \vec{a}, \vec{b}, c, \vec{d}, \vec{r})$ of a computationally unlimited attacking signer \tilde{A} on Bob obtained by a successful execution of $gSignBlind$ on common input \vec{pk}, m , there is exactly one choice ω, v_j, w_j for Bob such that he returns this output on the given input and view³. Throughout this proof all steps refer to protocol $gSignBlind$ (Figure 6–2 on p.132).

≤ 1 : There is at most one such choice because the assignment of m' in step (4) uniquely determines ω , and steps (10) and (11) uniquely determine values \vec{v}, \vec{w} , respectively. Namely, $\omega = \log_m m'$, $v_j = d_j' / d_j$ for all $j \in [1, n]$, and $w_j = r_j' - v_j r_j$.

≥ 1 : We are left to show that the one choice ω, \vec{v}, \vec{w} determined above matches the given components not yet taken into account above, namely $\vec{z}, \vec{a}, \vec{b}, c$ and $\vec{z}', \vec{a}', \vec{b}'$: Schoenmakers [210] and Chen, Pedersen [77] have argued by using the usual Fiat-Shamir heuristics that if the public group key pk , a message m and a signature $z = (z_1, z_2)$ are given (to the verifier up front), then the remaining messages of the signer in $gSignBlind$ constitute a proof of knowledge of some private individual key $rk_i = (x_{i1}, x_{i2})$ over the function:

$$f((x_{i1}, x_{i2})) = (g^{x_{i1}}, g^{x_{i2}}, m^{x_{i1}}, m^{x_{i2}}) = (pk_{i1}, pk_{i2}, z_1, z_2) \text{ for some } i \in [1, n] . \quad (6.4)$$

While the verifier receives this proof of knowledge from the signer, the verifier itself provides (by computing the output signature $z' = (z_1', z_2')$) a proof of knowledge of some private key (x_{i1}', x_{i2}') such that

$$f((x_{i1}', x_{i2}')) = (g^{x_{i1}'}, g^{x_{i2}'}, m^{x_{i1}'}, m^{x_{i2}'}) = (pk_{i1}, pk_{i2}, z_1', z_2') . \quad (6.5)$$

Because Bob himself knows none of the private keys and he has oracle access only to group member Alice, under the SDL Assumption he can prove knowledge of at most Alice's private key. Thus $(x_{i1}', x_{i2}') = (x_{i1}, x_{i2})$. Then follows for z' :

$$(z_1', z_2') = (m^{x_{i1}'}, m^{x_{i2}'}) = (m^{\omega x_{i1}}, m^{\omega x_{i2}}) = (z_1^\omega, z_2^\omega) . \quad (6.6)$$

This matches the way how z' is computed in step (4). We further need to show that the one choice ω, \vec{v}, \vec{w} determined above also leads to expressions for a_j', b_j' and c' that match step (5) and step (6), respectively. We begin with a_{j1}' , which turns out as follows:

$$\begin{aligned} a_{j1}' &= g_1^{r_{j1}'} pk_{j1}^{-d_j'} && \sigma' \text{ valid for } m' \text{ wrt. } pk_i \\ &= g^{v_j r_{j1} + w_{j1}} pk_{j1}^{-v_j d_j} && \text{step (11), step (10)} \end{aligned}$$

3) The component ρ of $view_{\tilde{V}}$ denotes an internal choice of the attacker \tilde{A} . It of course needs not be taken from the domain of internal choices of the honest signer in $gSignBlind$.

$$\begin{aligned}
&= (g^{r_{j1}} pk_{j1}^{-d_j} v_j^{w_{j1}}) g^{w_{j1}} && \text{re-ordering} \\
&= a_{j1}^{v_j} g^{w_{j1}} \pmod{p} . && \text{Bob's verify in step (9)} \tag{6.7}
\end{aligned}$$

Analogous rewritings show that the following equations modulo p hold as well:

$$a_{j2}' = a_{j1}^{v_j} g^{w_{j1}} \quad \text{and} \quad b_{j1}' = b_{j1}^{\omega v_j} m^{\omega w_{j1}} \quad \text{and} \quad b_{j2}' = b_{j2}^{\omega v_j} m^{\omega w_{j3}} .$$

Since only valid views $view_{\lambda}$ on Bob are considered, we know about c that it satisfies the verification equation of Bob's check after step (9), in particular $c = \prod_{j=1}^n d_j$. Otherwise, Bob would abort the protocol, and thus $view_{\lambda}$ were invalid. This assertion together with Bob's steps (7) and (10) implies that c' can be expressed as follows:

$$c' = c \prod_{j=1}^n v_j = \prod_{j=1}^n d_j v_j = \prod_{j=1}^n d_j' \pmod{q} .$$

Finally, step (6) assures that $c' = hash(m', z', \vec{a}', \vec{b}')$. Hence, Bob's output $(\vec{z}', \vec{a}', \vec{b}', \vec{d}', \vec{r}')$ is valid with respect to \vec{pk} .

SIGNER ANONYMITY

Signer anonymity is a security property that refers to the signer of protocol $gSignBlind$. Formally, this is clear from the Definition 6.1 on p.126 (computational signer anonymity), because the verifier Bob is assumed to be an attacker. The signer in protocol $gSignBlind^{(CPGS)}$ is similar to the signer in another group signature protocol, which we will call CP , presented by Chen and Pedersen in [77], Section 5, Fig.2. In fact, the signer of protocol $gSignBlind^{(CPGS)}$ computes two signatures of the signer in $gSignBlind^{(CP)}$ in parallel with different signing keys x_1, x_2 for the same message m , while using the same challenge c for both signatures.

Without going into details it is clear that if an attacker \mathcal{A} on $gSignBlind^{(CPGS)}$ could be successful, then it would also be successful against $gSignBlind^{(CP)}$. We just need to take only every second signature into account (of those that are produced on request of the attacker and of the two signatures that are given to the attacker at the end).

COMPUTATIONAL UNFRAMEABILITY

Chen and Pedersen have argued that under the SDL-Assumption 3.20 on p.29 their scheme gives computational unframeability. Consider a collusion of some group members and a group center (who has access to all public individual keys). This collusion tries to frame an honest group member i with private key rk_i of their group. Assume they can come up with a pair (m, σ) such that protocol *identify* on input the respective private group key, public key and (m, σ) returns index i . This contradicts the SDL-Assumption 3.20 on p.29 because the attacker collusion would find a representation of the honest victim's public individual key component $pi_{i,1}$, which is assumed to be chosen uniformly at random by the honest victim.

COALITION RESISTANCE

Under the SDL-Assumption, the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 also gives coalition resistance. Polynomial-time attacking group members can produce a valid signature with non-negligible probability only if they can prove knowledge of at least one private key. So a later identification by the group center will always succeed. \square

6.4.1 Efficiency Comparison

Finally we compare the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 (A) to the proposal (B) of Lysyanskaya and Ramzan [149]. Since the computation and communication cost of (B) is constant in the size of the group, we are interested in the trade-off group size below which (A) is more efficient than (B). In both cases, the by far most expensive computations are modular exponentiations.

(A) and (B) use modular exponentiations modulo a prime p of size $k = 1024$ bit, where k is the security parameter. All exponents of (A) are of size about $k_0 = 160$ bit (see the remark following Assumption 3.20 on p.29). The exponents of (B) fall into two categories; those of size $k_0 = 160$ bit and those of full size k . We count the modular exponentiations to k -bit exponents as 6 modular exponentiations to k_0 -bit exponents. This is a conservative estimate because given exponents of average Hamming weight the computation time of a modular exponentiation increases linearly in the length of the exponent.

As before, the parameter n denotes the size of the group. (B) uses an additional security parameter ℓ , which has not received much discussion, neither by Camenisch and Stadler [51] nor by Lysyanskaya and Ramzan [149]. It appears that ℓ should increase with k , but a deeper analysis is outstanding. As an example, Camenisch and Stadler have instantiated $k = 600$ and $\ell = 64$. Since proposal (B) uses k for the length of an RSA module, a choice of $k = 1024$ as suggested in the beginning is recommended. It is probably also recommended to use a value of $\ell = 100$.

The following Table 6-1 on p.138 approximates the computational and communication effort of signer and recipient in the signing protocol and the computational effort of verifying a signature. The group size in the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 is denoted n as before.

		Modular Exponentiations		Number of bits sent	
		(A)	(B)	(A)	(B)
$g\text{SignBlind}$	signer	$8n - 2$	$14\ell + 12$	$(4n + 2)K + 3nk$	$(4\ell + 2)K$
	recipient	$16n + 3$	$40\ell + 14$	k	$2k$
$verify$	—	$8n$	14ℓ	—	—

TABLE 6-1 Computational and Communication Effort

The bit lengths of public keys and signatures are approximated in the following Table 6-2 on p.138:

Length [bit]	Public Key		Signature	
	(A)	(B)	(A)	(B)
	$2nK$	$3K + 16$	$(4n + 2)K + 3nk$	$(4\ell + 2)K$

TABLE 6-2 Bit Lengths of Public Keys and Signatures

A rough comparison shows:

6.4 CRYPTOGRAPHIC MECHANISM

- (i) In terms of performance, the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 (A) is more efficient than (B) for group sizes up to about $n = 2\ell$, which is about 200. As the security parameter k needs to be increased over time, the trade-off group size increases as well.
- (ii) In terms of communication complexity and signature size, the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 (A) is more efficient than (B) for group sizes up to about $n = \ell$, which is about 100.
- (iii) In terms of the public key size, the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 (A) is superior only for the trivial group size of 1.

At the bottom line it appears safe to say that today the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130 (A) is not less efficient in most respects than Lysyanskaya and Ramzan's proposal (B) for group sizes 100 and 200 members depending on which criteria are given priority.

7

Group Credential Schemes

In this section, a modular refinement of all credential schemes of Section 5 on p.59 is provided. Our goal is to achieve privacy for issuers of credentials against verifiers as sketched in Section 5.1.2 on p.66. In many cases, this also enhances the privacy of recipients. A credential that is silent about its issuer reveals less information about its holder than one that discloses its issuer. On the other hand, issuers should not be allowed to stay anonymous while issuing wrong credentials or to illegitimate holders. In case a credential is disputed, its issuer should be identifiable. These requirements are pretty similar to those of group signature schemes, and so it is natural to interrelate the concepts of credentials (Section 5 on p.59) and group signatures (Section 6 on p.123). The idea of building an e-cash scheme around a group oriented blind signature scheme has also been proposed by Lysyanskaya and Ramzan [149] and by Traoré [223]. The group signature scheme underlying the proposal of Traoré is more efficient than that of Camenisch and Stadler [51] in particular for signing, but Traoré uses more ad-hoc complexity theoretic assumptions. We introduce the more general concept of *group credentials*.

7.1 Definition of Group Credential Schemes

A formal definition of group credential schemes can be obtained by refining credential schemes (Definition 5.1 on p.75) in a similar fashion as blind signature schemes are refined into blind group signature schemes in Definition 6.1 on p.126. The following definition does not present the requirements of group credential schemes in full detail because all of them appear either in one of the credential definitions 5.7, 5.9, 5.11, or are straight forward adaptations of requirements of the group signature definition (Definition 6.1 on p.126).

Definition 7.1 Static Group Credential Scheme (Sketch)

A static group credential scheme with group size n , security parameter k , credential types $T = \{t_1, \dots, t_m\}$ and recognition characteristic ($0 \leq FAR, FRR \leq 1$) is a credential scheme with the same credential types T and recognition characteristic (FAR, FRR) and the following refinements.

DOMAINS:

In addition to the domain families set out in Definition 5.1 on p.75, a group credential scheme has the following 2 additional domain families PI , RG whose respective members are:

- A *public individual key* domain PI_{prek} and
- a *private group key* domain RG_{prek} .

The private keys of a static group credential scheme are those generated and held by individual group members, while the public keys are those by which anyone can verify all signatures of all signers of the respective group. Public keys will be referred to more specifically as *public group keys*.

OPERATIONS:

In addition to the operations set out in Definition 4.7 on p.53, a static group credential scheme has a set $\{genIKey, genGKey, identify, verifyId\}$ of 4 additional operations. To enhance readability, we call the issuing operation of a group credential scheme $gIssue$. The key generating operation $genKey$ is refined into the following two operations:

Generate Individual Keys $(rk_{t,i}, pi_{t,i}) \leftarrow genIKey(prek, t)$

A probabilistic operation that takes a prekey $prek$ and returns an individual key pair $(rk_{t,i}, pi_{t,i})$ of type $t \in T$. $rk_{t,i}$ and $pi_{t,i}$ are called the respective *private* and *public individual key* of group member indexed i .

Generate Group Keys $(rg_t, pk_t) \leftarrow genGKey(prek, (pi_{t,1}, \dots, pi_{t,n}))$

A probabilistic operation that takes a prekey and an n -tuple of public individual keys $pi_{t,1}, \dots, pi_{t,n} \in PI$ of type t . It outputs a group key pair $(rg_t, pk_t) = RG \times PK$ of type t . rg_t and pk_t are called a *private* and *public group key*.

The operation $genKey$ of a static group credential scheme is then defined as follows:

$$(rk, pk) \leftarrow genKey(prek), \text{ where } rk = (rk_1, \dots, rk_n) \text{ and}$$

$$(rg, pk) \leftarrow genGKey(prek, (pi_1, \dots, pi_n)) \text{ and for } i = 1 \dots n: (rk_i, pi_i) \leftarrow genIKey(prek) .$$

Furthermore there are two additional operations:

Identify $pi_t \leftarrow identify((rg_t, pk_t), \psi, \chi)$

A deterministic operation that takes a private and a public group key $(rg_t, pk_t) \in RG \times PK$ of type t , a pseudonym $\psi \in \Psi$ and a credential $\chi \in C$. The result is a public individual key $pi_t \in PI \cup \{\perp\}$ of type t or it is undefined, denoted \perp .

Verify Identity $accept \leftarrow verifyId((pi_t, pk_t), \psi, \chi)$

A deterministic operation that takes a public individual and a public group key $(pi_t, pk_t) \in PI \times PK$ of type t , a pseudonym $\psi \in \Psi$ and a credential $\chi \in C$ and returns a Boolean result. If the result is *TRUE* then we say that the group member with public individual key pi is held *responsible* for group credential χ (on pseudonym ψ).

In addition to the credential requirements given in Definitions 5.7, 5.9, and 5.11, these operations satisfy the additional requirements of group effectiveness, signer anonymity, unframeability and coalition resistance. They are analogous to those in Definition 6.1 on p.126 if messages are replaced by pseudonyms and signatures by credentials. ♦

7.2 New Cryptographic Mechanisms

Constructions for all four categories of group credential schemes can be obtained in a modular fashion. We describe a coin group credential mechanism, which will be used in Section 8 on p.145.

7.2.1 Coin Group Credential Scheme

We can turn the CC Mechanism 5.10 on p.111 into a coin static group credential(n) mechanism (CGC(n) Mechanism) if we replace in step (5) of protocol *issue* (Figure 5–14 on p.113) the member protocol *signBlind* of the Chaum-Pedersen(ℓ) Signature Mechanism 4.8 on p.55 by the protocol *gSignBlind* of the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130.

Since *gSignBlind* is one-time, restrictive and blind just as *signBlind*, the proofs of effectiveness, type unforgeability, transfer prevention, and issue-wise fail-safe unlinkability carry over from the CC Mechanism to the CGC(n) Mechanism.

7.3 Conjunctions of Personal and Coin Credentials with Static Group Credentials

The Group Coin Credential Mechanism constructed above uses the same families of source and target pseudonym domains as the respective Personal Credential Mechanism 5.8 on p.92 and the Coin Credential Mechanism 5.10 on p.111. Hence, we can form analogous conjunctions of credentials as described in Section 5.7 on p.121, where a static group coin credential may replace a coin credential. Note that this includes conjunctions of a personal credential and a static group coin credential, which will be used in Section 8.4 on p.151 and 8.5 on p.159.

7 GROUP CREDENTIAL SCHEMES

8

Mobile Patient Assistants

*“Laws are no substitute for engineering ...
It’s no longer good enough to install security patches in response to attacks ...
Today’s systems must anticipate future attacks.”
— Bruce Schneier [207]*

An interesting field of application for group credentials is where holders and issuers have a common privacy interest against verifiers. We consider the example of compulsory health insurers, whose insurance fees are related to the income of their policy holders rather than their state of health. Hence at least up to a certain limit, no direct link between the fees paid by a policy holder and the cost incurred need to be observable by a compulsory health insurer. As an example for this application we consider the German health care system. This is joint work with Matthias Schunter [26,27,28].

8.1 Introduction

In most western democracies the increasing diversification of healthcare providers and competition between them provides pressure for lean administration, charging and clearing. As more and more computers become networked, we can expect the integration of almost all healthcare related processes between physicians, hospitals, online medical databases, pharmacies, health insurers, and so on. However, the legitimate privacy interests of patients and physicians are likely to be ignored if today’s paper-based procedures are naively simulated by networked computer systems. We show electronic solutions for charging and clearing expenses which allow insurers to enforce an annual limit on total expenditure while maintaining the privacy of patients and physicians.

Former and current paper-based procedures have relied heavily on identifiable patient data in order to ensure data integrity and accuracy. Although in many countries data protection laws apply to such information, the most effective protection against privacy breaches still is the cost of collecting, transferring, storing and analyzing large volumes of paper files. Patients usually accept that they will have little if any privacy against healthcare professionals directly involved in their treatment, but wish to protect their privacy against outsiders like healthcare insurers, attorneys, employers and landlords.

The transition from a paper-based healthcare environment to one based on networked computers is unlikely to respect the distinction between medical and non-medical parties. Firstly, the interaction of clinical and non-clinical participants in the treatment process will be rationalized by computers in just the same way as the interaction between the clinical participants. Secondly, both integration processes are driven by the same medium: the Internet. Finally, some non-medical parties have legitimate and illegitimate interests in identifiable patient data. So the easier it is to access, transfer, store and analyze large amounts of data, the more we must protect identifiable patient data; and if these protection measures are to be effective, they will have to be built into the technical infrastructure.

On the one hand, the new technologies support new potential abuses of medical data for surveillance, control and marketing purposes [194,242]. On the other hand they also facilitate a new level of privacy, by building systems that avoid the storage of large amounts of identifiable patient data in central repositories. One particular solution is to equip patients with mobile patient assistants, e.g., palm pilots, personal digital assistants or other mobile user devices [201]. Mobile patient assistants can help to develop telemedicine in a way that respects the privacy of patients and physicians alike. They can also help patients to avoid prescription errors [219,220] and to better manage chronic diseases like diabetes [232] or cancer.

Healthcare providers are about to invest millions in new communication and computing infrastructures [242,105]. The market for chip-cards for both patients and physicians is expected to grow rapidly in the future. However, such investments will only pay if the fielded technologies meet legal requirements such as data protection, and are acceptable by all participants, i.e., patients, physicians and health insurers. The G7 and some national initiatives [45] have stimulated such technologies, the topic has been suggested for further research to the Commission of the European Communities [17,18] and specific solutions for the US market are under development [148]. To derive an acceptable solution we will state the duties and goals of each participant and then answer the key question:

Who needs what data in order to fulfill their duties and meet their goals?

We will then discuss suitable technical measures for implementing the charging and clearing process in a privacy oriented way. Our analysis indicates that chip-cards are too restricted to really protect the privacy of patients and physicians. Shorter precursors of this paper appeared in [26,27].

8.2 Contractual Framework

We will now describe the participants and transaction flows of the German healthcare system [4,46,44,127] with respect to billing and payment for medical services. We will then deduce the security interests of the various participants.

The German healthcare system¹ consists of five supply sectors [4,46]. *Medical outpatient treatment* includes registered physicians and specialists, e.g., dentists, who have their own independent practices. *Paramedical outpatient treatment* includes members of professions allied to medicine like physio-therapists and speech therapists. *inpatient treatment* consists of all hospitals for acute cases

1) A German-English and English-German glossary about the German healthcare system is found in [4].

and special hospitals. *Public health services* are provided by state and local public health departments and by the laboratories. The *pharmaceutical supply* is provided by pharmacies.

Health insurers are the clearing houses of the healthcare system. In practice they delegate the clearing tasks to several client-specific organizations (*actual clearing houses*). There are compulsory and private health insurers, each with about half the market in healthcare payments. Roughly speaking, contributions to the former are income related, whereas those to the latter are risk-related. There is a level of income below which compulsory health insurance is mandatory. The privacy interests of patients (and physicians) inherently conflict with the screening interests of private health insurers to such an extent that we suggest our solution for compulsory health insurers only.

Throughout this paper we distinguish four kinds of *healthcare providers* (Figure 8–1) and sketch

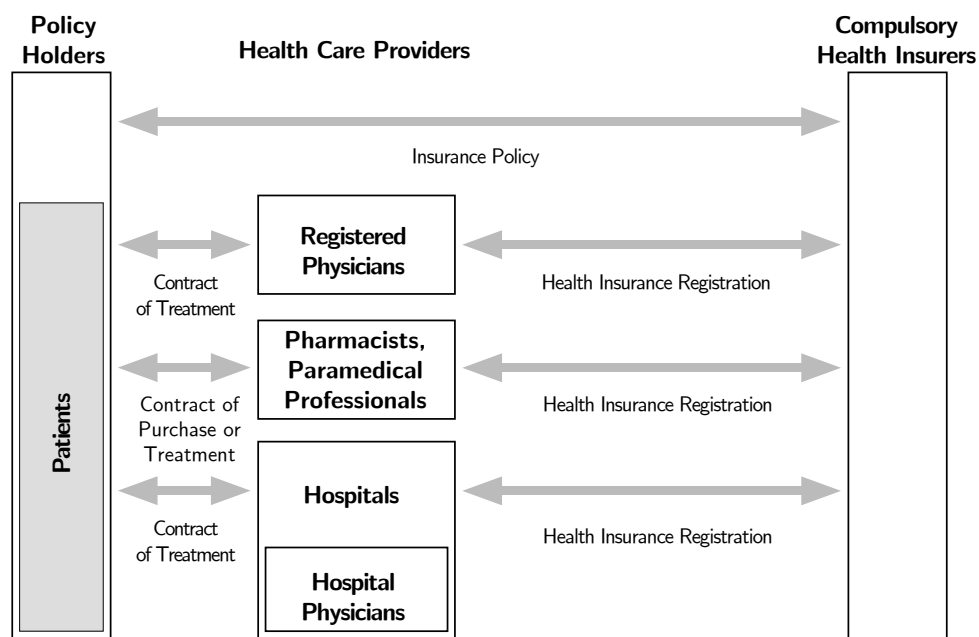


FIGURE 8–1 Contractual Framework

their business relationships.

- 1) *Registered physicians* are outpatient general physicians or specialists who are registered by compulsory health insurers. They may provide medical treatment, write letters of referral and issue prescriptions for medication or paramedical treatment. They do not claim directly to the health insurers, but to the local associations of registered physicians: *Kassenärztliche Vereinigungen (KV)*. They also serve as clearing houses: Each KV gets a lump sum from the compulsory health insurers and pays the invoices of registered physicians. The registration is done by a joint registration committee of the health insurances and the KVs. KVs are explained in more detail in the following Section 8.3.

- 2) *Pharmacies and paramedical providers* serve patients more or less according to prescriptions. They may neither issue prescriptions nor write letters of referral. Their actual clearing houses are the health insurers.
- 3) *Hospital physicians* are physicians who are registered by compulsory health insurers and who are employed by a hospital. Like registered physicians, they may provide medical treatment, write letters of referral and issue prescriptions for medication or paramedical treatment. They do not claim directly from the health insurers either, but from their hospitals which serve as their first-line clearing houses.
- 4) *Specialists* are all those outpatient and hospital physicians who provide medical treatment according to letters of referral. Specialists running their own practices are not depicted in Figure 8-1. In the following they can be treated in the same way as hospital physicians except that they invoice to their KV rather than to a hospital.

8.3 Paper-based Charging and Clearing

We will now describe in more detail how the costs of medical treatment and supplies are claimed in the German system. Interactions between two participants consist of “real” actions and of “paper” actions. For example, a physician treats a patient, sends an invoice for the treatment and is finally reimbursed. We regard the first and last of these actions as “real” and the second as a paper action. Our focus is on electronic transactions substituting the paper actions, particularly those containing identifying patient data (Figure 8-2).

Consider a typical treatment process: A patient requests treatment from her GP by presenting a valid health insurance card (“Krankenversichertenkarte”), which contains the data necessary for charging medical treatment. The GP may provide some treatment on his own and in addition:

- 1) prescribe some medication, and
- 2) refer the patient to a specialist or hospital.

During the process of healthcare, these steps can be iterated with various medical professionals taking responsibility for the patient and delegating it further. In each of the three cases, the GP produces a medical record that contains accounting data and possibly diagnostic, therapeutic or prognostic information about the patient. Usually, the patient passes a relevant excerpt of this record to the next healthcare provider, who then continues the process of treatment. Each healthcare provider copies the relevant part of the patient’s record and forwards it to the appropriate clearing house with his invoice.

8.3.1 Analysis

Since 1992 the compulsory health insurers have equipped their policy holders with personal health insurance cards (“Krankenversichertenkarten”). These are memory chip cards containing the same administrative data that had previously been held on a paper-based health insurance record card (“Krankenschein”). If a patient requests a medical service from a healthcare provider, she has to identify herself by her health insurance card. This is basically a way to enforce the identification of patients—the primary requirement of health insurers. The patients’ privacy requirements, however, have simply been ignored.

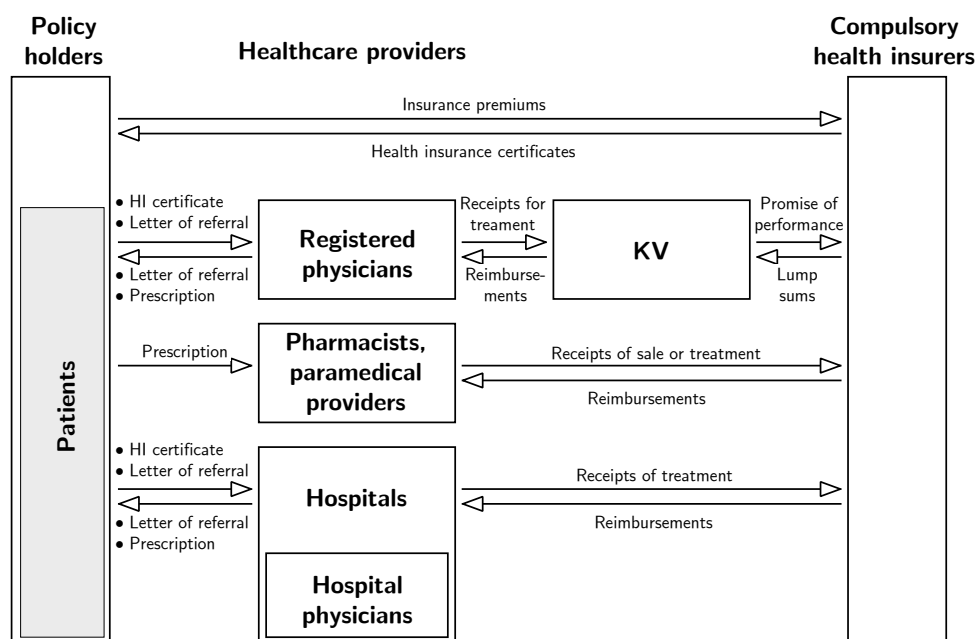


FIGURE 8-2 Flow of Billing Related Information

The paper-based refund system implements a kind of post-paid system. Health insurance cards can be regarded as a special kind of credit cards. Showing them authorizes a patient to receive medical treatment. The patient's health insurer pays lump sums to the various clearing houses which reimburse expenses that are properly supported with receipts endorsed by the KVs and hospitals.

Alternatively, healthcare providers could also claim directly to the patients, as most private insurers do (such as private health or car insurers). In this case, policy holders get to know the detailed cost of their treatment and could, if suitably motivated, help to control costs by asking their healthcare providers for less expensive services and checking all invoices carefully. They might also occasionally decide not to use their health insurer but to pay some items themselves.

Usually, receipts for everyday's commercial transactions do not contain much personal information about the payer; but receipts in healthcare are different. In paper-based systems the invoices of the healthcare providers and thus the receipts received by the clearing houses contain a tremendous amount of highly personal and sensitive information about both patients and physicians. For example, every participant involved gets to know each patient's prescription, and with private insurers, all documents referring to a patient are linked from the treating physician through to the insurer. The mere existence of such information tempts people to use it for secondary purposes.

8.3.2 Participants and Their Specific Security Requirements

We now ask which participants really need to have which information in order to fulfill their tasks. We will analyze the services to be provided by each participant and then consider additional constraints posed by the specific confidentiality and privacy needs of the various participants.

AVAILABILITY AND INTEGRITY REQUIREMENTS

Physician: Each patient shall receive no more treatment and medication than is prescribed. In particular, each prescribed treatment shall be provided at most the specified number of times. (This is all the more important if drugs are dispensed by vending machines [221].)

Policy holder: If she presents a valid enrollment in a suitable health insurance plan and, if applicable, a letter of referral or prescription to a healthcare provider of her choice, then the provider shall indeed offer the requested service or perform the treatment prescribed.

Physicians, Pharmacies, Paramedical providers: Any of their expenses should be reimbursed by the health insurers if the healthcare provider is registered and if the claims are properly supported by a respective proof of treatment, letter of referral, etc.

Health insurers: Only registered physicians should be able to issue prescriptions. Each policy holder should be able to use prescriptions at most once or according to a therapy plan, respectively. Each health insurer should reimburse expenses only once and only if they have been spent for its own policy holders. Health insurers should be able to cap the total reimbursement per year (“Deckelungsprinzip”).

Healthcare providers usually need some administrative details about their patients, but much less personal data needs to be communicated between providers. Patients need non-repudiable prescriptions from their physicians. Providers need to verify the prescriptions before giving treatment or medication. They also need to obtain receipts. In paper-based practice, the medical prescription serves for both purposes. But although insurers need to know their policy holders, and physicians usually know their patients well, their views need not be linkable. A patient pseudonym is usually quite sufficient for charging. We will see a few exceptions when we discuss particular implementations of the charging and clearing process in Section 8.4 on p.151.

CONFIDENTIALITY AND PRIVACY REQUIREMENTS

Physician and Patient: Medical treatment requires a relationship based on trust between patient and physician. The *privacy* of their relationship should be protected comprehensively against third parties’ interests; diagnoses and therapies should be strictly confidential. This specific rule should override, for example, any obligations to escrow cryptographic keys [161].

Physician: Health insurers should not in general be able to monitor physicians’ prescription and treatment habits. The cost control interest of health insurers does not justify more detailed control than can be exercised using statistical mechanisms, such as spot-checking a small sample of treatments.

Policy holder: The policy holder’s right to ask a healthcare provider of her choice for second opinions implies that different healthcare providers should not monitor policy holders by exchanging views on them. In certain situations, patients might seek a stronger form of privacy, which we will call *untraceability*: even where a patient is referred from one provider to another, they cannot collude to link their views afterwards and find out which records belong to the same patient.

Obviously, the above requirements can be met by legal regulations, but technical means are more effective—especially if they can be enforced by the policy holders themselves. Therefore, we introduce pseudonyms for policy holders as well as for physicians and we propose to employ them consistently in all interactions [194,195].

8.3.3 Exceptions

Exceptional cases must be supported because the physician in charge of a patient is ultimately liable for how the patient is treated and may require flexibility. For example, one should be able to deal with emergencies in which the patient is not able to confirm or consent to anything, and cases where a physician decides not to tell the whole story to his patient.

8.4 Implementation Options

We now discuss the most important implementation options of the charging and clearing process of a healthcare system. We basically consider compulsory health insurers who provide broad coverage of medical expenses. So it is realistic to assume that individuals are enrolled in at most one insurer's health insurance plan. The case of enrollment in multiple insurance plans will be considered below in Section 8.4.8 on p.158.

Healthcare insurance is based on a sensible separation of duty. Physicians decide about therapies and prescriptions, whereas insurers finance the corresponding treatments and medications. Since the physicians' decisions may incur significant cost, physicians need to be legitimated by health insurers before they may charge for medical treatment. Similarly, pharmacies and paramedical providers are legitimated by insurers. Both types of provider legitimation are called *registration*.

There are three types of legitimation contents, namely a patient's proof of enrollment in a health-insurance plan, a referral to a specialist, and a prescription, which we will describe in detail in Section 8.4.1 on p.151. In principle, these patient legitimations should be non-repudiable and unforgeable. These requirements suggest that we implement all legitimations by cryptographic primitives based on digital signatures [94]. (For additional privacy-protecting requirements see Section 8.4.2 on p.153.) An implication of using digital-signature-based mechanisms is that healthcare providers need not trust in the health insurers, because, if need be, providers can prove all their claims to an arbiter such as a court.

Given that patients are mobile, we assume a system architecture in which each policy holder is equipped with a mobile patient assistant, i.e., a mobile user device [188,143] able to manage patient legitimations and in particular to produce and verify digital signatures. We further assume that each healthcare provider has equipment to communicate with mobile patient assistants. Good candidates for mobile patient assistants are palm pilots and palmtop computers (Section 5.1.3 on p.67). They are more appropriate than smartcards because they have a user interface and power supply, more computing power and more storage capacity.

8.4.1 Enrollments, Referrals and Prescriptions

The patient legitimations are characterized as follows (see Figure 8–3):

Patients' *enrollments* (*E*) prove that they are policy holders enrolled in a health insurance plan. Different health insurance plans may vary in coverage, deductions, fees, and so on; and while individ-

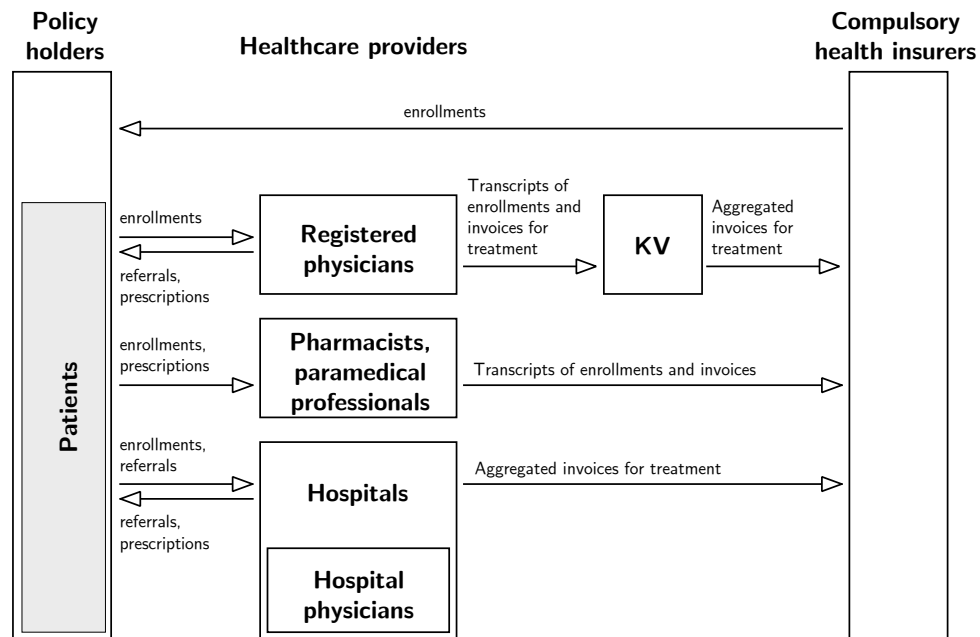


FIGURE 8-3 Use of Patient Legitimations.

uals are not supposed to give away or sell their enrollments as such, they should have enough power of delegation to ask a friend or relative to go and get a prescribed medication for them. Furthermore, the patients' privacy should be maintained as much as possible even if the health insurer and the physician collude.² To provide the strongest protection, patients will use pseudonyms in order to enroll at their insurers.

Patient's *prescriptions* (P) prove that a certain medication or service has been prescribed. In order to get medications or services, patients need to use an enrollment and a prescription, or send a delegate.

Patient's *referrals* (R) prove that patients have been referred to a hospital or a specialist. In order to receive specialist treatment, patients need to use an enrollment and a prescription.

Since enrollments, referrals and prescriptions can be issued by different parties, it appears most natural to implement them separately by enrollment certificates, prescription certificates and referral certificates, respectively; mobile patient assistants can store and manage any combination of them as required for each healthcare encounter. This raises issues of pseudonym management but can also allow patients some freedom to negotiate their purchase with the pharmacy or other provider. For example, they might negotiate for drugs similar to those prescribed³.

2) Note that we do not assume that the patient as a person is anonymous but rather that the clearing and billing is anonymous, and that no publicly readable data identifies a patient. We do not want to stop physicians from storing their patients' personal data in local files.

3) What "similar" means has to be specified between health insurers, pharmacies and paramedical providers beforehand. Any medication or service not similar to the prescribed one is not reimbursed.

8.4.2 Anonymity Options

Our goal is to implement all legitimations of health care providers and patients and the charging of medical expenses in a way that maintains the privacy of patients and of physicians against the insurer. Individuals act as policy holders to the insurer and as patients to the healthcare providers. In neither of these capacities they decide directly about the cost to be incurred, so they may use different unlinkable pseudonyms with each health insurer and healthcare provider. In addition, patients who are referred from one physician to another can be untraceable by these physicians. Similarly, patients who receive and fulfill prescriptions can be untraceable by the providers involved. We suggest implementations for additional untraceability in Section 8.4.4 on p.153.

Physicians are different in that they decide for which patient they charge what amount of reimbursement from the respective health insurer. So we want physicians to remain anonymous as long as they act honestly, but be identifiable at least after the fact if they fake bills or charge for overly expensive treatment. This affects the way how physicians are registered, charge for medical treatment, issue prescriptions and issue referrals. An effective way of achieving recoverable anonymity is to form groups of physicians and to enable each member of a group to sign in behalf of the group without revealing their individual identity. In case of a dispute later on, the group should be able to re-identify any members by their signatures. In principle, physicians' groups can be managed by the KVs since the health insurers usually pay lump sums to the KVs, which then redistribute the payments to individual physicians.

In order to provide sufficient anonymity for physicians and patients alike, the physicians' groups should contain a mix of different specialists and general practitioners and not only a few of each. For example, one could group the practitioners of a geographical region, or the physicians of one or more hospitals, etc. The appropriate size of these groups depends on the level of anonymity required, the efficiency of the underlying mechanisms, and additional constraints of the KVs. In the following, we assume a group size of up to 100 physicians.

Keeping pharmacies and other providers anonymous is of little use because they do not reveal much more information about patients than what health insurers learn from prescriptions anyway.

8.4.3 Cryptographic Extensions

Policy holders will use source pseudonyms with their health insurers and are then free to use other pseudonyms with each physician, pharmacy and paramedical provider they visit. Signature schemes and credential schemes can be used to implement enrollments, prescriptions and referrals. A particular extension of coin credentials is helpful in the following applications.

A special case of coin credentials are *check credentials*. At issuing time, a maximum "value" is fixed, and the holder is free to use it later for any amount not exceeding the initial value. Efficient implementations have been given by Brands in [34,35], but will not be considered further in this work.

8.4.4 Implementing the Legitimations

Physicians' registrations are usually obtained on a long-term basis, i.e., physicians can exercise them as often as they like (limited only perhaps by their age). So the following implementation appears natural: Physicians choose their personal signing keys and have them certified by the health insurers directly or by an association of health insurers that takes care of certifying healthcare providers.

Finally the KVs admit physicians who provide certified signing keys to certain groups. Furthermore, physicians authorize their invoices for medical treatment and the prescriptions to their patients by a group signature, which the KV can use later to endorse its next aggregated invoices to the respective health insurers.

Policy holders' enrollments can be implemented by personal credentials. This appears most natural because enrollments shall be exercisable arbitrarily often, but shall not be transferable between owners. The pure implementation in Section 5.4 on p.90 achieves this, but needs extensions in order to let friends or relatives of patients exercise enrollments for them (see the first example of Section 2.5 on p.11). Alternatively, enrollments can be implemented by coin credentials. When enrolling in a health insurance plan, a policy holder could obtain a batch of coin credentials and could later give away one of them together with a prescription to a friend or relative, which they can use through their own mobile patient assistant. A convenient shortcut in either case is to configure one's mobile assistant to support only the intended shows and then lend one's own mobile patient assistant to the friend or relative. If health insurers want to distinguish different types of enrollments (e.g., medical treatment, dental treatment, etc.), then for each type the most appropriate implementation can be chosen.

Referrals and prescriptions need to maintain the issuing physicians' privacy in a way so that the receiving healthcare provider can endorse its invoice properly without revealing the identity of the issuing physician. Hence referrals and prescriptions could be implemented by group signatures or by coin group credentials. Both implementations reveal to the health insurer only the group to which the issuing physician belongs, but not his identity. In case of a dispute, his identity can be recovered by help of the group. The groups for writing referrals and issuing prescriptions can be the same as those for registration because they serve the same purpose, i.e., maintaining the physicians' privacy against insurers.

Coin group credentials differ from group signatures by an additional level of privacy for the policy holders. If referrals or prescriptions are implemented by coin group credentials, then even if all health care providers collaborate, they could not determine whose referrals or prescriptions are used where. Otherwise, the health care providers could link all their views on the patients. If the health insurer could get hold of data collected at the providers, then it could correlate patients' addresses with the locations of their hospitals, physicians and other providers. In order to protect the patient-physician relationship, this additional level of privacy is recommended for referrals. It seems however less important for prescriptions, because pharmacies and paramedical providers are not anonymous to health insurers anyway.

8.4.5 Charging for Medical Treatment

If physicians want to charge for treatment, they first need to see an enrollment from their patients, so as to learn which insurance plan to bill to. In the current paper-based system there is little control over what treatment a physician actually provides and what he later charges for. In particular, the compulsory insured patients do not learn what expenses their physicians claim. This can be implemented as follows: Physicians sign their invoices with a group signature, endorse them with a proof that the patient is properly enrolled and send them to their KV. The KV archives the invoices by member physician, aggregates them into, e.g., monthly or quarterly invoices and sends them to the

respective health insurers on a regular basis. When the health insurer reimburses a lump sum, the KV refunds the respective member physicians.

If health insurers want more control, they could ask their policy holders to consent to each medical treatment as a condition of reimbursement. Here are two implementations, the first more patient privacy protecting than the second:

- Patients provide a proof of enrollment that is customized to the treatment they consent to. The customized proof can be obtained by implementing enrollments by check credentials that can be filled in by the patients before use.
- Patients consent by signing their treatment with an ordinary digital signature. The physicians keep the signed reports with the patient's record. If a health insurer detects that some budget is exceeded, it may ask for patients' signed consent. In addition, the KVs may spot-check physicians, i.e., ask for the consent of randomly selected patients.

Exception handling is necessary if patients cannot or will not consent to treatments.

8.4.6 Charging for Paramedical or Specialist Treatment and Medications

When visiting a pharmacy or a paramedical provider a patient must show an enrollment and a prescription. Similarly, when visiting a specialist, a patient must show an enrollment and a referral. In each case, both legitimations will be accepted only if they are shown for the same target pseudonym. (See Section 5.7 on p.121 and Section 7.3 on p.143 for implementations). The idea of matching pseudonyms is sketched in Figure 8–4. An enrollment E is issued to a policy holder for a source pseudonym Δ . Later a physician issues a referral or prescription R/P to this policy holder for source pseudonym ∇ . Finally, the patient shows both legitimations for the same target pseudonym \circ .

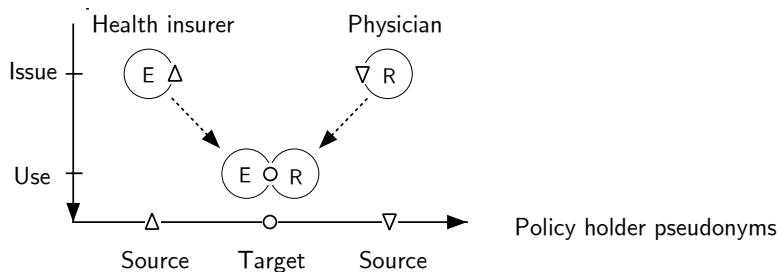


FIGURE 8–4 Showing an Enrollment and a Referral for one Target Pseudonym

In principle, enrollments can be implemented by personal or coin credentials, whereas prescriptions and referrals can be implemented by group signatures or coin group credentials. Double showing of referrals and prescriptions can be prevented online by checking the health insurer's database of all referrals or prescriptions ever shown. Alternatively, double showing can be prevented offline by the observer inside a mobile patient assistant. In this case, overshoot prevention relies only on the tamper resistance of observers. Since tamper resistance is not as trustworthy as cryptographic measures, double showing should still be detectable after the fact. As referrals and prescriptions need to be shown for

the same target pseudonyms as the accompanying enrollments, the policy holders' identities must be encoded into the source pseudonyms of these credentials by the insurer, who is ultimately liable.

Checking overshooting of legitimations online requires the health insurer's database to be highly available and also gives the insurer timing information about when patient legitimations are shown. We therefore want to avoid any need for online access to the health insurer during patient visits to a physician. We consider only enrollments that can be shown offline, and consider the implementation options for referrals and prescriptions orthogonally (Table 8-1 on p.156).

ENROLLMENTS IMPLEMENTED BY PERSONAL CREDENTIALS

If referrals or prescriptions are implemented by group signatures, then the content of a referral or prescription can simply be signed just as a normal message.⁴ This is efficient, but leaves referrals or prescriptions traceable by collaborating physicians and hospitals or by physicians and non-medical providers, respectively.⁵ Note that they can link any issuing and showing of these referrals and prescriptions. In most cases, this appears to be a minor issue. Double showing can easily be detected if the physicians are required to include a target pseudonym in their referrals or prescriptions. This can be checked online when the prescription is shown.

Important Advantages (+) and Disadvantages (-)		Referrals or Prescriptions			
		Group Signatures		Coin Group Credentials	
		offline	online	offline	online
Enrollments	Personal Credential	relies totally on tamper resistance	+ efficient - online - traceable ^a	+ offline + untraceable - many keys	+ untraceable - online - many keys
	Coin Credential	+ efficient - traceable - transferable ^b	less interesting ^c	+ offline + untraceable - transferable - many keys	less interesting ^c

TABLE 8-1 Combinations of Implementations and their Characteristics

- a. collaborating health care providers can trace referrals or prescriptions
- b. without non-digital measures, enrollments are easy to transfer between individuals
- c. like option to the left, but enrollments can be shown online only

If referrals or prescriptions are implemented by coin group credentials, then the only known way of enclosing different contents of referrals or prescriptions is by using a different group signing key for each issuing physician and each prescription. However, the prescriptions must be kept separate from the issuing physician's identity, because the former shall remain visible while the latter shall be kept anonymous. Therefore we suggest to form *referral groups* and *prescription groups*, i.e., one group of physicians for each possible referral or prescription content. The group signature for a referral or prescription is then to be verified with the respective group's public verifying key. Coin credentials add an extra layer of privacy for the patients, because even if the health insurer and all healthcare providers collaborate, they cannot determine which referrals or prescriptions are shown by the same patient(s).

4) We assume that signatures are randomized, i.e., that any two signatures for the same prescription are different.
 5) Powerful health insurers who may try to bribe or blackmail health care providers.

This solution may or may not be appropriate for an entire national health system. For example, on the German pharmaceutical market there are about 80,000 medications⁶ that are available only on prescription or at a pharmacy. Combinations, dosages and other treatments mean that there are at least several hundred thousand possible prescription contents. The situation is probably similar for referral contents. Maintaining a referral or prescription group for each content is probably practical for smaller subsets of contents, but it is unclear how this solution scales.

If coin group credentials are shown offline, then the observer inside the patient's mobile assistant is supposed to prevent the patient from using a referral or prescription twice. If the observer is broken, the health insurer should still be capable of detecting double showing after the fact. Hence, enrollments are implemented by bond credentials, which bear a patient's identity in their source and target pseudonyms just like coin credentials.

If double show detection of coin group credentials is done online, then the recipient simply checks with the healthcare provider if the coin group credential at hand is still unused.

ENROLLMENTS IMPLEMENTED BY COIN CREDENTIALS

If enrollments are implemented by coin credentials, then overshoot prevention of referrals or prescriptions can be enforced offline. Online solutions can be derived by replacing the role of the observer inside the patients' personal devices by an online request to the insurer. This appears to be less interesting because it requires all non-medical providers to be online and tends to reduce service availability of the system.

If referrals or prescriptions are implemented by group signatures, then double showing can be prevented by requiring physicians to include the patients' target pseudonyms into the referral or prescription content. If we assume that patients can never show two coin credentials for the same pseudonym, then they cannot double show any prescription while going undetected. This assumption holds for most existing coin credential schemes [35,73]. If referrals or prescriptions are implemented by coin group credentials, then double showing of a prescription reveals the policy holder's identity because the prescriptions must be shown for the same target pseudonym as the accompanying enrollment coin credential. Note that overshoot detection of the coin group credential scheme is not needed since it is already provided by the coin credentials implementing the enrollments. This solution has been presented in earlier versions of this work [26,27].

DELEGATING PURCHASE OF MEDICATIONS

Next we consider the case of patients who want friends or relatives to go and buy medications for them. If enrollments are implemented by personal credentials, delegation to buy medications can be done by giving away one's mobile assistant to the delegate. In order to prevent misuse of the mobile patient assistant, either the patients could block their assistants for anything but the intended purchase, or the patients could have pre-established pseudonyms with their pharmacies, so that the delegates can later reuse the established pseudonyms without being asked for a second biometric verification. The latter solution presumes more trust in the delegate because there might be other pre-established pseudonyms, which the delegate might use as well. The other way to delegate buying med-

6) Not counting homeopathic medications.

ications is by transferring a copy of the enrollment and of the prescription from the patient's mobile assistant into the delegate's mobile patient assistant. The delegate must then be able to show both for the same pseudonym without biometric verification. This extension of certificates has been introduced in the second paragraph of Section 2.5 on p.11.

If enrollments are implemented by coin credentials or coin group credentials, then a patient needs to transfer one of these coins and the prescription into the delegate's mobile patient assistant. For the coin credential, a transfer protocol is needed that moves the coin directly from the patient's mobile assistant into the delegate's. If prescriptions are implemented by group signatures, they can simply be copied.

8.4.7 Capping the Total Annual Reimbursement

The simplest method of enforcing cost control with this system is for the insurer to track the budget of each physicians' group. If any group exceeds its budget, it is asked to re-cover the submitters of the highest invoices in order to find out who caused the trouble.

Another way of enforcing, for example, a yearly cap of $\text{€}L$ is to let physicians claim their expenses in a virtual currency. At the end of the year the exchange rate of the virtual currency is calculated as the quotient of $\text{€}L$ over the sum of all claims. Each physician then gets reimbursed according to this exchange rate. This is how German compulsory health insurers operate today. Of course, shorter intervals of reimbursement are possible.

8.4.8 Discussion

If compulsory health insurers provide specialized health insurance plans so that policy holders may enroll in the plans of several insurers at the same time, then a common clearing center can be used to cap the overall cost. As well as consenting to treatment, patients must also decide which insurer to bill for a treatment. We are then concerned about whether the common clearing center needs to be online, e.g. to check prescriptions for double showing. What are the most suitable implementation options?

For enrollments, personal credentials appear to be the most suitable implementation. They cannot be transferred between individuals, yet it is easy to delegate personal credentials to a friend or relative. Besides, only one personal credential needs to be issued to each policy holder for each insurance plan. This solution is presented in more detail in Section 8.5 on p.159. An open question of an implementation by personal credentials is how to patients can be enabled to consent to their medical treatments. The main advantages of coin credentials are that health insurers keep more control over the number of enrollments for each policy holder. Extending them to check credentials enables patients to consent to their medical treatments in a privacy protecting way. Their main disadvantage is transferability between individuals (at least by lending or sharing personal devices).

For prescriptions, group signatures appear most appropriate. Their double showing can be strictly prevented regardless of whether enrollments are implemented by personal or coin credentials, and they do not make delegation any more difficult.

For referrals, offline coin group credentials are promising. They provide an additional level of untraceability, do not require the health insurer to be online every time a specialist is visited, and their double showing must probably not be strictly prevented. In such a combination, the ideal imple-

mentation of enrollments are offline personal credentials. In this case, coin group credentials could achieve full privacy and untraceability. This solution is also presented in Section 8.5 on p.159.

8.5 Implementation of the Clearing Process

The discussion of implementation options in Section 8.4.8 on p.158 has identified a practical solution that yields strong privacy for patients and physicians. The enrollments are implemented by personal credentials (Section 5.7 on p.90), referrals by coin group credentials (Section 7.1 on p.141), and prescriptions by group signatures. In order to sign their invoices, registered physicians and hospital physicians use group signatures, whereas pharmacies and paramedical providers use ordinary digital signatures. Ordinary digital signatures [122,201] and group signatures [77,70,77] have been intensely discussed and efficient implementations are proposed in the literature. Analogous to the definition of blind signatures and blind group signatures, we use the operations *genKey*, *sign*, *verify* for the former and *genIKey*, *genGKey*, *gSign*, *verify*, *identify* for the latter. In the following description of the charging and clearing process we use the following simplified syntax of protocol calls in order to focus on the essentials and keep the notation readable. Pseudonyms are denoted by small symbols as in Section 8.4.6 on p.155. Observers are not included as participants in protocol calls. Neither are the pseudonyms shared by mobile patient assistants and their respective observers, pseudonym witnesses, co-witnesses nor biometric identities because they are all handled exactly as described in Section 5 on p.59.

8.5.1 Initialization

First of all, security parameters and prekeys $prek_S$, $prek_G$, $prek_{CG}$, $prek_{PC}$ are chosen for an ordinary signature scheme (e.g., DSA [164], RSA [201]), the Blind CPGS(ℓ, n) Mechanism 6.2 on p.130, the CGC(n) Mechanism (Section 7.2 on p.143) and the PC Mechanism (Section 5.4 on p.90), respectively.

The health insurer H generates issuing keys for a personal credential scheme, usually one key for each type of health insurance plan. Policy holders and health care providers are initialized as follows (Figure 8–5).

INITIALIZATION OF POLICY HOLDERS

- (i) Initially, the health insurer equips every policy holder with a tamper resistant observer that fits into their mobile patient assistant. The policy holders personalize their observers with their biometric identities. To save cost, the health insurer could team up with other authorities that have already fielded some kinds of observers to individuals. In this case, the health insurer provides only the necessary health insurance application software, which is then co-existing with other kinds of application software within their policy holders' observers.

INITIALIZATION OF REGISTERED AND HOSPITAL PHYSICIANS

Physicians get organized in groups, managed by the KVs and hospitals, which charge for medical treatment and issue referrals. To keep things simple, we consider only one group G of physicians who charge for similar types of treatment and one group G' of physicians who issue letters of similar referrals.

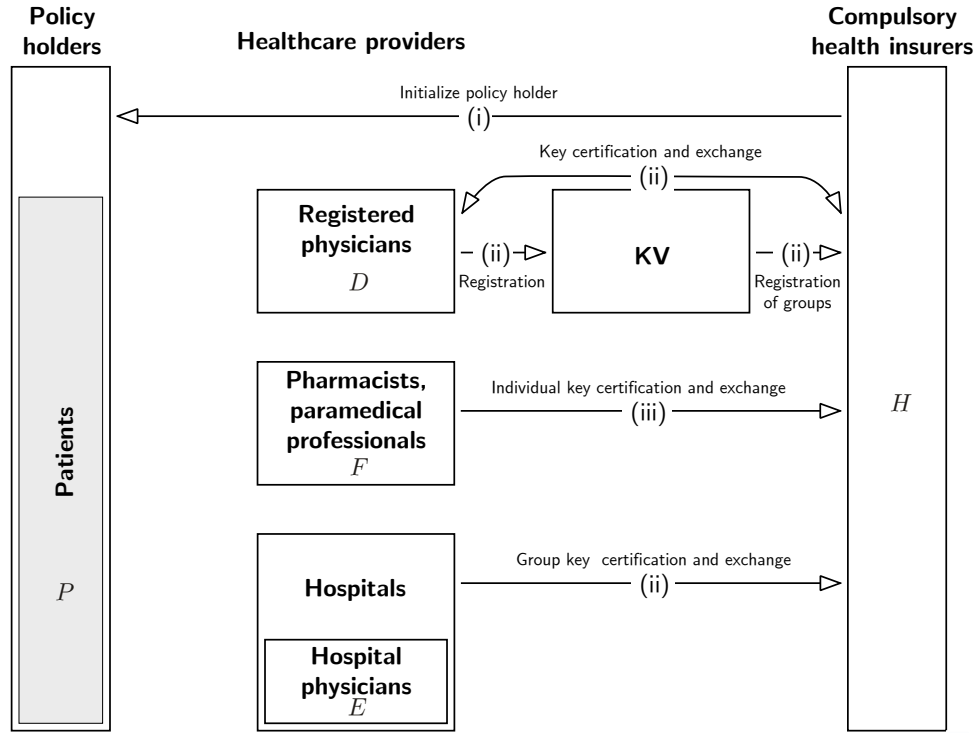


FIGURE 8-5 Initialization Phase

- (ii) Each registered physician D and hospital physician E generates an individual key pair for a group signature scheme (left hand equations of (8.1) on p.160). This will enable her to sign invoices for medical treatment. In addition, each physician who is licensed to write letters of referral generates an individual key pair of a coin group credential scheme (right equations of (8.1) on p.160). We show this only for one referral content and therefore write only a bullet for the type index in the right hand equations of (8.1) on p.160.

$$\begin{aligned} (rk_D, pi_D) &\leftarrow genIKey(prek_G) , (rk'_D, pi'_D) \leftarrow genIKey(prek_{CG}, \bullet) \\ (rk_E, pi_E) &\leftarrow genIKey(prek_G) , (rk'_E, pi'_E) \leftarrow genIKey(prek_{CG}, \bullet) . \end{aligned} \quad (8.1)$$

The KV generates group key pairs from appropriate public individual keys of the physicians they have registered, and so do the hospitals for the physicians they employ:

$$(rg_G, pk_G) \leftarrow genGKey(prek_G, pi_D, pi_E, \dots), (rg'_G, pk'_G) \leftarrow genGKey(prek_{CG}, pi'_D, pi'_E, \dots),$$

Whether these groups may contain registered physicians and hospital physicians is subject to negotiation. Finally, the health insurer registers each group by certifying its public key.

INITIALIZATION OF PHARMACIES AND PARAMEDICAL PROVIDERS

- (iii) Each pharmacy and paramedical provider F generates a key pair for an ordinary digital signature scheme and publishes the public key pk_F .

$$(rk_F, pk_F) \leftarrow genKey(prek_S) .$$

The health insurer registers each pharmacy and paramedical provider F by certifying their public keys. Alternatively, some or all health insurers could form an association that organizes the certification of pharmacies and paramedical providers in a centralized fashion.

8.5.2 Policy Holders' Views

We are going to walk through a complete charging and clearing example including one health insurer H , one of its policy holders P , a physician D , a specialist E and a pharmacy or paramedical provider F . This subsection contains all actions in which the policy holder participates, and the following subsection explains how registered physicians charge their expenses to health insurers. An overview is given in Figure 8-6. The final subsection shows how the health insurer can cap the total cost.

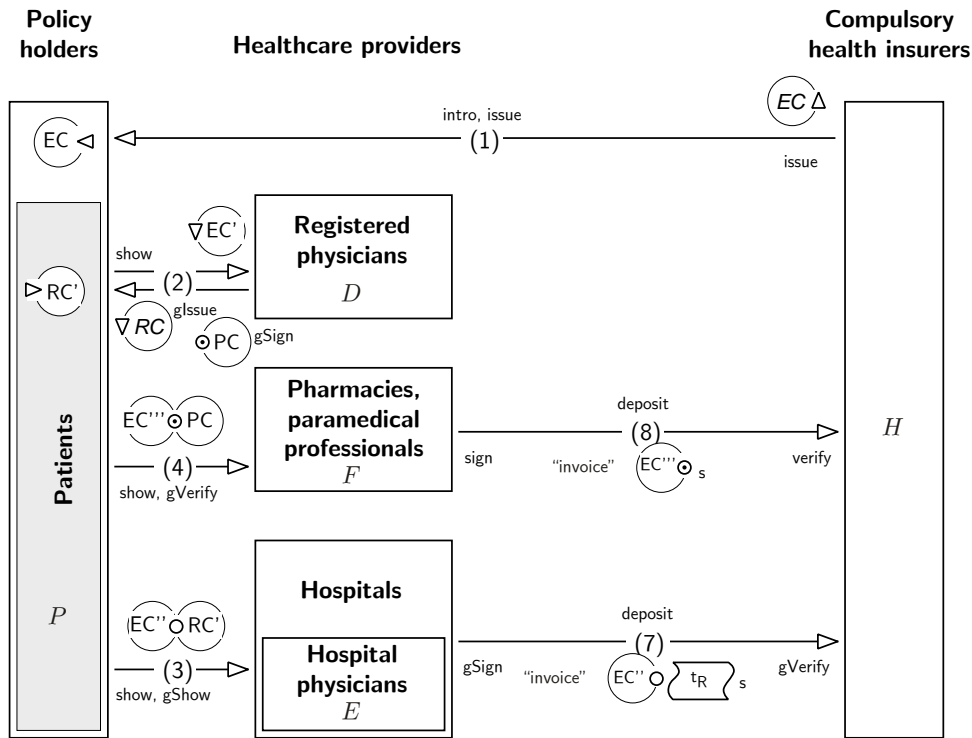


FIGURE 8-6 Charging and Clearing of Referrals and Prescriptions

GETTING AN ENROLLMENT

- (1) To get enrolled in a health insurance plan, a policy holder P introduces a pseudonym \triangleleft to the insurer H (operation *intro*) and obtains a personal credential EC for intermediate pseudonym \triangleleft in return:

$$P[\triangleleft, EC] \leftarrow issue([pk_H], {}^H[rk_H, \Delta], {}^P[\Delta]) .$$

GETTING A REFERRAL OR PRESCRIPTION

- (2) When patient P visits a registered physician D , then P first proves to be enrolled in a suitable health insurance plan by using the personal credential EC for her intermediate pseudonym \triangleleft . The physician D in turn gets to know P by his target pseudonym ∇ and matching credential EC' .

$$({}^D[\nabla, EC'], {}^P[\nabla]) \leftarrow show([pk_H], {}^P[\triangleleft, EC]) .$$

If during the course of treatment, D wants to write a letter of referral he issues a coin group credential RC' to the patient P for the target pseudonym ∇ , which is now re-used as a source pseudonym. The patient receives the coin group credential RC' for another intermediate pseudonym \triangleright . The actual referral content determines the group G' whose issuing key D must use:

$${}^P[\triangleright, RC'] \leftarrow gIssue([pk_{G'}], {}^D[rk_D \nabla], {}^P[\nabla]) .$$

If D wants to write a prescription he first asks P for a pharmacy target pseudonym \odot , and then signs the prescription content and the target pseudonym \odot using a group signature (in Figure 8-6 this signed prescription is denoted by a circle labeled by \odot):

$$PC \leftarrow gSign(pk_G, ri_D, (prescription, \odot)) .$$

If the patient visits a specialist E , then the specialist is supposed to run the same transactions while using his own keys instead of D 's.

SHOWING A REFERRAL

- (3) In order to obtain specialist treatment, patient P shows both her enrollment and referral for a joint target pseudonym \circ to a specialist E :

$$\begin{aligned} ({}^E[\circ, EC''], {}^P[\circ]) &\leftarrow show([pk_H], {}^P[\triangleleft, EC]) \\ {}^E[\circ, RC', \tau_R] &\leftarrow gShow([pk_{G'}], {}^P[\triangleright, RC']) . \end{aligned}$$

If the target pseudonym \circ has been shown before, then the patient's observer refuses to assist in using it again. If the observer's tamper resistance is defeated, the health insurer can still identify the policy holder by means of the transcript τ_R . E accepts only if both operations succeed and reveal the same target pseudonym \circ .

SHOWING A PRESCRIPTION

- (4) In order to obtain her medications, patient P shows her enrollment at the pharmacy or paramedical provider F and lets F verify her prescription:

$$\begin{aligned} ({}^E[\odot, EC'''], {}^E[\odot]) &\leftarrow show([pk_H], {}^P[\triangleleft, EC]) \\ gVerify(pk_G, PC, (prescription, \odot)) &= TRUE . \end{aligned}$$

This is accepted only if both operations succeed for the same target pseudonym \odot and this prescription has not been shown before endorsed by EC'' . The latter is checked online by contacting the health insurer's database. F obtains the signed prescription for charging purposes. The same protocol is used by paramedical providers.

8.5.3 Healthcare Providers' Views

CHARGING FOR MEDICAL TREATMENT (Figure 8-7)

- (5) The physician D claims expenses for medical treatment anonymously. D uses a group signature to sign his invoice, and the pseudonym and credential of his patient. This data and the group signature are sent to KV:

$$\sigma = gSign(pk_G, rk_D, (invoice, \nabla, EC')) .$$

The credential EC' for pseudonym ∇ tells the KV that a patient has seen one of the member physicians of group G , which is managed by the KV.

$$gVerify(pk_G, (invoice, \nabla, EC'), \sigma) = TRUE . \quad (8.2)$$

In order not to allow physicians to simply re-use transcripts, the health insurer will check if the target pseudonym ∇ has been used before. Since the enrollment credentials (EC') point at their issuing health insurers, physicians cannot re-use the same enrollment credential to bill the same medical treatment to different health insurers.

It might be even more desirable if the transcripts also carried some kind of patient's consent to the treatment to be reimbursed (cf. Section 8.4.5 on p.154).

The KV archives all signed invoices by member physician and aggregates the details into monthly or quarterly invoices to respective health insurers. These invoices itemize the medical treatments and respective expenses during the billing period, are endorsed by the corresponding anonymized invoices of member physicians' and claim the total expenses from the respective health insurers. The monthly or quarterly invoices are signed by the KV.

- (6) The KV obtains lump sum reimbursements from the health insurers and refund the claiming physicians accordingly:

The charging and clearing of medical treatment is summarized in Figure 8-7.

CHARGING FOR SPECIALIST TREATMENT (Figure 8-6)

- (7) The specialist E claims expenses in much the same way as physician D , but in addition E attaches to his invoice the transcript τ_R of the patient's referral:

$$\sigma \leftarrow gSign(pk_G, rk_E, (invoice, \odot, EC'', \tau_R)) .$$

In addition to the verification in step (6) above, the health insurer can check for double showing by looking up the target pseudonym \odot in its database. If it is found, then the previous transcript τ_R' will re-identify the cheater:

$$pi = make(extract(\{\tau_R, \tau_R'\})) .$$

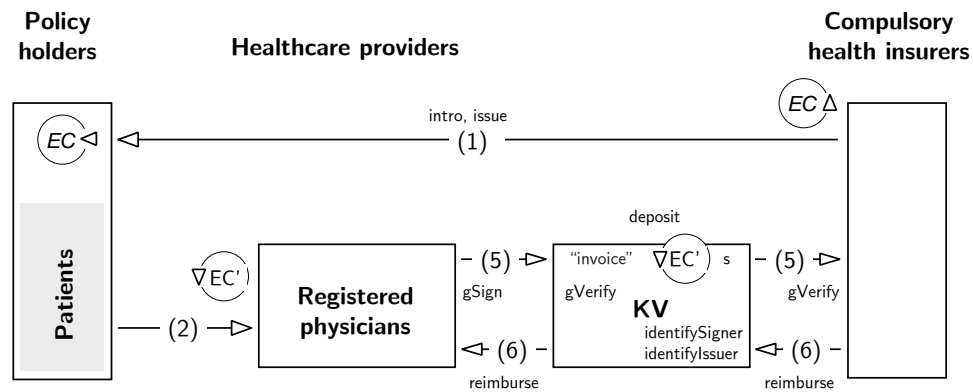


FIGURE 8-7 Charging and Clearing of Medical Treatment

During reimbursement, the hospital takes the role of the KV in step (6).

CHARGING FOR MEDICATION AND PARAMEDICAL TREATMENT

- (8) The pharmacy or paramedical provider F claims expenses in much the same way as a specialist. Only the invoice contains a group signature for the prescription and an ordinary digital signature by the pharmacy or paramedical provider and the insurer reimburses them directly rather than via a hospital as in step (7).

8.5.4 Limiting the Total Cost

The above concept of charging and clearing enables the health insurers to limit the overall cost of the system. Each health insurer can monitor the expenses for medical treatment, referrals and prescriptions given or issued by each group of physicians. If certain groups exceed their budgets, the respective KV or hospital can be asked to re-negotiate the reimbursements for their group members, or re-identify those physicians who have claimed expenses significantly above average:

In addition, the KVs could also recommend practices for subsequent spot-checking and a small percentage of policy holders can be asked to participate in cross-section studies.

In addition, the health insurers can monitor pharmacies and paramedical providers individually.

8.6 Security

A security analysis of such a system in real life is beyond the scope of this work. This section shall only summarize the evidence that the cryptographic kernel of the proposed implementation would satisfy a more rigorous proof.

8.6.1 Availability and Integrity Requirements

Physician: Each patient shall receive only the treatment and medication prescribed and exactly as often as prescribed.

Referrals and prescriptions are implemented by coin group credentials and group signatures, respectively. Double showing of referrals and prescriptions is prevented by each patient's observer or detected by the health insurer after the fact. Double showing of group signatures can also be detected because pharmacies and paramedical providers check the health insurer's pseudonym database online.

Policy holder: If she presents a valid insurance certificate, letter of referral or prescription to a health-care provider of her choice, the provider shall offer the requested product or service.

Again, effectiveness of enrollments, referrals and prescriptions follow from the respective credential schemes. Availability of medical or paramedical services is a contractual rather than a technical matter.

Pharmacies and Paramedical providers: Their bills get paid by the insurers if they are properly supported by proofs of treatment or purchase of medication.

Healthcare providers use transcripts of referral or prescriptions to prove referral; proofs that the product or service was covered by the patient's health insurance plan is embedded in the prescription or transcript of enrollment. These transcripts also reveal their issuers' group and so insurers can verify physicians' registrations.

Health insurers: Only registered physicians should be able to issue prescriptions. Each policy holder should be able to use prescriptions at most once (or according to a therapy plan). Each health insurer should reimburse expenses only once and only if they have been spent for its own policy holders. Health insurers should be able to cap the total reimbursement per year ("Deckelungsprinzip").

Health insurers accept only invoices from registered groups of physicians, pharmacies etc. because these providers have to submit their public keys during initialization. Unforgeability and overshoot prevention of enrollments, referrals and prescriptions have been discussed above.

As personal credentials can only be used with the same observer present at the time of issue, and as the observer is personalized to exactly one biometric identity, it is infeasible to use someone else's enrollments unless the tamper-resistance of observers is broken. And as the defeat of an observer only allows the abuse of the credentials issued to its holder, such a defeat does not allow widespread fraud. It is also to be expected that insurers may limit the lifetime of enrollments by issuing new keys periodically. The total reimbursement per year can be controlled by monitoring and budgeting the groups of physicians.

8.6.2 Privacy Requirements

Physician and Patient: Medical treatment requires trust between patient and physician. This relationship must be protected against third parties' interests; diagnoses, therapies and prognoses should be strictly private. This rule should override, for example, a general obligation to escrow cryptographic keys. In general, health insurers do not need to know and thus should not know which physicians their policy holders visit.

Physicians achieve their privacy by charging and prescribing anonymously in one or more groups, and patients enforce their privacy by using different pseudonyms for different health care providers. This

8 MOBILE PATIENT ASSISTANTS

combination guarantees that no participant other than the patient and the physician can link any two of their visits.

Physician: At least by default, health insurers should not be able to monitor the physicians' prescription and other treatment habits. The interest of health insurers in cost control only justifies aggregate and spot checks.

In our proposal, the health insurers can profile only groups of physicians, not individual physicians.

Policy holder: The policy holder's right to ask a for second opinions implies that different healthcare providers should not monitor policy holders by exchanging views on them.

Since a policy holder can use different pseudonyms for each visit to a provider, no two visits or purchases can be linked from the data she provides. This feature is supported by the personal user devices being indistinguishable on the network, e.g., no machine readable serial numbers must be present.

9

Conclusions and Open Questions

Four basic categories of credentials have been defined according to their characteristic holder authorizations. Credentials that can be shown only by those who have received them in the first place are called personal credentials. They can serve as electronic substitutes, e.g., for anonymous non-transferable membership cards. Credentials that are consumed when being shown are called coin credentials. They can serve as electronic substitutes for anonymous cash, coupons, electronic tickets and specialized or customized currencies. Credentials that are transferable and do not get consumed are called free credentials and those which are not transferable and get consumed are called bond credentials. Bond credentials might help to implement, e.g., ballots in a voting scheme.

A typical privacy requirement of credentials is unlinkability for holders. An efficient implementation of show-wise unlinkable personal credentials is given. It relies on tamper resistant hardware to the least possible extent, namely to prevent holders from transferring their credentials by enforcing biometric recognition. The new solution relies less on tamper resistant hardware and is more efficient than previous proposals. Efficient implementations of unlinkable coin credentials have been proposed before. The proposal by Brands is shown to be easily turned into an unlinkable bond credential scheme. The concept of issuer anonymity (group credentials) is introduced for the four basic categories of credentials. Issuer anonymity holds relative to a group and can be removed in case of disputes later on. A practical implementation of coin group credentials is given, which is more efficient than previous solutions for group sizes up to about 100 members.

As an application of several categories of privacy oriented credentials, we have considered a compulsory health insurance system, where patients and physicians seek strong privacy protection from health insurers and other, possibly illegitimate, 3rd party interests. If patients are willing to use some kind of mobile user device such as a palmtop and health insurers are willing to rely on some tamper resistant security modules implanted into the mobile patient assistants, then patients can receive medical and paramedical treatment without revealing to their health insurers what treatment they receive, nor from which physician. A complete electronic solution is described for charging and clearing medical and paramedical expenses in a way that protects the legitimate security and privacy interests of all participants and, particularly, the patient-physician relationship. For example, the original proposal of

9 CONCLUSIONS AND OPEN QUESTIONS

a health insurance card in Germany 1992 suggested the card as a medium to communicate administrative and medical data between physicians, pharmacies and other health care providers. Patients and patient advocates appreciated the increased reliability of chip-card technology, but harshly criticized the patients' lack of control over what is stored or communicated by their cards. So patients rightfully see little added value in any smart card solution. Our proposals suggest a technological leap towards mobile patient assistants, which can give patients appropriate control over both administrative and medical data.

Three interesting questions remain open for further cryptographic research: (1) Find a more general definition of restrictive blind signature schemes, which captures at least the signature schemes suggested by Chaum, Pedersen [73] and Brands [34]. (2) Construct group oriented personal credentials. (3) Construct any Boolean combination of credentials of possibly different categories.

The design of new protocols and algorithms will remain a creative process, and finding new or improved solutions needs a lot of experience. Therefore, one of the real challenges of cryptology is to provide the measure sticks, theoretical foundations and heuristics to facilitate rigorous quality control of new inventions. The golden goal, of course, is to make the definitory frameworks general enough so that not every new invention requires a modified definition but yet applicable enough so that using the theorems is more efficient than proving from scratch.



Index of Symbols

Symbol	Definition if applicable	Description
$ k _2$		length of a bitstring $k \in \{0, 1\}^*$ or binary length of a number $k \in \mathbb{N}$
$O(f(n)) = g(n)$	$\exists c > 0, n_0 \in \mathbb{N} \forall n > n_0 : f(n) \leq cg(n)$	big O notation
\mathbb{N}	$\{1, 2, 3, \dots\}$	the set of natural numbers without zero
\mathbb{N}_0	$\{0\} \cup \mathbb{N}$	the set of natural numbers including zero
\mathbb{IP}	$\{2, 3, 5, 7, 11, \dots\}$	the set of primes
\mathbb{IP}_k	$\{p \in \mathbb{IP} \mid (p _2 = k)\}$	the set of primes of binary length k bit
\mathbb{Z}	$-\mathbb{N} \cup \mathbb{N}_0$	the ring of integers
$ S $		cardinality of a set S
$[a, b]$	$\{a, \dots, b\}, \mathbf{a \leq b}$	interval of consecutive integers (see Knuth [116])
$i = a \dots b$	$\forall i \in [a, b]$	for all i running from integer a to integer b , where $a \leq b$
$\gcd(a, b)$		greatest common divisor of two integers a and b
$c = cra(a, b)$	$c = a \pmod{q_1}$ and $c = b \pmod{q_2}$	Chinese Remainder Algorithm on two arguments $a \in \mathbb{Z}_{q_1}$ and $b \in \mathbb{Z}_{q_2}$. The moduli q_1, q_2 are prime.
\mathbb{Z}_n	$(\mathbb{Z}_n, +, \times)$	the ring of residues modulo the natural number n , where $+, \times$ denote addition and multiplication modulo n .
\mathbb{Z}_n^*	$\{a \in \mathbb{Z}_n \mid \gcd(a, n) = 1\}$	the group of units of \mathbb{Z}_n , n an integer. The size of \mathbb{Z}_n^* is $\varphi(n)$ (see below)
\mathbb{Z}_p	$(\mathbb{Z}_p, +, \times)$	the finite field of residues modulo a prime p , where $+, \times$ denote addition and multiplication modulo p .

INDEX OF SYMBOLS

Symbol	Definition if applicable	Description
$ord(a)$	least $k \in \mathbb{N}$ s.t. $a^k = 1$	Order of a group element a .
$ord(G)$	least $k \in \mathbb{N}$ s.t. $\forall a \in G : a^k = 1$	Order (or exponent) of a group G .
G_q		the subgroup of \mathbb{Z}_p^* of order q , $q \in \mathbb{N}$ a divisor of $p-1$
G_q^*	$\{a \in G_q \mid ord(a) = q\}$	the subset of generators of G_q , $q \in \mathbb{N}$ a divisor of $p-1$. The size of G_q^* is $\varphi(q)$ (see below).
$\varphi(n)$	$\varphi(n) = \prod_{i=1}^k (q_i - 1)q_i^{e_i - 1}$	Euler's totient function, where the factorization of n into prime factors is $n = \prod_{i=1}^k q_i^{e_i}$.
$x \bmod n$		residue of x modulo n relative to a given representation of \mathbb{Z}_n
$x^{-1} \bmod n$	$y \in \mathbb{Z}_n^* : xy = 1 \pmod{n}$	multiplicative inverse of x modulo n (defined only for $x \in \mathbb{Z}_n^*$)
\leftarrow		evaluate the right hand side and assign the result to the left hand side
$a \equiv b$	$a_1 b_2 = a_2 b_1 \pmod{q}$	equivalence of two witnesses
$pr_1 \parallel pr_2$		general composition of protocols pr_1 and pr_2 (Section 3.2.3 on p.21)
$pr_1 ; pr_2$		sequential composition of protocols pr_1 and pr_2 (Section 3.2.3 on p.21)

B

Index of Authors

A

Adams, Carlisle M 9
Asokan, N 98
Ateniese, G 125

B

Bakker, Albert x
Barber, Barry x
Beaver, Don 75
Beaver, Donald R 16
Bellare, Mihir 31, 36
Bengio, Samy 40
Biskup, Joachim ix
Blaze, Matt 98
Blaze, Matthew 42, 58
Bleumer, Gerrit 42, 58, 61, 98
Blobel, Bernd x
Boyd, Colin 123
Brands, Stefan 33, 43, 52, 59, 65, 69, 110, 111, 115
Brassard, Gilles 34, 60
Brickell, Ernest 66

C

Camenisch, Jan 29, 125
Camenisch, Jan L 62, 66, 125
Campbell, Duncan iv
Canetti, Ran 31
Chaum, David 30, 33, 34, 40, 43, 51, 52, 55, 59, 61, 62, 63, 65, 66, 67, 69, 124
Chen, Lidong 64, 65, 124, 130
Cramer, Ronald 65, 69, 124, 129
Crépeau, Claude 34
Croft, R 124

D

Damgård, Ivan Bjerre 63, 65, 124, 129
Davey, John x
Davida, George I 61
Desmedt, Yvo 40, 124
Dwork, Cynthia 64

E

El Gamal, Taher 62
Even, Shimon 67
Evertse, Jan-Hendrik 43

F

Feige, Uriel 21, 34, 37
Ferguson, Niels 52, 65, 69
Fiat, Amos 33, 34, 37, 52, 65
Flikkenschild, Erik x
Frankel, Yair 52, 61, 66, 124
Franklin, Matthew x, 33, 52, 65, 84

G

Garfinkel, Simson iv
Gemmell, Peter 66
Goldreich, Oded 28, 31, 34, 36, 60, 65, 67
Goldwasser, Shafi 15, 28, 34, 38
Goutier, Claude 40
Govaerts, René 65
Graham, Ronald x
Grimson, Jane 3
Grimson, William 3
Guillou, Louis Claude 33, 62

INDEX OF AUTHORS

H

Halevi, Shai 31
Harris, S 124
Hasselbring, William 3
Hellman, Martin E 30
Heys, Howard M 9
Hoare, Charles Antony Richard 27
Hong, Lin 61, 8
Horster, Patrick 62

J

Jain, Anil 61, 8

K

Katsikas, Sokratis x
Kluge, Eike-Henner x
Knuth, Donald E x
Kravitz, David 66

L

Lenstra, Arjen K 30
Lotspiech, J 65
Louwerse, Kees x
Lysyanskaya, Anna 64, 65, 67, 75, 125, 138, 141

M

Maher, David x
Manna, Zohar 75
Matt, Brian J 61
Maurer, Ueli 66
McLean, John 75
Menezes, Alfred J 20
Micali, Silvio 15, 28, 34, 38
Michels, Peter 62
Miller B 61
Mu, Yi 125

N

Naccache, David 66
Naor, Moni 52, 65
Nguyen, Khanh Quoc 125
Nyberg, Kaisa 62

O

Odlyzko, Andrew M x
Ohta, Kazuo 3, 40, 42, 43, 66, 124
Okamoto, Tatsuaki 3, 40, 42, 43, 65, 66, 124

P

Pankanti, Sharath 61, 8
Patashnik, Oren x
Pedersen, Torben Pryds 40, 52, 55, 65, 66, 69, 124, 130
Petersen, Holger 62
Pfitzmann, Andreas ix
Pfitzmann, Birgit ix, 28, 30, 52, 65, 75, 84, 90
Piveteau, Jean-Marc 62, 66
Pnueli, Amir 75
Pointcheval 62
Pointcheval, David 52, 54, 58, 62

Q

Quisquater, Jean-Jacques 33, 62

R

Rackoff, Charles 15, 28, 34, 38
Radu, Christian 52, 65
Ramzan, Zulfikar 67, 125, 138, 141
Rao, Josyula R 72
Reiter, Michael K x
Rivest, Ronald ix
Rivest, Ronald L 30, 64, 65, 75
Rogaway, Philip 31
Rohatgi, Pankaj 72
Rueppel, Rainer R 62

S

Sadeghi, Ahmed-Reza 52
Sahai, Amit 64, 75
Salomaa, Arto 60
Schneier, Bruce 60
Schnorr, Claus 29, 33, 55, 62, 65
Schoenmakers, Berry 124, 129
Schunter, Matthias ix, 75, 145
Shamir, Adi 21, 33, 34, 37
Shoup, Victor 98
Simmons, Gustavus J. 60
Stadler, Markus 29, 125
Stadler, Markus A 62, 66
Stern, Jacques 52, 54, 58, 62
Stinson, Douglas R 60
Strauss, Martin 42, 58, 98

T

Tompa, Martin 34
Traoré, Jacques 125, 141
Tsiounis, Yiannis 66, 102
Tsudik, Gene 125

V

van Antwerpen, Hans 65
van de Graaf, Jeroen 43
van Heijst, Eugene 30, 66, 124
van Oorschot, Paul C 60
van Solms, Sebastiaan 66
Vandevalle, Joos 65
Vanstone, Scott A 60
Varadharajan, Vijay 125
Vaudenay, Serge x
Verheul, Eric R 30

W

Waidner, Michael x, 75, 84, 98
Wolf, Stefan 64, 75
Woll, Heather 34

Y

Yacobi, Yakov 67
Yao, Andrew Chi-Chih 38
Yung, Moti 33, 52, 65, 66, 84, 102

INDEX OF AUTHORS



References

The references are divided into paper articles (Section C.1 on p.1) and electronic resources (Section C.2 on p.13). Cross links are provided.

C.1 Bibliography

- [1] Anderson R (ed): Information Hiding; LNCS 1174, Springer-Verlag, Berlin 1996.
- [2] Anderson R (ed): Personal Medical Information Security, Engineering, and Ethics; Springer-Verlag, Berlin 1997.
- [3] van Antwerpen CJ: Electronic Cash; Centre for Mathematics and Computer Science (CWI), Amsterdam, October 11, 1990.
- [4] Arnold M, Brauer HP, Deneke V, Fiedler E: The Medical Profession in the Federal Republic of Germany; Deutscher Ärzte-Verlag, Köln-Lövenich 1982.
- [5] Asokan N., Shoup V, Waidner M: Optimistic Fair Exchange of Digital Signatures; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 591-606.
- [6] Ataniese G, Tsudik G: Group Signatures à la Carte; ACM-SIAM Symposium on Discrete Logarithms, SODA 1999; ACM Press, 848-849;
<http://www.isi.edu/~gts/pubs.html>
- [7] Ataniese G, Tsudik G: A Coalition-Resistant Group Signature; ISI Technical Report (<http://www.isi.edu/~gts/pubs.html>)
- [8] Bach E, Shallit J: Algorithmic Number Theory, Vol 1; MIT Press, Cambridge MA 1996.
- [9] Beaver DR: Security, Fault Tolerance, and Communication Complexity in Distributed Systems; Ph. D. Thesis, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts, May 1990.
- [10] Beaver DR: Foundations of Secure Interactive Computing; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 377-391.
- [11] Beaver DR: Formal Definitions for Secure Distributed Protocols; DIMACS Workshop in Distributed Computing and Cryptography, Princeton 1989; DIMACS Series in Discrete Mathe-

REFERENCES

- matics and Theoretical Computer Science Vol. 2; AMS, Providence 1991; ACM, Baltimore 1991; 47-64.
- [12] Bellare M: Practice-Oriented Provable Security; in Ivan Damgård (Ed.) Lectures on Data Security, LNCS 1561, Springer-Verlag, Berlin 1999.
 - [13] Bellare M, Goldreich O: On Defining Proofs of Knowledge. Crypto '92, LNCS 740, Springer-Verlag, Berlin 1993, 390-420.
 - [14] Bellare M, Rogaway P: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols; 1st ACM Conference on Computer and Communications Security, Proceedings, Fairfax, November 1993, acm Press, New York 1993, 62-73.
 - [15] Beth T: Efficient Zero-Knowledge Identification for Smart Cards. Eurocrypt '88, LNCS 330, Springer-Verlag, Berlin 1988, 77-84.
 - [16] Bigün J, Chollet G, Borgefors G (Eds.): Audio and Video-based Biometric Person Authentication (AVBPA '97), LNCS 1206, Springer-Verlag, Berlin 1997.
 - [17] Biskup J: Medical Database Security; Data Protection and Confidentiality in Health Informatics – Handling Health Data in Europe in the Future, Edited by the Commission of the European Communities DG XIII/F AIM, Proc. of the AIM Working Conference, Brussels, 19-21 March 1990, IOS Press, Amsterdam 1991, 214-230.
 - [18] Biskup J: Protection of privacy and confidentiality in medical information systems; Database Security, III: Status and Prospects (Spooner DL, Landwehr CE, Eds), North-Holland 1990, 13 - 23.
 - [19] Biskup J, Bleumer G: Cryptographic Protection of Health Information: Cost and Benefit; International Journal of Bio-Medical Computing, 43 (1996), 61-68. []
 - [20] Blaze M, Bleumer G, Strauss M: Divertible protocols and atomic proxy cryptography; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 127-144.
 - [21] Bleumer G: Group Signatures and Group Credentials; Hildesheimer Informatik-Berichte 2/97 (Januar 1997), Institut für Informatik, Universität Hildesheim (<http://www.mathematik.uni-hildesheim.de/FB4/Berichte/1997.html>).
 - [22] Bleumer G: On Protocol Divertibility. AT&T Labs-Research; TR97.34 <http://www.research.att.com/library/trs>.
 - [23] Bleumer G: Biometric yet Privacy Protecting Person Authentication; Information Hiding Workshop '98, LNCS 1525, Springer-Verlag, Berlin 1998, 101-112.
 - [24] Bleumer G: Biometric Authentication and Multilateral Security; Multilateral Security in Communications (Müller G, Rannenber K, Eds.), Addison-Wesley, München 1999, 157-172.
 - [25] Bleumer G, Pfitzmann B, Waidner M: A remark on a signature scheme where forgery can be proved; Eurocrypt '90, LNCS 473, Springer-Verlag, Berlin 1991, 441-445.
 - [26] Bleumer G, Schunter M: Datenschutzorientierte Abrechnung medizinischer Leistungen; Datenschutz und Datensicherheit 21/2 (1997) 88-97.
 - [27] Bleumer G, Schunter M: Privacy Oriented Clearing for the German Health Care System; in [2], 175-194.
 - [28] Bleumer G, Schunter M: Digital Patient Assistants: Privacy vs. Cost in Compulsory Health Insurance; Health Informatics Journal (1998) 4, 138-156.

- [29] Blum M, Micali S: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing* 13/4 (1984) 850-864.
- [30] Boly J-P, Bosselaers A, Cramer R, Michelsen R, Mjøl̄snes S, Muller F, Pedersen TP, Pfitzmann B, Radu C, Rooij Pd, Schoenmakers B, Schunter M: Extended Protocols; CAFE (Esprit 7023) Deliverable MAS 7304 (confidential), June 1,1995.
- [31] Bos J, Chaum D: SmartCash: A Practical Electronic Payment System; Centrum voor Wiskunde en Informatica, Computer Science/Departement of Algorithmics and Architecture, Report CS-R9035, August 1990.
- [32] Bourdeau RH, Cheng BHC: A Formal Semantics for Object Model Diagrams; *IEEE Transactions on Software Engineering* 21/10 (1995) 799-821.
- [33] Boyd C: Digital Multisignatures; in Beker HJ, Piper FC (eds.): *Cryptography and Coding*; Clarendon Press, 1989.
- [34] Brands S: An Efficient Off-line Electronic Cash System Based On The Representation Problem; Centrum voor Wiskunde en Informatica, Computer Science/Departement of Algorithmics and Architecture, Report CS-R9323, March 1993 (available from [230]).
- [35] Brands S: Untraceable Off-line Cash in Wallet with Observers; *Crypto '93*, LNCS 773, Springer-Verlag, Berlin 1994, 302-318.
- [36] Brands S: Off-Line Electronic Cash Based on Secret-Key Certificates. Second International Symposium of Latin American Theoretical Informatics (LATIN '95), April 3-7, 1995 (available from [230]).
- [37] Brands S: Restrictive Blinding of Secret-Key Certificates. *Eurocrypt '95*, LNCS 921, Springer-Verlag, Berlin 1995, 231-247.
- [38] Brands S: Rapid Demonstration of Linear Relations Connected by Boolean Operators; *Eurocrypt '97*, LNCS 1233, Springer-Verlag, Berlin 1997, 318-333.
- [39] Brands S, Chaum D, Cramer RJF, Ferguson N, Pedersen TP: Transaction Systems with Observers; Draft, CWI Amsterdam / Rijksuniversiteit Leiden / Aarhus University, December 1992.
- [40] Brassard G, Chaum D, Cr epeau C: Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences* 37 (1988) 156-189.
- [41] Brassard G: *Modern Cryptology*; Springer-Verlag, Berlin 1988.
- [42] Brickell EF, McCurley KS: An Interactive Identification Scheme Based on Discrete Logarithms and Factoring. *Journal of Cryptology* 5/1 (1992) 29-39.
- [43] Brickell E, Gemmell P, Kravitz D: Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Change; 6th ACM-SIAM Symposium on Discrete Algorithms (SODA) 1995, ACM Press, New York 1995, 457-466.
- [44] Buchholz EH: *Unser Gesundheitswesen: Ein einf uhrender  berblick zum Gesundheitswesen der Bundesrepublik Deutschland*, Springer-Verlag, Berlin, 1988.
- [45] Bundesministerium f ur Bildung, Wissenschaft, Forschung und Technologie: *INFORMATIONSGESELLSCHAFT: Chancen, Innovationen und Herausforderungen*; Rat f ur Forschung, Technologie und Innovation, Bundesministerium f ur Bildung, Wissenschaft, Forschung und Technologie, 1995.

REFERENCES

- [46] Bundesamt für Sicherheit in der Informationstechnik: Chipkarten im Gesundheitswesen; Schriftenreihe zur IT-Sicherheit Band 5, Bundesanzeiger Verlag, Köln 1995.
- [47] Camenisch JL: Efficient and Generalized Group Signatures; Eurocrypt '97, LNCS 1233, Springer-Verlag, Berlin 1997, 465-479.
- [48] Camenisch JL, Piveteau J-M, Stadler MA: An Efficient Electronic Payment System Protecting Privacy; ESORICS 94 (Third European Symposium on Research in Computer Security), Brighton, LNCS 875, Springer-Verlag, Berlin 1994, 207-215.
- [49] Camenisch JL, Piveteau J-M, Stadler MA: Blind Signatures Based on the Discrete Logarithm Problem; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 428-432.
- [50] Camenisch JL, Piveteau J-M, Stadler MA: Fair Blind Signatures; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 209-219.
- [51] Camenisch JL, Stadler MA: Efficient Group Signature Schemes for Large Groups; Crypto '97, LNCS 1294, Springer-Verlag, Berlin 1997, 410-424.
- [52] Camenisch JL, Piveteau J-M, Stadler MA: An Efficient Fair Payment System; 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 1996, ACM Press, New York 1996, 88-94.
- [53] Camenisch JL, Maurer U, Stadler M: Digital Payment Systems with Passive Anonymity-Revoking Trustees; ESORICS 96, LNCS 1146, Springer-Verlag, Berlin 1996, 33-43.
- [54] Camenisch JL, Maurer U, Stadler M: Digital Payment Systems with Passive Anonymity-Revoking Trustees; Journal of Computer Security, 5/1 (1997), IOS-Press, Amsterdam 1997, 69-89.
- [55] Campbell D (Dick Holdsworth ed): Development of Surveillance Technology and Risk of Abuse of Economic Information; European Parliament, Directorate General for Research, Directorate A, The STOA Programme, April 1999.
- [56] Canetti R, Goldreich O, Halevi S: The random oracle methodology, revisited; 30th Symposium on Theory of Computing (STOC) 1998, ACM, New York 1998, 209-218.
- [57] Chaum D: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88.
- [58] Chaum D: Blind Signatures for untraceable payments; Crypto '82, Plenum Press, New York 1983, 199-203.
- [59] Chaum D: Blind Signature System; Crypto '83, Plenum Press, New York 1984, 153.
- [60] Chaum D: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044.
- [61] Chaum D: Showing Credentials without Identification - Signatures Transferred Between Unconditionally Unlinkable Pseudonyms; Eurocrypt '85, LNCS 219, Springer-Verlag, Berlin 1986, 241-244.
- [62] Chaum D: Card-Computer Moderated Systems; US Patent 4,926,480, Issued: May 15, 1990.
- [63] Chaum D: Online cash checks; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 288-293.
- [64] Chaum D: Showing Credentials without Identification: Transferring Signatures between Unconditionally Unlinkable Pseudonyms; Auscrypt '90, LNCS 453, Springer-Verlag, Berlin 1990, 246-264.
- [65] Chaum D: Achieving Electronic Privacy; Scientific American (August 1992) 96-101.

- [66] Chaum D, van Antwerpen CJ: Undeniable signatures; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 212-216.
- [67] Chaum D, den Boer B, van Heijst E, Mjøl̄snes S, Steenbeek A: Efficient offline electronic checks; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 294-301.
- [68] Chaum D, Evertse J-H, van de Graaf J: An improved protocol for demonstrating possession of discrete logarithms and some generalizations. Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 127-141.
- [69] Chaum D, Fiat A, Naor M: Untraceable Electronic Cash; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 319-327.
- [70] Chaum D, van Heijst E: Group signatures; Eurocrypt '91, LNCS 547, Springer-Verlag, Berlin 1991, 257-265.
- [71] Chaum D, van Heijst E, Pfitzmann B: Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 470-484.
- [72] Chaum D, Pedersen TP: Transferred cash grows in size; Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 390-407.
- [73] Chaum D, Pedersen TP: Wallet Databases with Observers; Crypto '92, LNCS 740, Springer-Verlag, Berlin 1993, 89-105.
- [74] Chen L: Witness Hiding Proofs and Applications; DAIMI PB-477, Computer Science Department Aarhus University, August 1994.
- [75] Chen L: Oblivious Signatures; ESORICS 94, LNCS 875, Springer-Verlag, Berlin 1994, 161-172.
- [76] Chen L: Access with Pseudonyms; Cryptography: Policy and Algorithms, LNCS 1029, Springer-Verlag, Berlin 1996, 232-243.
- [77] Chen L, Pedersen TP: New Group Signature Schemes; Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 171-181.
- [78] Commission of the European Communities: Research and Technology Development on Telematic Systems in Health Care; Commission of the European Communities, Rue de Treves 61, B-1040 Bruxelles Belgium.
- [79] Cramer R, Damgård I, Schoenmakers B: Proofs of partial knowledge and simplified design of witness hiding Protocols. Crypto 94, LNCS 839, Springer-Verlag, Berlin 1994 174-187; CWI Quarterly Journal 8/2 (1995) 111-128.
- [80] Cramer RJF, Pedersen TP: Improved Privacy in Wallets with Observers (Extended Abstract); Eurocrypt '93, LNCS 765, Springer-Verlag, Berlin 1994, 329-343.
- [81] Croft R, Harris S: Public key cryptography and re-usable shared secrets; in Beker HJ, Piper FC (eds.): Cryptography and Coding; Clarendon Press, 1989
- [82] Damgård IB: Collision free hash functions and public key signature schemes; Eurocrypt '87, LNCS 304, Springer-Verlag, Berlin 1988, 203-216.
- [83] Damgård IB: Payment Systems and Credential Mechanisms with Provable Security Against Abuse by Individuals; Crypto '88, LNCS 403, Springer-Verlag, Berlin 1990, 328-335.
- [84] Davida G: Chosen Signature Cryptanalysis of the RSA (MIT) Public Key Cryptosystem; TR-CS-82-2, University of Wisconsin, Milwaukee (October 1982).

REFERENCES

- [85] Davida G, Frankel Y, Tsiounis Y, Yung M: Anonymity Control in E-Cash Systems; *Financial Cryptography '97*, LNCS 1318, Springer-Verlag, Berlin 1997, 1-16.
- [86] Davida G, Frankel Y, Matt BJ: On Enabling Secure Applications Through Off-Line Biometric Identification; *IEEE Symposium on Security and Privacy*, IEEE Press 1998, 148-159.
- [87] Davida G, Frankel Y: Perfectly Secure Authorization and Passive Identification for an Error Tolerant Biometric System; *Cryptography and Coding 1999*, LNCS 1746, Springer Verlag 1999, 104-113.
- [88] Davies DW, Price WL: *Security for Computer Networks, An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer*; (2nd ed.) John Wiley & Sons, New York 1989.
- [89] Deane F, Barrelle K, Henderson R, Mahar D: Perceived acceptability of biometric security systems; *Computers & Security* 14/3 (1995) 225-231.
- [90] Desmedt Y: Threshold cryptography. *Auscrypt '92*, LNCS 718, Springer-Verlag, Berlin 1993, 3-14.
- [91] Desmedt YG, Frankel Y: Threshold Cryptosystems. *Crypto '89*, LNCS 435, Springer-Verlag, Berlin 1990, 307-315.
- [92] Desmedt Y, Goutier C, Bengio S: Special uses and abuses of the Fiat-Shamir passport protocol. *Crypto '87*, LNCS 293, Springer-Verlag, Berlin 1988, 21-39.
- [93] Diffie W: The Impact of a Secret Cryptographic Standard on Encryption, Privacy, Law Enforcement and Technology; in [133].
- [94] Diffie W, Hellman ME: New Directions in Cryptography; *IEEE Transactions on Information Theory* 22/6 (1976) 644-654.
- [95] Dobbertin H: The Status of MD5 After a Recent Attack. *CryptoBytes* 2/2 (1996), RSA Labs, Redwood City 1996 (see [241]).
- [96] Dobbertin H, Bosselaers A, Preneel B: RIPEMD-160: A Strengthened Version of RIPEMD. in: Goos G, Hartmanis J, van Leeuwen J (ed.), *Fast Software Encryption, Third International Workshop*, Cambridge, U.K., February 21-23, 1996, Proceedings, LNCS 1039, Springer-Verlag, Berlin 1996, 71-82.
- [97] Druffel L: Information Warfare; in Denning PJ, Metcalfe RM (eds): *Beyond Calculation*; Copernicus Springer-Verlag, New York 1997, 193-212.
- [98] Dwork C, Lotspiech J, Naor M: Digital signets: Self-enforcing protection of digital information; *28th Symposium on Theory of Computing (STOC) 1996*, ACM, New York 1996, 489-498.
- [99] ElGamal T: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory* 31/4 (1985) 469-472.
- [100] Even S: Secure Off-Line Electronic Fund Transfer Between Nontrusting Parties; *SMART CARD 2000: The Future of IC Cards*, Proceedings of the IFIP WG 11.6 International Conference; Laxenburg (Austria), 19.-20. 10. 1987, North-Holland, Amsterdam 1989, 57-66.
- [101] Even S, Goldreich O, Yacobi Y: Electronic wallet; *Crypto '83*, Plenum Press, New York 1984, 383-386.
- [102] Even S, Goldreich O, Yacobi Y: Electronic Wallet; *1984 International Zurich Seminar on Digital Communications, Applications of Source Coding, Channel Coding and Secrecy Coding*, March

- 6-8, 1984, Zurich, Switzerland, Swiss Federal Institute of Technology, Proceedings IEEE Catalog no. 84CH1998-4, 199-201.
- [103] Goldreich O, Pfitzmann B, Rivest RL: Self-Delegation with controlled propagation—or—what if you lose your laptop; Crypto '98, LNCS 1462, Springer-Verlag, Berlin 1998, 153-168.
 - [104] Even S, Yacobi Y: Relations Among Public Key Signature Systems; Technical Report no. 175, March 1980, Computer Science Department, Technion, Haifa, Israel.
 - [105] Famulla R, Gerlof H: Das Kartenhaus in der Medizin wächst viel langsamer als gedacht; Arzt Online Nr. 6, das Computermagazin der Ärzte Zeitung, Oktober 1996. <http://www2.aerztezeitung.de/de/htm/medpc/karten/06ao1201.htm>
 - [106] Feige U, Fiat A, Shamir A: Zero-Knowledge Proofs of Identity; Journal of Cryptology 1/2 (1988) 77-94.
 - [107] Feige U, Shamir A: Witness Indistinguishable and Witness Hiding Protocols; Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC), May 14-16, 1990, Baltimore-Maryland, ACM Press, 416-426.
 - [108] Ferguson N: Single Term Off-Line Coins; Eurocrypt '93, LNCS 765, Springer-Verlag, Berlin 1994, 318-328.
 - [109] Ferguson N: Extensions of Single-Term Coins; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994, 292-301.
 - [110] Fiat A, Shamir A: How to Prove Yourself: Practical Solutions to Identification and Signature Problems; Crypto '86, LNCS 263, Springer-Verlag, Berlin 1987, 186-194.
 - [111] Frankel Y, Desmedt Y: Classification of ideal homomorphic threshold schemes over finite Abelian groups. Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 25-34.
 - [112] Frankel Y, Tsionis Y, Yung M: Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-cash; Asiacrypt '96, LNCS 1163, Springer-Verlag, Berlin 1996, 286-300.
 - [113] Franklin M, Yung M: Secure and Efficient Off-Line Digital Money; 20th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 700, Springer-Verlag, Heidelberg 1993, 265-276.
 - [114] Garfinkel S: PGP Pretty Good Privacy; O'Reilly & Associates, Sebastopol 1995.
 - [115] Garfinkel S: Database Nation; O'Reilly & Associates, Sebastopol 2000.
 - [116] Graham RL, Knuth DE, Patashnik O: Concrete Mathematics; Addison-Wesley, Reading (Mass.) 1989. (A list of Errata by 12/1996 is available from [236])
 - [117] Goldreich O: A Uniform-Complexity Treatment of Encryption and Zero-Knowledge; Journal of Cryptology 6/1 (1993) 21-53.
 - [118] Goldreich O: Foundations of Cryptography (Fragments of a Book); Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, February 23, 1995. Available from [234].
 - [119] Goldreich O, Krawczyk H: On the Composition of Zero-Knowledge Proof Systems. SIAM Journal on Computing 25/1 (1996) 169-192.
 - [120] Goldreich O: Modern Cryptography, Probabilistic Proofs and Pseudorandomness; Springer-Verlag, Berlin 1998.

REFERENCES

- [121] Goldwasser S, Micali S: Probabilistic Encryption. *Journal of Computer and System Sciences* 28 (1984) 270-299.
- [122] Goldwasser S, Micali S, Rivest RL: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; *SIAM J. Comput.* 17/2 (1988) 281-308.
- [123] Goldwasser S, Micali S, Rackoff C: The Knowledge Complexity of Interactive Proof Systems; *SIAM J. Comput.* 18/1 (1989) 186-207.
- [124] Gordon DM: Discrete logarithms in $GF(p)$ using the number field sieve. *SIAM Journal on Discrete Mathematics* 6/1 (1993) 124-138.
- [125] Guillou LC, Quisquater JJ: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. *Eurocrypt '88, LNCS 330, Springer-Verlag, Berlin 1988, 123-128.*
- [126] Harrison MA, Ruzzo WL, Ullman JD: Protection in Operating Systems. *Communications of the ACM* 19/8 (1976) 461-471.
- [127] Häußler S, Liebold R, Narr H: *Die Kassenärztliche Tätigkeit*; Springer-Verlag, Berlin, 1984.
- [128] Hayes B: Anonymous one-time signatures and flexible untraceable electronic cash; *Auscrypt '90, LNCS 453, Springer-Verlag, Berlin 1990, 294-305.*
- [129] van Heijst E, Pedersen TP: How to make efficient Fail-stop signatures; *Eurocrypt '92, LNCS 658, Springer-Verlag, Berlin 1993, 366-377.*
- [130] Hellman ME: Response to NIST's Proposal; *Communications of the ACM* 35/7 (1992) 47-49.
- [131] Hirschfeld R: Making Electronic Refunds Safer; *Crypto '92, LNCS 740, Springer-Verlag, Berlin 1993, 106-112.*
- [132] Hoare CAR: An Axiomatic Basis for Computer Programming; *Communications of the ACM* 12/10 (1969) 576-580,583.
- [133] Hoffmann LJ (ed.): *Building in Big Brother*; Springer-Verlag, New York 1995.
- [134] Hormel M, Konen W, Fuhrmann S, Flügel A: Neural Systems for Complex Identification Tasks: The Access Control System ZN_Face and the Alarm Identification SENECA for Steel Casting Processes; *ICANN 1996, LNCS 1112, Springer-Verlag 1996. (see also [237])*
- [135] Horster P, Michels M, Petersen H: Meta-Message Recovery and Meta-Blind Signature Schemes Based on the Discrete Logarithm Problem and Their Applications; *Asiacrypt '94, LNCS 917, Springer-Verlag, Berlin 1995, 224-237.*
- [136] Hoshen J, Sennott J, Winkler M: Keeping tabs on criminals; *IEEE Spectrum* 32/2 (1995), 26-32.
- [137] Grimson J, Grimson W, Hasselbring W: The SI Challenge in Health Care; *Communications of the ACM* 43/6 (2000) 49-55.
- [138] ISO/IEC 9594-8: Information technology - Open Systems Interconnection - Specification - The Directory - Part 8: Authentication framework; ISO International Standard, First edition 15.12.1990, with Technical Corrigendum 1, 15.12.1991.
- [139] Jacobson N: *Basic Algebra I*; (2nd. ed.) Freeman, New York 1985.
- [140] Jain A, Hong L, Pankanti S: Biometric Identification; *Communications of the ACM* 43/2 (2000) 91-98.
- [141] Jones VE, Ching N, Winslett M: Credentials for Privacy and Interoperation; *New Security Paradigms Workshop, August 1995, IEEE Press, Los Alamitos 1995, 92-100.*

- [142] Kohl J, Neumann BC: The Kerberos™ Network Authentication Service (V5); Internet Draft 1/9/1992 via ftp: prep.ai.mit.edu, path: pub/kerberos.
- [143] Kozyrakis CE, Patterson DA: A New Direction for Computer Architecture Research; Computer 31/11 (1998) 24-32.
- [144] van Leeuwen J (ed.): Handbook of Theoretical Computer Science (Vol.B: Formal Models and Semantics); Elsevier Science Publishers B.V., Amsterdam, 1990.
- [145] Lenstra AK, Lenstra HW: Algorithms in Number Theory. in: [144], 673-715.
- [146] Lenstra AK, Verheul ER: Selecting Cryptographic Key Sizes; Journal of Cryptology 14/4 (2001) 255-293.
- [147] Lysyanskaya A, Rivest RL, Sahai A, Wolf S: Pseudonym Systems; in Heys HM, Adams CM (Eds.) Proceedings of Selected Areas in Cryptography SAC '99, LNCS 1758, Springer-Verlag, Berlin 2000.
- [148] Low SH, Maxemchuk NF: Anonymous Credit Cards; 2nd ACM Conference on Computer and Communications Security, Fairfax, November 1994, ACM Press, New York 1994, 108-117.
- [149] Lysyanskaya A, Ramzan Z: Group Blind Digital Signatures: A Scalable Solution to Electronic Cash; Financial Cryptography 1998, LNCS 1465, Springer-Verlag 1998, 184-197.
- [150] Macchiavelli N: The Prince; Oxford Univ Press, Oxford 1987.
- [151] Manna Z, Pnueli A: The Temporal Logic of Reactive and Concurrent Systems; Springer-Verlag, New York 1992.
- [152] Manna Z, Pnueli A: Temporal Verification of Reactive Systems; Springer-Verlag, New York 1995.
- [153] McCurley KS: The Discrete Logarithm Problem; Pomerance C (ed.): Cryptology and Computational Number Theory; Proceedings of Symposia in Applied Mathematics, Volume 42, American Mathematical Society, Providence 1990, 49-74.
- [154] McLean J: Security Models; in: John Marciniak (ed.): Encyclopedia of Software Engineering; Wiley Press, 1994.
- [155] McMahan PV: SESAME V2 Public Key and Authorisation Extensions to Kerberos; Proceedings of the Symposium on Network and Distributed System Security, San Diego, California, February 16-17, 1995, IEEE Computer Society Press, California, 114-131.
- [156] Maxemchuk NF, Low SH: The Use of Communication Networks to Increase Personal Privacy in a Health Insurance Architecture; Manuscript, 1995
- [157] Menezes AJ, van Oorschot PC, Vanstone SA: Handbook of Applied Cryptography; CRC Press, Boca Raton 1997. Also see [238]
- [158] Miller B: Vital signs of identity; IEEE spectrum 31/2 (1994) 22-30.
- [159] Mjølsetnes SF: Privacy, Cryptographic Pseudonyms, and the State of Health; Asiacrypt '91, Proceedings, Fujiyoshida, November 11-14, Springer-Verlag, Berlin 1993, 493-494.
- [160] Naccache D, von Solms S: On Blind Signatures and Perfect Crimes; Computers & Security 11/6 (1992) 581-583.
- [161] Neumann PG: Risks of Surveillance; Communications of the ACM 36/8 (1993) 122.
- [162] Neumann PG: Using Formal Methods to Reduce Risks; Communications of the ACM 39/7 (1996) 114.

REFERENCES

- [163] Neuman BC, Kohl J: The Kerberos Network Authentication Service (V5); Internet RFC-1510, 1993.
- [164] National Institute of Standards and Technology (NIST): Digital Signature Standard. Federal Information Processing Standards Publication (FIPS PUB 186), February 1, 1993.
- [165] National Institute of Standards and Technology (NIST): Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB 180-1), 17 April 1995.
- [166] National Institute of Standards and Technology (NIST): Key Management Guideline Part 3 Draft; October 2001; see [239].
- [167] Nguyen KQ, Mu Y, Varadharajan V: Divertible Zero-Knowledge Proof of Polynomial Relations and Blind Group Signature; Australasian Conference on Information Security and Privacy ACISP '99, LNCS 1587, Springer-Verlag, Berlin 1999, 117-128.
- [168] Nyberg K, Rueppel R: A New Signature Scheme Based on the DSA Giving Message Recovery. 1st ACM Conference on Computer and Communications Security, Proceedings, Fairfax, November 1993, acm Press, New York 1993, 58-61.
- [169] Nyberg K, Rueppel R: Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem. Eurocrypt '94, LNCS 950, Springer-Verlag, Berlin 1995, 182-193.
- [170] Okamoto T: A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems; ACM Transactions on Computer Systems 6/4 (1988) 432-441.
- [171] Ohta K, Okamoto T: A Digital Multisignature Scheme Based on the Fiat-Shamir Scheme. Asiacrypt '91, Proceedings, Fujiyoshida, November 11-14, Springer-Verlag, Berlin 1993, 139-148.
- [172] Okamoto T: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes; Crypto '92, LNCS 740, Springer-Verlag, Berlin 1993, 31-44.
- [173] Okamoto T, Ohta K: Divertible zero-knowledge interactive proofs and commutative random self-reducibility; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 134-149.
- [174] Okamoto T, Ohta K: Disposable Zero-knowledge Authentications and Their Applications to Untraceable Electronic Cash; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 481-496.
- [175] Okamoto T, Ohta K: Universal Electronic Cash; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 324-337.
- [176] Oppliger R: Authentication Systems for Secure Networks; Artech House, Norwood, MA 1996
- [177] Pedersen TP: Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem. DAIMI PB - 388, Computer Science Department, Aarhus University, Ny Munksgade, Building 540, DK-8000 Aarhus C, Denmark, March 1992.
- [178] Pedersen TP: Electronic Payments of Small Amounts; DAIMI PB-495, Computer Science Department Aarhus University, August 1995.
- [179] Pedersen TP, Pfitzmann B: Fail-Stop Signatures; SIAM Journal on Computing 26/2 (1997) 291-330.
- [180] Pfitzmann B: Für den Unterzeichner unbedingt sichere digitale Signaturen und ihre Anwendung; Diplomarbeit am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe 1989.
- [181] Pfitzmann A: Sicherheit in Rechnernetzen: Sicherheit in verteilten und durch verteilte Systeme; Lecture Notes in German, 1994.

- [182] Pfitzmann B: Sorting Out Signature Schemes; 1st ACM Conference on Computer and Communications Security, 3.-5.11.1993, Fairfax, acm press 1993, 74-85.
- [183] Pfitzmann B: Digital Signature Schemes; LNCS 1100, Springer-Verlag, Berlin 1996.
- [184] Pfitzmann B, Pfitzmann A: How to Break the Direct RSA-Implementation of MIXes; Eurocrypt '89, LNCS 434, Springer-Verlag, Berlin 1990, 373-381.
- [185] Pfitzmann A, Pfitzmann B, Schunter M, Waidner M: Trusting Mobile User Devices and Security Modules; Computer 30/2 (1997) 61-68.
- [186] Pfitzmann B, Sadeghi AR: Coin-Based Anonymous Fingerprinting; Eurocrypt '99, LNCS 1592, Springer-Verlag, Berlin 1999, 150-164.
- [187] Pfitzmann B, Waidner M: How to Break and Repair a "Provably Secure" Untraceable Payment System; Crypto '91, LNCS 576, Springer-Verlag, Berlin 1992, 338-350.
- [188] Pfitzmann B, Waidner M: A General Framework for Formal Notions of "Secure" System; Hildesheimer Informatik-Berichte 11/94, ISSN 0941-3014, Institut für Informatik, Universität Hildesheim, April 1994.
- [189] Pfitzmann B, Waidner M: Properties of Payment Systems: General Definition Sketch and Classification; IBM Research Report RZ 2823 (#90126) 05/06/96, IBM Research Division, Zurich, May 1996.
- [190] Pohlig SC, Hellman ME: An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance. IEEE Transactions on Information Theory 10/1 (1978) 106-110.
- [191] Pointcheval D, Stern J: Security proofs for signature schemes; Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 387-398.
- [192] Pointcheval D, Stern J: Provably Secure Blind Signature Schemes; Asiacrypt 96, LNCS 1163, Springer-Verlag, Berlin 1996, 252-265.
- [193] Pointcheval D: Strengthened security for blind signatures; Eurocrypt '98, LNCS 1403, Springer-Verlag, Berlin 1998, 391-405.
- [194] Pommerening K, Pseudonyme - ein Kompromiß zwischen Anonymisierung und Personenbezug; in: Trampisch HJ, Lange S (Hrsg.): Medizinische Forschung — Ärztliches Handeln; 40. Jahrestagung der GMDS, Bochum, September 1995, MMV Medizin Verlag, München 1995, 329-333.
- [195] Pommerening K: Chipkarten und Pseudonyme; F!FF Kommunikation 1/96, 9-12.
- [196] Rabin MO: Digitalized Signatures; Foundations of Secure Computation, ed. by R.A. DeMillo, D.P. Dobkin, A.K. Jones, R.J. Lipton; Academic Press, N.Y. 1978, 155-166.
- [197] Radu C, Govaerts R, Vandewalle J: A Restrictive Blind Signature Scheme with Applications to Electronic Cash; Communications and Multimedia Security II; Chapman & Hall, London 1996, 196-207.
- [198] Radu C, Govaerts R, Vandewalle J: Efficient electronic cash with restricted privacy; Financial Cryptography '97, Springer-Verlag, Berlin 1997, 57-69.
- [199] Rao JR, Rohatgi P: Can Pseudonyms Really Guarantee Privacy?; 9th Usenix Symposium, August 2000.
- [200] Reiter MK, Rubin AD: Crowds: Anonymity for Web Transactions; DIMACS Technical Report 97-15, April 1997, revised August 1997, <http://www.research.att.com/~reiter/papers/dimacs->

REFERENCES

- tr9715-revised. ps.gz, to appear in ACM Transactions on Information and System Security 1/1 (1998).
- [201] Rivest RL, Shamir A, Adleman L: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems; Communications of the ACM 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96-99.
 - [202] Rivest RL: Response to NIST's Proposal; Communications of the ACM 35/7 (1992) 41-47.
 - [203] Sakurai K, Yamane Y: Blind Decoding, Blind Undeniable Signatures, and Their Applications to Privacy Protection; in [1], 257-264.
 - [204] Salomaa A: Public-Key Cryptography (2nd ed.); Springer-Verlag, Berlin 1996
 - [205] de Santis A, Persiano G: Communication Efficient Zero-Knowledge Proofs of Knowledge (With Applications to Electronic Cash); STACS 92, LNCS 577, Springer-Verlag, Heidelberg 1992, 449-460.
 - [206] Schneier B: Applied Cryptography (2nd ed.); John Wiley, New York 1996.
 - [207] Schneier B: Cryptography, Security, and the Future; Communications of the ACM 40/1 (1997) 138.
 - [208] Schnorr CP: Efficient identification and signatures for smart cards; Crypto '89, LNCS 435, Springer-Verlag, Heidelberg 1990, 239-252.
 - [209] Schnorr CP: Efficient Signature Generation by Smart Cards; Journal of Cryptology 4/3 (1988) 161-174.
 - [210] Schoenmakers B: Efficient Proofs of Or; Manuscript, 1993.
 - [211] Schunter M: Optimistic Fair Exchange; Dissertation, Universität des Saarlandes, Saarbrücken 2000.
 - [212] Sherman RL: Biometrics Futures; Computers & Security 11/2 (1992) 128-133.
 - [213] Simmons GJ: The Prisoners' Problem and the Subliminal Channel; Crypto '83, Plenum Press, New York 1984, 51-67.
 - [214] Shoup V, Gennaro R: Securing Threshold Cryptosystems against Chosen Ciphertext Attack; Eurocrypt '98, LNCS 1403, Springer-Verlag, Heidelberg 1998, 1-16.
 - [215] Simmons GJ (ed.): Contemporary Cryptology – The Science of Information Integrity; IEEE Press, Hoes Lane 1992.
 - [216] Stadler M: Publicly Verifiable Secret Sharing; Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 190-199.
 - [217] Stadler M, Piveteau J-M, Camenisch J: Fair Blind Signatures; Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, 209-219.
 - [218] Stinson DR: Cryptography, Theory and Practice; CRC Press, Boca Raton 1995.
 - [219] Struif B: Das elektronische Rezept mit digitaler Unterschrift; Reimer H, Struif B (eds.): Kommunikation & Sicherheit, TeleTrust Deutschland e.V., Darmstadt 1992, 71-75.
 - [220] Struif B: Sicherheit und Datenschutz bei elektronischen Rezepten; Multicard'94, Elektronische Kartensysteme - Anspruch und Wirklichkeit, Kongreßdokument I, 23.-25. Februar 1994, Berlin, 71-80.
 - [221] Sudhakaran Ram: Medical Prescription Dispensing Robot; RISKS-17.73 (1996) <ftp://unix.sri.com/risks>,

- [222] Tompa M, Woll H: Random self-reducibility and zero knowledge proofs of possession of information; 28th Symposium on Foundations of Computer Science (FOCS) 1987, IEEE Computer Society, 1987, 472-482.
- [223] Traoré J: Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems; Australasian Conference on Information Security and Privacy ACISP '99, LNCS 1587, Springer-Verlag, Berlin 1999, 228-243.
- [224] Yiannis Tsiounis, Moti Yung: On the Security of ElGamal based Encryption; 1st International Workshop on Practice and Theory in Public Key Cryptography (PKC 98), LNCS 1431, Springer-Verlag, Berlin 1998, 117-134.
- [225] Wenk E: Computers and Society; 26/2 (1996), 34.
- [226] Winner L: Computers and Society; 25/4 (1995), 50.
- [227] Yao AC: Theory and Applications of Trapdoor Functions. 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 80-91.

C.2 Web Links

- [228] André Bacard: Anonymous Remailers; FAQ, Version Nov 15, 1996, <http://www.well.com/user/abacard/remail.html>
- [229] AuthenTec: FingerLoc™; <http://www.semi.harris.com>.
- [230] Stefan Brands
<http://www.cwi.nl/>
- [231] CEN TC251
<http://www.centc251.org>
- [232] DiabCard: Improved Communication in Diabetes Care Based on Chip Card Technology; <http://www.med.auth.gr/localsrv/medical/projects/lomi-pro/diabcard.htm>
- [233] Identix (Fingerprinting Systems)
<http://www.identix.com>
- [234] Oded Goldreich (Zero Knowledge Proofs)
<ftp://ftp.wisdom.weizmann.ac.il/pub/oded/bookfrag>
- [235] Will Harris (Typography):
<http://www.will-harris.com>
- [236] Donald E. Knuth
<http://www-cs-staff.Stanford.EDU/~knuth/>
- [237] Peter Kruizinga (Face Recognition)
<http://www.cs.rug.nl/~peterkr/FACE/face.html>
- [238] Menezes AJ, van Oorschot PC, Vanstone SA: Handbook of Applied Cryptography
<http://www.cacr.math.uwaterloo.ca/hac/>
- [239] NIST Computer Security Resource Center (CSRC): Key Management Guideline Part 3:
<http://csrc.nist.gov/CryptoToolkit/tkkeymgmt.html>
- [240] OMG CORBAmed
<http://www.omg.org/>

REFERENCES

- [241] RSA-Labs
<http://www.rsa.com/rsalabs/PUBS>
- [242] The Clinical Information Consultancy: Healthcards; <http://www.compulink.co.uk/~cic/euhci.htm>



Index of Keywords

A

- abort *see* execution of a protocol 17
- acceptor 20
- active attack 24, 52
- active attacker 24
- actual clearing houses 147
- airfare tickets 9
- anonymity
 - relative 66
- anonymizing group 123
- ATM (automatic teller machine) 67
- attack
 - active 24, 52
 - computational 24
 - computationally unlimited 24
 - generic 31
 - passive 24, 52
- attack goals
 - existential forgery 37, 52
 - selective forgery 37, 52
 - total break 37, 52
 - universal break 37, 52
- attacker
 - active 24
 - computational 24
 - computationally unlimited 24
 - passive 24, 52
- attacker model 24
- attacking participants 24
- attacking protocol 25
- attacking schedule 24
- authorization
 - enrollment 151
 - prescription 152
 - referral 152
- automatic teller machine (ATM) 67

B

- big-O 15
- biometer 69
 - biometric template 78
 - personalization 78
- biometric identity
 - credential scheme 76
- biometric observer *see* observer 69, 90
- biometric recognition 70
- biometric template 74, 78, 79, 82, 83, 84, 86, 93, 97, 100
- biometric template *see* biometer 78
- biometric verification 70
- blind signature
 - valid 54
- blind signature *see* signature scheme 53
- blinder
 - signature scheme 53
- blindness
 - signature scheme 54
- bonds 9
- bottom 20
- broken observer 82

C

- candidate
 - proof-of-knowledge scheme 35, 41
- captured agent trust 67
- cash 8
- cell phone 67
- centralized group signature scheme 124
- certificate 6
 - certifier 6
 - content 6
 - holder 6
 - non-personal 9
 - owner 6
 - personal 10

INDEX OF KEYWORDS

- personal bonus/malus 10
- public key 59
- tradeable 10
- certification problem 60, 61
- Chaum-Pedersen Signature Scheme 55, 56
 - extended signing protocol 56
- check
 - off-line
 - post-paid 66
 - pre-paid 66
- check credentials 153
- Chen-Pedersen Group Signature Scheme
 - extended signing protocol 133
- choosing an element at random *see* ITM 15
- coalition resistance
 - group signature scheme 129
- co-make a pseudonym 76
- co-making function
 - credential scheme 76
 - extended proof-of-knowledge scheme 41
- common input parameter *see* protocol 20
- common input *see* protocol 20
- common output parameter *see* protocol 20
- common output *see* protocol 20
- communication tape of an ITM
 - read-only 15
 - write-only 15
- communication topology *see* protocol 16
- completeness
 - proof-of-knowledge scheme 36
- composed participant 22
- composition tape
 - initial content 24
 - polynomial-size 24
- composition tape of an ITM 15
- compositional input to an ITM 15
- compositional output of an ITM 15
- computation time of a protocol 16, 20
- computation time of an ITM 15
- computational attack 24
- computational attacker 24
- computational security requirement, security requirement
 - computational 25
- computationally unlimited attack 24
- computationally unlimited attacker 24
- computationally unlimited ITM 15, 16
- confidentiality requirements 6
- content 6
- control center 61
- convince 35
- co-witness
 - extended proof-of-knowledge scheme 41
- credential
 - check 153
 - credential scheme 76
 - K -show 61
 - nonce 61
 - personal 59
 - tradeable 59
 - valid 78
- credential scheme 72
 - biometric identity 76
 - co-make a pseudonym 76

- co-making function 76
- credential 76
- intermediate pseudonym 72
- intro pseudonym 72
- k -overshow detection 84
- k -overshow prevention 84
- making function 76
- overshow prevention 84
- prekey 76
- private key 76
- pseudonym 76
- pseudonym co-witness 76
- pseudonym witness 76
- security parameter 76
- source pseudonym 72
- target pseudonym 72
- transcript 76
- unlinkability
 - show-wise fail-safe 87
 - witness makes a pseudonym 76
- critical transactions 68
- cryptographic mechanism 23
- cryptographic scheme 23
 - implementation 25
 - instance 25
 - prekey 23
- cut-and-choose 63, 65

D

- decentralized group signature scheme 124
- digital signature 59
 - blind 62
- discrete log framework (DLF) 28, 29
- discrete log setting (DLS) 28, 29
- discrete logarithm 28
- dividability 66
- divisability 66
- DLF (discrete log framework) 28, 29
- double signing 125
- dual control 2
- dynamic group signature scheme 124

E

- effectiveness
 - signature scheme 54
- efficient ITM 15, 16
- electronic wallet 67
- enrollment authorization (E-authorization) 151
- ensemble 14
- event 13
- execution
 - abort 17
 - successful 17
- execution function *see* protocol 17, 18
- execution of an ITM 17
- existential forgery *see* attack goals 37, 52
- existential forgery under passive attack 55
- expected computation time of a protocol 20
- explicit introduction of pseudonym 77, 80, 84

- extended node protocol 85
- extended proof of knowledge
 - co-witness matches a candidate 41
- extended signing protocol
 - Chaum-Pedersen Signature Scheme 56
 - Chen-Pedersen Group Signature Scheme 133
- extendedness
 - extended proof-of-knowledge scheme 42
- external interface of participant 19
- external interface *see* protocol 19

F

- family 14
- flipping a coin *see* Turing machine 15

G

- general composition of ITMs 18
- general composition of protocols 21
- generic attack 31
- GMR-Definition 51
- group credential 59
- group credential scheme
 - private group key 142
 - public individual key 142
- group effectiveness
 - blind group signature scheme 127
- group key
 - private 127, 142
 - public 127, 142
- group signature scheme
 - centralized 124
 - coalition resistance 129
 - decentralized 124
 - double signing 125
 - dynamic 124
 - group effectiveness 127
 - private group key 126
 - public group key 126, 142
 - public individual key 126
 - responsible for a signature 127, 142
 - signer anonymity 128
 - static 124
 - unframeability 128
- group-signatures 123

H

- health insurers 147
- healthcare providers 147
 - prescription groups 156
 - referral groups 156
 - registration 151
- hierarchy of legitimation types 11
- holder 6
- holder authorization 7, 10, 60
- holder authorization problem 60
- holder framing 81
- honest participant 24

- hospital physicians 148

I

- implementation of a cryptographic scheme 25
- implicit introduction of pseudonym 77, 80, 84
- in-band inflow 68
- in-band outflow 68
- index set 14
- individual key
 - private 127, 142
 - public 127, 142
- inflow
 - in-band 68
 - out-of-band 69
- information warfare *iv*
- initial content 24
- initial node protocol 85
- Inpatient treatment 146
- input tape of an ITM 15
- input to an ITM 15
- instance of a cryptographic scheme 25
- instance of a protocol 20
- instance of an ITM 17
- intact observer 82
- integrity requirements 6
- interactive protocol 16
- interactive Turing machine 15
 - computation time 15
 - computationally unlimited 15, 16
 - efficient 15, 16
 - polynomial-time 15, 16
- interactive Turing machine (ITM)
 - choosing an element at random 15
- communication tape
 - read-only 15
 - write-only 15
- composition tape 15
- compositional input 15
- compositional output 15
- execution 17
- input 15
- input tape 15
- instance 17
- internal choice 15
- output 15
- output tape 15
- random tape 15
- read-messages 15
- run 17
- view 21
- work tape 15
- write-messages 15

- interest group 70
- intermediate pseudonym 72
- internal choice *see* ITM 15
- internal interface *see* participant 21
- internal interface *see* protocol 21
- intro pseudonym 72
- issue *see* credential scheme 77
- issuer legitimation 7
- issuer *see* legitimation: provider 5

INDEX OF KEYWORDS

ITM
 execution 17
ITM composition
 general 18
 member ITM 18
 represented by an ITM 18
ITM representing a general composition of ITMs 18
ITM *see* Turing machine 15

K

Kerberos 59
key pair
 native 78
knowledge error function 36
knowledge extractor 36

L

legitimation 5
 buy 9
 create 8
 deposit 8
 issuer 7
 k-transfer 9
 licence 11
 negative 11
 owner 5
 pay 8
 provider 5
 issuer 5
 verifier 5
 sell 9
 show 8
 type
 hierarchy 11
 negative 11
 withdraw 8
legitimation problem
 authorization problem
 control center 61
legitimation system 5
licence *see* legitimation 11

M

make (witness makes a candidate) 41, 53
make a candidate 35
make a pseudonym 76
making function
 credential scheme 76
 proof-of-knowledge scheme 35, 41
 restrictive blind signature scheme 53
match (co-witness matches a candidate) 41
Medical outpatient treatment 146
member ITM 18
member protocol 21
message
 signature scheme 53
micro payment scheme 66

mobile user device 5, 67
 cell phone 67
 organizer 67
 palmtop 67
 personal digital assistant 67
 wallet with observer 69
mortgage bonds 9
MUD *see* mobile user device 5, 67
multi-party computation 25

N

native key pair 78
negative legitimation 11
negligible probability 14
non-negligible probability 14
non-triviality 36

O

oblivious authentication 65
observer 59, 69
 biometric 69, 90
 broken 82
 intact 82
one-time blind signature scheme 52, 54
organizer 67
out of band inflow 69
outflow
 in-band 68
 out-of-band 69, 86
out-of-band outflow 69, 86
outpatient treatment
 medical 146
 paramedical 146
output of an ITM 15
output tape of an ITM 15
overshow detection
 credential scheme 84
overshow prevention
 credential scheme 84
overshooting 65
overspending
 detection 65
 prevention 65
overwhelming
 probability 14
owner 6
owner *see* legitimation 5
ownership 6

P

pair of message and signature
 valid 54
palmtop 67
Paramedical outpatient treatment 146
parameter
 typed 26
participant

- attacking 24
 - honest 24
- participant of a protocol 16
- party of a protocol 16
- passive attack 24, 52
- passive attacker 24, 52
- perfect security requirement 25
- perfect witness indistinguishability
 - of protocol 38
- personal agent trust 67
- personal control *see* security requirement 5
- personal digital assistant 67
- pharmaceutical supply 147
- pharmacies and paramedical providers 148
- phone ticks *see* micro payment scheme 66
- pick-and-test algorithm 15, 26
- point-of-sale terminal 67
- polymorphic identifiers 24
- polynomial composition 22
- polynomial-size of initial content 24
- polynomial-time ITM 15, 16
- POS (point-of-sale) terminal 67
- PPT (probabilistic polynomial-time Turing machine) 15
- PPT ITM 15
- prekey 23
 - blind signature scheme 53
 - credential scheme 76
 - proof-of-knowledge scheme 35, 41
 - scheme 23
- prescription authorization (P-authorization) 152
- prescription groups 156
- privacy
 - unlinkability 150
- privacy, patient and physician 150
- private group key 127, 142
 - group credential scheme 142
 - group signature scheme 126
- private individual key 127, 142
- private input parameter *see* protocol 20
- private key
 - credential scheme 76
 - native 78
 - signature scheme 53
- private native key 78
- private output parameter 20
- private output parameter *see* protocol 20
- probabilistic execution function of ITM 17
- probabilistic execution function of protocol 17
- probabilistic output function of protocol 19
- probabilistic polynomial-time (PPT) Turing machine 15
- probabilistic polynomial-time Turing machine (PPT) 15
- probabilistic Turing machine 15
- probabilistic view function of protocol 21
- probability
 - negligible 14
 - non-negligible 14
 - overwhelming 14
- proof of knowledge
 - convincing 35
 - knowledge error function 36
 - witness makes a candidate 41, 53
 - zero-knowledge (ZKP) 62
 - non-interactive 62
- proof of knowledge scheme
 - witness makes a candidate 35
- proof-of-knowledge mechanism
 - witness indistinguishability 38
- proof-of-knowledge scheme 35
 - candidate 35, 41
 - completeness 36
 - divertible
 - computationally 42
 - perfect 42
 - extended
 - co-making function 41
 - co-witness 41
 - extendedness 42
 - witness addition 41
 - making function 35, 41
 - non-triviality 36
 - perfect witness indistinguishability 38
 - prekey 35, 41
 - validity 36
 - witness 35, 41
 - witness indistinguishability 38
- protocol
 - (deterministic) execution function 17, 18
 - attacking 25
 - common input 20
 - common input parameter 20
 - common output 20
 - common output parameter 20
 - communication topology 16
 - computation time 16, 20
 - expected computation time 20
 - external interface 19
 - instance 20
 - interactive 16
 - internal interface 21
 - participant 16
 - composed 22
 - internal interface 21
 - party 16
 - private input parameter 20
 - probabilistic execution function 17
 - probabilistic output function 19
 - probabilistic view function 21
 - round 16
 - selective view function 21
 - shared input 20
 - shared input parameter 20
 - shared output 20
 - shared output parameter 20
 - starlike 16
 - step 16
 - sub-protocol 22
 - view
 - valid 21
 - view of an ITM 21
- protocol composition
 - general 21
 - member protocol 21
 - polynomial-time 22
 - sequential 21
- provider *see* legitimation 5
- proxy-signatures 123

INDEX OF KEYWORDS

- pseudonym
 - credential scheme 76
 - intermediate 72
 - introduction
 - explicit 77, 80, 84
 - implicit 77, 80, 84
 - source 60, 72
 - target 60, 72
 - pseudonym co-witness
 - credential scheme 76
 - pseudonym witness
 - credential scheme 76
 - public group key
 - group signature scheme 126, 142
 - public group key (public key) 127, 142
 - Public health services 147
 - public individual key 127, 142
 - group credential scheme 142
 - group signature scheme 126
 - public key
 - native 78
 - signature scheme 53
 - public key certificate 59
 - public native key 78
 - publicly verifiable encryption 98
- R**
- random oracle 31
 - random tape of an ITM 15
 - random variable 13
 - read-messages of an ITM 15
 - re-blind 63
 - recognition characteristic 80
 - recognize
 - observer identity 78
 - referral authorization (R-authorization) 152
 - referral groups 156
 - registered physicians 147
 - registration *see* healthcare providers 151
 - relative anonymity 66
 - representation 30
 - responsible for a signature *see* group signature scheme 127, 142
 - restrictive blind signature scheme 52
 - round *see* protocol 16
 - run of an ITM 17
- S**
- sample space 13
 - scheme
 - implementation 25
 - instance 25
 - prekey 23
 - proof-of-knowledge 35
 - security parameter 23
 - school reports 8
 - securities 9
 - security module 68
 - security parameter
 - blind signature scheme 53
 - credential scheme 76
 - proof-of-knowledge scheme 35, 41
 - scheme 23
 - security requirement
 - attacker model 24
 - attacking participant 24
 - honest participant 24
 - perfect 25
 - personal control 5
 - success condition 25
 - success measure 25
 - unconditional 25
 - security requirements
 - confidentiality requirements 6
 - integrity requirements 6
 - selective forgery *see* attack goals 37, 52
 - selective view function of protocol 21
 - sequential composition 21
 - shared care 3
 - shared input parameter *see* protocol 20
 - shared input *see* protocol 20
 - shared output parameter *see* protocol 20
 - shared output *see* protocol 20
 - show *see* credential scheme 77, 78
 - show-wise fail-safe unlinkability
 - credential scheme 87
 - signature
 - signature scheme 53
 - signature scheme
 - blind
 - blinder 53
 - group-oriented 67
 - one-time 52, 54
 - prekey 53
 - randomized 65
 - restrictive 52, 65
 - making function 53
 - witness 53
 - witness equivalence 53
 - security parameter 53
 - blind signature 53
 - blindness 54
 - effectiveness 54
 - message 53
 - private key 53
 - public key 53
 - signature 53
 - signer anonymity
 - group signature scheme 128
 - signing groups 123
 - simple event 13
 - source pseudonym 72
 - specialists 148
 - SPM *see* stationary provider machine 67
 - starlike *see* protocol 16
 - static group signature scheme 124
 - stationary provider machine
 - point-of-sale terminal 67
 - stationary provider machine (SPM) 67
 - step *see* protocol 16
 - stocks 9
 - Subgroup Diffie-Hellman Assumption 30

- Subgroup Discrete Log Assumption 29
- Subgroup Representation Assumption 30
- sub-protocol 22
- success condition 25
- success measure 25
- successful
 - execution of a protocol 17

T

- target pseudonym 72
- threshold-signatures 123
- total break *see* attack goals 37, 52
- traceability 60
- transcript
 - credential scheme 76
- transferability 66
- trust model
 - captured agent trust 67
 - personal agent trust 67
 - undercover agent trust 68
- Turing machine
 - flipping a coin 15
 - interactive (ITM) 15
 - probabilistic 15
 - probabilistic polynomial-time (PPT) 15
- type of content
 - non-personal 9
 - personal 10
 - personal bonus/malus 10
 - tradeable 10
- typed parameter 26

U

- unconditional security requirement 25
- undercover agent trust 68
- unforgeability
 - credential scheme 83
- unframeability
 - group signature scheme 128
- universal break *see* attack goals 37, 52
- universal cash 66
- untraceability *see* privacy 150

V

- valid credential 78
- valid pair of message and signature 54
- valid signature 54
- valid view 21
- validity
 - proof-of-knowledge scheme 36
- verifiable proxy encryption 98
- verifier *see* legitimation: provider 5
- view of an ITM *see* protocol 21

W

- wallet 59
- wallet with observer 59, 69
- witness
 - proof-of-knowledge scheme 35, 41
- witness addition
 - extended proof-of-knowledge scheme 41
- witness domain
 - restrictive blind signature scheme 53
- witness equivalence
 - restrictive blind signature scheme 53
- witness indistinguishability 38
 - proof-of-knowledge mechanism 38
 - proof-of-knowledge scheme 38
- work tape of an ITM 15
- write-messages of an ITM 15

Z

- zero-knowledge *see* proof of knowledge 62

INDEX OF KEYWORDS