Privacy-preserving Personal Information Management

Mohamed Abdelhamid Layouni

School of Computer Science
McGill University, Montreal
August 2009

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

"To my dear parents"

Abstract

The spread of Information and Communication Technologies (ICTs) has transformed the way we deliver services, and has made them in general more efficient and more accessible to users. With these improvements however came new challenges. The extensive use of electronic services in our daily life, and the massive gathering of transactional data have led to serious privacy violations.

In this thesis we provide techniques to enhance users' privacy, and to give them greater control over their data. We propose a protocol allowing users to authorize access to their remotely-stored records, according to a self-chosen privacy policy, and without the storage server learning the access pattern to their records, or the index of the queried records. This prevents the storage server from linking the identity of the party retrieving a record to that of the record owner. In many applications, the association between the identity of the record retriever and that of the record owner represents sensitive information, and needs to be kept private. The proposed protocol is called Accredited Symmetrically Private Information Retrieval (ASPIR), and uses Brands's Anonymous Credentials [Bra00] and a Symmetrically Private Information Retrieval (SPIR) scheme by Lipmaa [Lip05], as building blocks.

Next, we extend the above ASPIR protocol to a setting where the stored records belong to multiple owners simultaneously. The new protocol, called Multi-Authorizer ASPIR, allows the owners of a record to authorize access to their data according to a self-chosen privacy policy, without the storage server learning the access pattern to their record. We present constructions for settings where the retrieving party has to provide authorizations either from all the owners of the target record, or from a subset of them of size greater that a certain threshold. We also consider the case of a General Access Structure, where the retrieval is allowed only if authorizations from certain pre-defined subsets of the owners are provided. The Multi-authorizer ASPIR protocol is more efficient than ASPIR, and can be built with any SPIR primitive.

Finally, we dedicate the last part of the thesis to applying privacy preserving techniques to a real world problem. In particular, we consider the area of e-health, and provide a privacy-preserving protocol for handling prescriptions in the Belgian healthcare system.

Résumé

La prolifération des services électroniques a eu des retombées positives sur nos sociétés. Les technologies de l'information ont révolutionné divers domaines clé de notre vie, notamment les services gouvernementaux, les affaires, la santé, les transports, les communications et l'éducation. Souvent, le passage au numérique, a rendu les services plus accessibles, plus rapides, plus faciles à utiliser et socialement plus inclusifs. Cependant, avec ces améliorations sont apparus aussi de nouveaux problèmes. En effet, l'utilisation des services électroniques au quotidien, et la collecte massives de données transactionelles sur les utilisateurs, ont conduit à l'établissement de ce qu'on appelle communément les "dossiers électroniques". Un dossier électronique est une compilation de données personelles récoltées lorsqu'un individu effectue des transactions élect-roniques ou reçoit des services. Ces dossiers sont de plus en plus utilisés par le gouvernement et les corporations pour prendre des décisions importantes sur les individus, sans que ces derniers ne soient capables d'y participer.

Cette thèse présente des techniques pour protéger davantage la vie privée des citoyens et leur donner plus de contrôle sur leurs données. On propose, entre autres, un protocole pour permettre à des utilisateurs d'autoriser l'accés à leurs données, sauvegardées sur un serveur distant, sans que celui-ci n'apprenne d'informations sur la fréquence et la distribution des accés, ou même sur l'indice des données récupérées. Ceci empêche le serveur d'établir des liens entre l'identité d'un propriétaire de données, et celle de l'agent qui a demandé l'accés à ses données. On peut penser à une multitude de scénarios où la divulgation de l'existence d'un tel lien est non souhaitable. Le protocole qu'on propose est nommé ASPIR de l'Anglais (Accredited Symmetrically Private Information Retrieval), et utilise les systèmes de certification de Brands [Bra00], ainsi que le système SPIR de Lipmaa [Lip05].

Dans un deuxième temps, on généralise le protocole ASPIR initial à un environnement où les entrées appartiennent à plusieurs parties. Le nouveau protocole, nommé Multi-Authorizer ASPIR, permet aux propriétaires d'autorizer l'accés à leurs données selon une politique qu'ils ont eux même choisie, et sans que le serveur n'apprenne des informations sur la fréquence et la distribution des accés. On présente des constructions pour des scénarios où le demandeur de données doit fournir une autorisation de la part de tous les (respectivement une partie des) propriétaires. Le protocole, Multi-authorizer ASPIR, est plus performant, et peut être implanté avec n'importe quel système SPIR.

Enfin, la dernière partie de la thèse est dédiée à l'application des techniques de protection de la vie privée à un exemple concret de la vie courante. L'exemple qu'on traite appartient au domaine de la santé. On présente alors un protocole pour gérer les ordonnances, compatible avec le système de santé Belge. Le protocol proposé préserve la vie privée des patients et des médecins.

تَلخِيـــص

لَقد كَانَ لِإنتِشَارِ الخَدَمَاتِ الإِلِكترُونِيَّةِ تَأَثِيرَاتٌ إِيجَابِيَّةٌ كَثِيرَةٌ عَلَى المُجتَمَع، فَقَد إِمتَدَّت تِكنُولُوجِيَا المَعلُومَات وَ الإِتِّصَالَات إِلَى مَجَالَاتٍ عَدِيدَة مِنَ الحَيَاة وَ طَوَّرَتهَا. نَذكُرُ مِنهَا عَلَى سَبِيلِ المِثَال مَجَالَات إِدَارَة الدَّولَة وَ المُعَامَلَات وَ الصِحَّة وَ النَّقل وَ الإِتِّصَالَات وَ التَّعَلِيم. لَقَد تَمَكَّنَت تِكنُولُوجِيَا المَعلُومَات مِن تَطوِيرِ الخَدَمَاتِ فِي مُعَظَمِ هَذِهِ المَجَالَاتِ وَ جَعَلَتهَا فِي أَغلَبِ الأَحيَانِ أَسرَع وَ أَسهَل فِي الإِستِعَمَالِ وَ أَكثَر مُلَاءَمَتًا لِمَبَادِئِ المُسَاوَاتِ الإِجتِمَاعِيَّة. عَلَى الرَغمِ مِن هذِهِ الجَوَانِبِ الإِيجَابِيَّة، خَلَقَت تِكنُولُوجِيَا المَعلُومَات العَدِيدَ مِنَ المَشَاكِل. فَقَد أَدَّى الإِستِعَمَالُ اليَومِي لِلأَجهِزَتِ الإِلِكترُونِيَّة، وَ التَخزِين المُكَثَّف لِلمَعلُومَات الشَخصِيَّة إِلَى ظُهُورِ مَا يُعَبَّرُ عَنهُ بِالمِلَفَّاتِ الشَخصِيَّة. هَذِهِ المِلَفَّاتُ تُستَخدَمُ بِصِفَةٍ مُتَزَايِدَةٍ مِن قِبَلِ الحُكُومَات وَ الشَرِكَات لِإِتِّخَاذِ قَرَارَات مُهِمَّة بِشَأنِ الأَفرَادِ مِن دُونِ أَن يَستَطِيعَ هَؤُلَاءِ أَن يُدلُوا بِرَأيِهِم فِي هَذِهِ القَرَارَات. يَبدُو مِنَ الوَاضِح أَنَّ هَذِهِ المُمَارَسَاتِ لَا تَخدِمُ مَصلَحَة المُواطِنِين.

نُقَدِّمُ فِي هَذَا العَمَل مَجمُوعَةً مِنَ التَقنِيَاتِ لِتَعزِيزِ حِمَايَة المَعلُومَاتِ الشَخصِيَّة لِلمُواطِنِين وَ مَنحِهِم قُدُرَاتٍ أَكبَر لِلتَحَكُّمِ فِي خُصُوصِيَّاتِهِم. مِن بَينِ هَذِهِ التَقنِيَات، نَقتَرِحُ برُوتُوكُول يَسمَحُ لِلمُستَخدِمِين مَنحَ تَرَاخِصٍ إِلَى جِهَةٍ مُعَيَّنَة لِتَمكِينِهَا مِن قِرَاءَة مَعلُومَاتِهِم المُخَزَّنَة عَن بُعد وِفقًا لِسِيَاسَة تَحَكُّم إِختَارُوهَا بِأَنفُسِهِم، وَ مِن دُونِ ان يَحصُلَ الكُمبِيُوتِر المُخَزِّن عَلَى مَعلُومَات عَن نَمَطِ القِرَاءَة أَو عَن هَوِيَّة التَسجِيلَاتِ التِي تَمَّت قِرَاءَتُهَا، أَو حَتَّى هَوِيَّة مَالِكِيهَا. اَلبرُوتُوكُول الَذِي نَقتَرِحُهُ يُسَمَّى بِـ (ASPIR) مِنَ العِبَارَةِ الإِنقلىزِيَّةِ (Accredited Symmetrically Private Information Retrieval) ، وَ يَستَعمِلُ تَصمِيمُهُ مَنضُومَةَ الوَثَائِقِ الشَخصِيَّةِ (Credentials) لِـ Brands [Bra00] وَ مَنضُومَةَ (SPIR) لِـ Lipmaa [Lip05] .

فِي مَرحَلَة ثَانِيَة مِن هَذَا العَمَل، نُعَمِّمُ برُوتُوكُول ASPIR لِيَشمِلَ الحَالَات التِي تَكُونُ فِيهَا التَسجِيلَات المُخَزَّنَة مُتَعَدِّدَة المِلكِيَّة. بِعِبَارَةٍ أُخرَى، اَلبرُوتُوكُول الجَدِيد وَ الَذِي يُسَمَّى بِـ Multi-Authorizer ASPIR يُخَوِّلُ لِمَالِكِي المَعلُومَاتِ بِـأَن يُسنِدُوا تَرَاخِيصًا لِجِهَاتٍ يَختَارُونَهَا، حَتَّى يَتَمَكَّنَ هَؤُلَاءِ مِن قِرَاءَةِ تَسجِيلَاتِهِم المُخَزَّنَةِ عَن بُعد بِـدُونِ أَن يَتَعَرَّفَ الكُمبِيُوتِر المُخَزِّن عَلَى نَمَطِ القِرَاءَةِ أَو عَن هَوِيَّة التَسجِيلَاتِ التِي تَمَّت قِرَاءَتُهَا. نَطرَحُ تَصمِيمَاتٍ تَتَطَلَّبُ مِنَ الجِهَةِ القَارِئَةِ أَن تُثبِتَ أَنَّ بِحَوزَتِهَا تَرخِيصًا مِن كُلِّ مَالِكِي التَسجِيل المَعنِي او مِن مَجمُوعَة مِنهُم حَسبَ الحَالَةِ. اَلبرُوتُوكُول الجَدِيد Multi-Authorizer ASPIR أَسرَعُ مِنَ البرُوتُوكُول ASPIR ، وَ يُمكِنُ تَجسِيدُهُ بِإِستِعمَال أَيِّ مَنضُومَة SPIR .

وَ أَخِيرًا نُخَصِّصُ الجُزءَ الأَخِيرِ مِن هَذِهِ الأُطرُوحَةِ لِـتَطبِيقِ تَقنِيَاتِ الحِفَاظ عَلَى الخُصُوصِيَّاتِ الشَخصِيَّةِ عَلَى مِثَال حَيٍّ. وَ بِـالتَحدِيدِ نَعتَنِي بِمِثَال مَلمُوس مِن مَجَال الصِحَّة. فَنُقَدِّمُ لِهَذَا الغَرَض مَنضُومَتًا لِمُعَالَجَةِ الوَصفَاتِ الطِبِّيَّةِ فِي نِظَام الرِعَايَةِ الصِحِّيَّةِ البِلجِيكِي، بِطَـرِيقَة تُحَافِظ عَلَى الخُصُوصِيَّاتِ الشَخصِيَّةِ لِـلمَرضَى وَ الأَطِـبَّاء.

Acknowledgements

First, I would like to thank Professor Claude Crépeau, my adviser, for his encouragements and support throughout my PhD, and for his many comments on the early versions of this thesis. The text of this thesis has been significantly improved thanks to his comments and observations.

I am also grateful to Dr. Stefan Brands, for his guidance especially in the early stages of my PhD, and for introducing me to two highly stimulating projects: The *ADAPID* project and the *On the Identity Trail* project. I would like to thank him also for the many interesting discussions on the topic of privacy-preserving credentials.

I would like to extend my thanks also to Prof. Hans Vangheluwe for his kindness, encouragements, availability, and for the numerous interesting discussions we have had. I would like to thank him also for generously funding my trips to attend several conferences and events in Canada and abroad.

My friend and former office-mate Carlton Davis has provided invaluable comments on an early version of the thesis manuscript. His comments have helped me strengthen my arguments, and have greatly improved the presentation of this thesis. I really appreciate his help.

In the course this thesis, I have had the pleasure working with many people. Among them: Maki Yoshida and Shingo Okamura from Osaka University; Bart De Decker and Kristof Verslype from the DistriNet Lab at KULeuven; Mehmet Tahir Sandıkkaya from the Katholieke Hogeschool Sint-Lieven; Claudia Diaz, Carmela Tron-

coso and many others from the COSIC Lab at KULeuven; Ximeng Sun and Miriam Zia from the MSDL Lab at McGill. I would also like to thank the COSIC Lab at KULeuven for hosting me during two research visits in the summers of 2006 and 2008.

Many thanks also go to my colleagues and friends at the Crypto and Quantum Info Lab (CQIL), and the School of Computer Science at McGill. Having them around made my time at McGill all the more enjoyable.

Last but not least, I would like to thank my parents, and the rest of my family and friends, for their unconditional support and patience.

This work has been funded in part by IWT-Vlaanderen (The Flemish Institute for Scientific and Technological Research in Belgium) through the *ADAPID* project (Advanced Applications for e-ID cards in Flanders), by SSHRC (The Social Sciences and Humanities Research Council of Canada) through the *On the Identity Trail* project, and by Tunisia's Universitary Mission in North-America through a doctoral scholarship award. Their support is greatly appreciated.

Contents

Abstract iii

Résumé vi

Acknowledgements x

List of figures xix

List of tables xxi

1 Introduction 3

1.1 Organization of the thesis 6

1.2 Notations 8

1.3 Definitions 8

2 Privacy-preserving Credentials 11

2.1 Introduction 11

2.2 Privacy-preserving credentials: A historical timeline 16

2.3 Comparison criteria 27

2.4 Chaum's blind signatures 31

2.4.1 Main idea 31

2.4.2 Setting and Assumptions 32
2.4.3 Credential format 32
2.4.4 Summary of security and privacy properties 33
2.5 Chaum-Pedersen signatures 33
2.5.1 Main idea 33
2.5.2 Setting and Assumptions 34
2.5.3 Credential format 34
2.5.4 Summary of security and privacy properties 35
2.6 Brands credentials 36
2.6.1 Main idea 36
2.6.2 Setting and Assumptions 37
2.6.3 Credential format I 38
2.6.4 Credential format II 39
2.6.5 Summary of security and privacy properties 40
2.7 SRSA-based Camenisch-Lysyanskaya credentials 40
2.7.1 Main idea 40
2.7.2 Setting and Assumptions 41
2.7.3 Credential format 41
2.7.4 Summary of security and privacy properties 43
2.8 DL-based Camenisch-Lysyanskaya credentials 43
2.8.1 Main idea 43
2.8.2 Setting and Assumptions 44
2.8.3 Credential format 45
2.8.4 Summary of security and privacy properties 46
2.9 Discussion 47

3 Accredited Symmetrically Private Information Retrieval (ASPIR) 51
3.1 Introduction 52
3.2 Related work 57
3.3 Building Blocks for the DL-based construction 58
3.3.1 Brands credentials 58
3.3.2 ElGamal homomorphic encryption 62
3.3.3 AIR-based Lipmaa $\binom{n}{1}$-OT 62
3.4 Accredited SPIR based on the DL problem 67
3.5 Security definitions 75
3.6 Security and privacy properties of the ASPIR protocol 80
3.6.1 Definitions 80
3.6.2 Analysis 83
3.6.3 Additional privacy for the Authorizer 87
3.7 Performance analysis 88
3.8 Accredited SPIR based on RSA 89
3.8.1 RSA-based Brands credentials 89
3.8.2 Combining ElGamal with Brands RSA-based Credentials 91
3.8.3 Security and privacy properties 94
3.8.4 Variant based on the Okamoto-Uchiyama cryptosystem 94
3.9 Conclusion 96

4 Efficient Multi-Authorizer ASPIR 99
4.1 Introduction 100
4.2 Preliminaries 103
4.2.1 Pairing-based signature scheme 104
4.2.2 Symmetrically private information retrieval 104

4.3 Protocol description . . . 106
4.3.1 Settings . . . 107
4.3.2 First construction . . . 107
4.3.3 Improved construction . . . 109
4.4 Security and privacy evaluation . . . 111
4.5 Performance analysis . . . 115
4.6 Extension to threshold access . . . 116
4.7 Extension to authorizers with unequal rights . . . 117
4.8 The case of an owner tuple possessing multiple records. . . . 120
4.9 Conclusion . . . 121
4.9.1 Possible extensions . . . 122

5 Privacy in Practice: A Protocol for e-Health 123

5.1 Introduction . . . 124
5.2 Related work . . . 125
5.3 Brief overview on the Belgian healthcare system . . . 128
5.4 Requirements . . . 131
5.4.1 Security requirements . . . 131
5.4.2 Privacy requirements . . . 132
5.5 Building blocks: brief overview . . . 133
5.5.1 Commitments . . . 133
5.5.2 Digital credentials . . . 135
5.5.3 Verifiable encryption . . . 136
5.6 The proposed protocol . . . 137
5.6.1 Setting . . . 137
5.6.2 Protocol description . . . 138

5.7 Protocol evaluation . . . 142
5.7.1 General Security Requirements . . . 143
5.7.2 Security requirements specific to the Belgian healthcare system. 143
5.7.3 Privacy . . . 145
5.8 Enhancing the patient's control over their data . . . 146
5.9 Concluding remarks . . . 147
6 Summary and Conclusion 149
A Credential Systems Comparison: A Compendium 153
B State-of-the-Art Credential Systems: A Detailed Overview 165
B.1 Chaum's blind signatures . . . 165
B.1.1 Summary of security and privacy properties . . . 165
B.2 Protocols for the Chaum-Pedersen credential system . . . 167
B.2.1 Credential issuing . . . 167
B.2.2 Credential showing . . . 169
B.2.3 Summary of security and privacy properties . . . 172
B.3 Protocols for the DL-based Brands credentials (format I and II) . . . 174
B.3.1 Credential issuing protocol I . . . 174
B.3.2 Credential issuing protocol II . . . 177
B.3.3 Credential showing . . . 179
B.3.4 Summary of security and privacy properties . . . 192
B.4 Protocols for the Strong-RSA-based Camenisch-Lysyanskaya credentials 196
B.4.1 CL-SRSA credential issuing . . . 196
B.4.2 CL-SRSA credential showing . . . 196
B.4.3 Summary of security and privacy properties . . . 196

B.5 Protocols for the DL-based Camenisch-Lysyanskaya credentials 200
B.5.1 CL-DL credential issuing 200
B.5.2 CL-DL credential showing 200
B.5.3 Summary of security and privacy properties 202

Bibliography 209

List of publications 223

List of figures

3.1 Br-DL-I Credential Issuing protocol 60

3.2 Br-DL-I basic showing protocol – signed proof of knowledge 61

3.3 Lipmaa $\binom{n}{1}$-OT 65

3.4 Modified version of the Br-DL-I Credential Issuing protocol 69

3.5 Accredited Symmetrically Private Information Retrieval : DL-based Construction 74

3.6 Br-RSA Credential Issuing protocol 91

3.7 Br-RSA basic Credential Showing protocol 92

4.1 Multi-Authorizer ASPIR (Improved Construction) 110

5.1 Overview of the Belgian Healthcare System 129

B.1 Basic Chaum-Pedersen signature scheme 168

B.2 Joint randomness selection 168

B.3 Chaum-Pedersen's Wallet-with-Observer Credential Issuing protocol . 170

B.4 Chaum-Pedersen's Wallet-with-Observer Credential Showing protocol 171

B.5 Brands Credential Issuing protocol I (with hidden attributes) 175

B.6 Signed proof of knowledge: $\mathcal{P} = (2x_1+2x_2+4x_3+11x_4 = 6)$ AND $(x_1+2x_2+3x_3+5x_4 \neq 8)$ 177

B.7 Brands Credential Issuing protocol II (with all attributes known to the issuer) 178

B.8 Brands Basic Credential Showing protocol 180

B.9 Brands One-Show Credential Issuing protocol I – (with hidden attributes) 182

B.10 Signed proof with generic witness: $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$ 184

B.11 Smartcard-assisted signed proof of : $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$ 188

B.12 Smartcard-assisted signed proof with inflow prevention : $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$ 190

B.13 Smartcard-assisted signed proof with inflow and outflow prevention : $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$ . . . 191

B.14 CL-SRSA Credential issuing protocol (with hidden attributes) 197

B.15 CL-SRSA Credential showing protocol 198

B.16 CL-DL Credential issuing protocol 201

B.17 CL-DL Credential showing protocol 203

List of tables

5.1 Access control matrix 134

5.2 Credential material per participant 137

A.1 Comparison Matrix 155

“Privacy is the right to be alone—the most comprehensive of rights, and the right most valued by civilized man.”

— Louis D. Brandeis

Chapter 1

Introduction

With the migration of services from the traditional paper-based world to the electronic world, government agencies and businesses started collecting huge amounts of transaction data about their customers. The transaction data collected is often heavily loaded in personal sensitive information (e.g., health data, consumption habits, financial data.) This phenomenon has been further exacerbated by the marked increase in computational and storage performances, and the sharp drop in technology prices. This reality is nicely described by Daniel Solove in his book *"The Digital Person"* [Sol04]. For instance, he says: "Small details that were once captured in dim memories or fading scraps of paper are now preserved forever in the digital minds of computers, in vast databases with fertile fields of personal data. Our wallets are stuffed with ATM cards, calling cards, frequent shopper cards, and credit cards — all of which can be used to record where we are and what we do. Every day, rivulets of information stream into electric brains to be sifted, sorted, rearranged, and combined in hundreds of different ways. Digital technology enables the preservation of the minutia of our everyday comings and goings, of our likes and dislikes, of who we are and what we own. It is ever more possible to create an electronic collage that covers much of a persons life — a life captured in records, a digital person composed in the collective computer networks of the world."

The fact that "digital dossiers" are being constructed about individual citizens is due in part to the spread of electronic services and the ubiquity of their use, but also, and more importantly to the spread of *unfair* data collection practices and the lack of regulations. For example, service providers routinely collect data about their customers, presumably to better serve them, but often a significant part of the data collected is not essential to the service being offered, or to the completion of the transactions it was presumably collected for. Collecting such non-essential data can be seen as an invasion of privacy, and storing it exposes the customer to further unnecessary risks. Both practises have long been denounced by numerous privacy advocates and groups (e.g., [Cha85, CE86, BCC88, Bra00, Rot01, Sol04].) The excessive collection of personal data can be prevented both by legislation, and by technological solutions that allow individual customers to control the amount of information they reveal in the first place. One of the main technologies giving users flexibility and control over the way they disclose personal information, is the so-called privacy-preserving credentials [Cha82, Cha85, Cha87, CFN88, Dam88, CP92, Bra00, Che95, LRSW00, CL02b, CL04, PV04]. Briefly put, this technology allows an individual to have access to a service without the service provider ever identifying him, but with the assurance that the individual has the right credentials to receive the service. More precisely, a certification authority issues to the individual a credential containing a number of attributes about his identity. The individual can then selectively disclose information or properties about his attributes to a verifier or a service provider, without revealing his actual identity. As a direct result of the above features, an individual using privacy preserving credentials can access a service without his accesses being traceable to his identity. In some specific constructions, the system is such that different accesses by the same user are not even linkable to each other.

This thesis builds upon previous privacy-preserving credential systems [Bra00, CL02b, CL04], and provides new ways to protect the privacy of individuals in light of the profound technological developments we witness today.

The invention of privacy-preserving credentials — a technology that minimizes personal data disclosure — is a first and important step towards reinforcing the privacy of individuals. This technology however was not designed to address some of the information flows we see today. In particular, privacy-preserving credential systems were intended to allow a credential holder to protect access *only* to personal information stored on a personal computer or carried on him on a portable device for example. The technology does not help however with protecting personal information stored on a remote server lying outside the user's control.

In this thesis we provide a technique for protecting access to remotely-stored user data. The technique is called *Accredited Symmetrically Private Information Retrieval* (ASPIR). It allows a data subject to decide when, how, and by whom his data can be accessed, without the database manager learning anything about the identity of the data subject, at the time the data is retrieved. The proposed solution combines symmetrically private information retrieval (SPIR) protocols [Lip05] and privacy-preserving digital credentials [Bra00].

We further improved the ASPIR technique and extended it to a setting where each record in the database is co-owned by a set of parties $(A_1, \cdots, A_n)$. The new protocol is called Multi-Authorizer ASPIR. We provide constructions that allow a Receiver party to retrieve a DB record only if he has authorizations from all owners of the target record (respectively, from a subset of the record owners of size greater than a threshold.) We also present constructions for a number of variant configurations; including configurations where the owners of the same record have unequal ownership rights, or where a tuple of owners has multiple database records. The Multi-Authorizer ASPIR solution we propose is more efficient than the basic ASPIR technique mentioned above, and it can be constructed with any SPIR primitive.

Finally, we applied privacy-preserving credential systems to a real-world setting: the Belgian Healthcare System. Real world healthcare systems are generally large and overly complex systems. Designing privacy-friendly protocols for such systems is a challenging task. We present a privacy-preserving protocol for the Belgian healthcare system. The proposed protocol protects the patients' privacy throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practise. The presented protocol relies on standard privacy-preserving credential systems, and verifiable public key cryptography, which makes it readily fit for implementation.

1.1 Organization of the thesis

We start in Chapter 2 by giving a brief survey of the major privacy-preserving credential systems available in the literature. The survey defines what a privacy-preserving credential system means. It also draws a historical timeline showing the sequence of contributions that helped advance the field of privacy as a whole. We highlight the main similarities and differences between various credential systems, and provide a comparison of the most representative ones. A summary of this comparison and details of the studied credential systems are provided in appendices A and B. We then describe, in Chapter 3, our Accredited Symmetrically Private Information Retrieval

(ASPIR) technique for protecting remotely-stored user data. In Chapter 4, we present an extension of the ASPIR technique to the multi-authorizer setting. In Chapter 5, we describe a privacy-preserving protocol compliant with the Belgian healthcare system. Finally, we provide a summary and conclusion in Chapter 6.

Remark on terminology. Throughout this thesis, we often use the term *proof* to mean a zero-knowledge proof. For example, when we say that a party A *proves* a statement, a predicate about a set of attributes, or the knowledge of a secret, we mean that A proves the above in zero knowledge. Briefly, a zero-knowledge proof [GMR85, GMW86] is an interactive protocol that allows one party (the Prover) to convince another (the Verifier) that a certain agreed-upon statement is true, without revealing anything other than the veracity of the statement. We also use the term *signed proof* to mean a signed proof of knowledge. Informally, a signed proof of knowledge is a zero-knowledge proof of knowledge [BG92] that has been made non-interactive. We do also use the term *proof* to mean a conventional mathematical proof. This distinction will be indicated if not already clear from the context.

Finally, we use the term *efficient* to mean probabilistic polynomial time, and the term *negligible* to refer to a function $\epsilon(n)$ that is smaller than $1/n^\alpha$ for all $\alpha > 0$ and sufficiently large n.

1.2 Notations

Algorithms

Let $\mathcal{A}(\cdot)$ be an algorithm, and let $\mathcal{A}(x)$ denote the probability distribution of $\mathcal{A}$'s output for a specific value x of the input. If $\mathcal{A}(\cdot)$ is deterministic then the distribution of the output is concentrated in one point.

Sampling

We denote by $x \leftarrow \mathcal{S}$ the experiment of sampling an element x from a probability distribution $\mathcal{S}$. If F is a finite set, then $x \leftarrow F$ is the experiment of sampling uniformly from the set F. In a similar fashion, the expression $(y, z) \leftarrow \mathcal{A}(x)$ simply means that the pair (y, z) is sampled uniformly from $\mathcal{A}(x)$.

Probability

Let $\mathcal{X}$ be a random variable. The expression *Prob*$_{\mathcal{X}}[x]$ denotes the probability that random variable $\mathcal{X}$ takes a specific value x. The same can be expressed using the equivalent notation *Prob*$[\mathcal{X} = x]$. More generally, for a Boolean predicate $\mathcal{P}(.)$, the expression Prob$_{\mathcal{X}}[\mathcal{P}(x)]$ denotes the probability that $\mathcal{P}(x)$ is *true* after x is sampled uniformly from the distribution of $\mathcal{X}$.

1.3 Definitions

In the following, we provide a number of definitions that will be needed in the remainder of the thesis. Our definitions follow the notations in [Bon98].

Group families. A group family $\mathbb{G} = \{G_{\mathcal{I}}\}$ is a set of finite cyclic groups $G_{\mathcal{I}}$, where $\mathcal{I}$ ranges over an infinite index set. The index $\mathcal{I}$ encodes the parameters of a specific element $G_{\mathcal{I}}$ of the group family $\mathbb{G}$. We denote by $|G_{\mathcal{I}}|$ the order of $G_{\mathcal{I}}$. The following is an example of group families. Let p and q be two primes such that $p = 2q + 1$, and let G_q be the subgroup of $\mathbb{Z}_p^*$ of order q. The subgroup G_q is a finite cyclic group, and the family $\mathbb{G} = \{G_{\mathcal{I}}\} := \{G_q\}$, parametrised by $\mathcal{I} = \{q, p = 2q + 1\}$, is a group family.

Instance generator. A instance generator $\mathcal{IG}$ for a group family $\mathbb{G}$, is a randomized algorithm which on input a (unary) parameter 1^n, runs in polynomial time in n, and returns a random index $\mathcal{I}$ and a generator of g of $G_{\mathcal{I}}$. Note that for each n, the instance generator $\mathcal{IG}$ induces a distribution on the set of indices $\mathcal{I}$.

Definition 1.3.1 (Discrete Logarithm Assumption (DL))**.** *Let $\mathbb{G}$ be a group family. A DL algorithm $\mathcal{A}$ for $\mathbb{G}$ is a probabilistic polynomial time (in $|\mathcal{I}|$) algorithm satisfying, for some $\alpha > 0$ and sufficiently large n :*

$$\mathit{Prob}[\mathcal{A}(\mathcal{I}, g, g^a) = a] > 1/\alpha^n$$

where g is a generator of $G_{\mathcal{I}}$. The probability is over the random choices of $(\mathcal{I}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choices of a in the range $[1, |G_{\mathcal{I}}|]$, and the random bits used by $\mathcal{A}$. The group family $\mathbb{G}$ satisfies the DL assumption if there is no DL algorithm for $\mathbb{G}$.

Definition 1.3.2 (Computational Diffie-Hellman Assumption (CDH))**.** *Let $\mathbb{G}$ be a group family. A CDH algorithm $\mathcal{A}$ for $\mathbb{G}$ is a probabilistic polynomial time (in $|\mathcal{I}|$) algorithm satisfying, for some $\alpha > 0$ and sufficiently large n :*

$$\mathit{Prob}[\mathcal{A}(\mathcal{I}, g, g^a, g^b) = g^{ab}] > 1/\alpha^n$$

where g is a generator of $G_{\mathcal{I}}$. The probability is over the random choices of $(\mathcal{I}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choices of a, b in the range $[1, |G_{\mathcal{I}}|]$, and the random bits used by $\mathcal{A}$. The group family $\mathbb{G}$ satisfies the CDH assumption if there is no CDH algorithm for $\mathbb{G}$.

Definition 1.3.3 (Decision Diffie-Hellman Assumption (DDH))**.** *Let $\mathbb{G}$ be a group family. A DDH algorithm $\mathcal{A}$ for $\mathbb{G}$ is a probabilistic polynomial time (in $|\mathcal{I}|$) algorithm satisfying, for some $\alpha > 0$ and sufficiently large n :*

$$\left| Prob[\mathcal{A}(\mathcal{I}, g, g^a, g^b, g^{ab}) = true] - Prob[\mathcal{A}(\mathcal{I}, g, g^a, g^b, g^c) = true] \right| > 1/\alpha^n$$

where g is a generator of $G_{\mathcal{I}}$. The probability is over the random choices of $(\mathcal{I}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choices of a, b, c in the range $[1, |G_{\mathcal{I}}|]$, and the random bits used by $\mathcal{A}$. The group family $\mathbb{G}$ satisfies the DDH assumption if there is no DDH algorithm for $\mathbb{G}$.

Chapter 2

Privacy-preserving Credentials

With the growing need for personal data protection, privacy-preserving credentials have become one of the most promising technologies capable of ensuring a fair and secure access control management, both for users and organizations. We give an overview of the state of the art in privacy-preserving credentials, and compare some of the most representative privacy-preserving credential systems known in the literature.

2.1 Introduction

In a time of ubiquitous computing and pervasive electronic surveillance, the need for new access control mechanisms that respect the privacy of users has become a pressing reality. Traditionally, users have entrusted central organizations, such as service providers, government agencies and employers, with managing access to their personal data. This approach is not convenient anymore from the users' point of view because it puts central organizations in a position where they can monitor users' interactions without restraint, and learn more information about users than needed. In addition, users have no means to know if the data held about them by the above central organizations is being communicated or sold to third parties. This clearly puts users in a precarious position.

Over time, researchers have developed a variety of mechanisms and measures to empower individual citizens and let them be in charge of protecting, and managing access to, their own personal information. These mechanisms are commonly called privacy-preserving protocols. Example of such mechanisms can be found in [Cha82, Cha85, Cha87, CFN88, Dam88, CP92, Bra00, Che95, LRSW00, CL02b, CL04, PV04].

Informally, a privacy-preserving protocol is one that satisfies the following properties:

- Each party is able to selectively disclose any partial information (e.g., a Boolean predicate) about his/her identity. In other words, no party should be able to learn any information about another party's identity, beyond what the latter has wilfully disclosed.
- No party can forge a false identity.
- No party can convince others of a false property about his/her identity.
- Any party that engages in an abusive behaviour, can be provably detected, and later deanonymized. Ideally the deanonymization should not rely on a trusted third party.

The aim of this chapter is to discuss privacy-preserving digital credentials — a major building block in the design of privacy-preserving protocols. Privacy-preserving digital credentials have been used in several application domains ranging from electronic payment systems (e.g., [CFN88, Dam88, OO89, Bra93, Bra95a, Ped96, FTY98, CHL06]), to e-voting (e.g., [CFSY96]), to access control (e.g., [BCC04, FAL04]) and digital rights management (e.g., [CPJ04].) In addition to introducing the notion of privacy-preserving credentials, we try to draw a historical timeline that, we hope, will give the reader an idea about the sequence of contributions that helped advance the field of privacy. At each point of the timeline, we give a brief summary describing the novelty brought by the contribution in question, if applicable, and how it relates to previous work. Finally, we compare some of the most representative credential systems among the ones described in the timeline. The aim of this comparison is to give researchers and practitioners in fields such as access control and identity management, an introduction to privacy-preserving credentials that can hopefully assist them in making design choices.

As in any branch of science, new constructions often reuse legacy techniques from the literature. Describing all of these constructions in detail in one document will lead to significant redundancy. Therefore, we would like to note that in order to avoid redundancy, we *do not* describe in detail *every single* work in the field of credentials. Instead we focus on the most representative contributions, especially those that came up with new paradigms or techniques.

Terminology

In the following we introduce the notion of *credentials*, as well as the different parties involved in a credential system and their roles. We then describe the sequence of operations performed by each party in a typical workflow where privacy-preserving credentials are used.

Digital credentials

A digital credential can be thought of as a set of assertions made by a certain issuer about the identity attributes of a user. Similar to their paper-based counterparts, digital credentials need to fulfill a certain number of security requirements such as unforgeability, and integrity. The X.509 public key certificate standard [IT05] is a well known example of digital credentials.

A more elaborate type of digital credentials, is the so-called privacy-preserving digital credentials, which are the focus here. In addition to the usual security properties required for conventional digital credentials (e.g., X.509 certificates), privacy-preserving credentials require a number of properties intended to protect the identity of honest credential holders. Among these, we note properties such as *Selective disclosure*, *Token untraceability*, *Tokens unlinkability*, *Multi-show unlinkability*, *Limited-show untraceability*, *Signed audit trails* etc. Detailed definitions of these properties are given in Section 2.3.

Parties involved and their interactions

We distinguish three types of participants in a credential system:

(1) An *issuer*, generally a recognized certification authority, who issues credentials to users. The issuer is also referred to as the *CA* in this document.

(2) A *user* to whom credentials are issued. The user — also referred to as the credential holder — shows his credentials to third parties in exchange for goods and services.

(3) A *verifier* to whom the user shows his credential.

The interaction through which an issuer issues a credential to a user is called an *issuing protocol*. After receiving a credential from an issuer, the user may show his credential to a verifier. The credential *showing* requires proving that the credential itself is valid, and that it belongs to the user. The showing may also include proving that the attributes embedded in the credential satisfy a certain predicate agreed upon by the user and the verifier.

The interaction through which a user shows his credential to a verifier is called a *showing protocol*. The messages exchanged in the issuing protocol between the user and the issuer, and in the showing protocol between the user and the verifier are referred to as the issuing and showing transcripts, respectively.

Optionally, the verifier may later *deposit* the showing transcript at the credential issuer, for instance to get paid for the provided services. The latter protocol is called a *depositing protocol*.

Credential handling and the Wallet-with-Observer setting

There is a wide spectrum of configurations for handling credentials. Each configuration is determined by the amount of control the user and the certification authority have on the user's credentials.

On one end of the spectrum, we find a configuration where the user is in full control of his credentials (i.e., he stores them locally and uses a personal computer to show them to verifiers.) Since the user is in full control, he may engage in fraud, for example, by showing his credentials more than the allowed number of times. In such a setting, fraudulent behaviour is detectable after the fact, but cannot be prevented.

On the other end of the spectrum, we find a setting where the issuer is in control of the whole credential life-cycle. That is, the user is supplied with a tamper-resistant smart-card manufactured by the issuer and containing a number of user credentials. Since the manufacturing of the card and the embedding of credentials is controlled by the issuer, the user might be at risk of having all of his transactions tracked down. For instance, the issuer could embed a secret serial number in some of the user's credentials and use it to recognize his transactions. A malicious issuer could also program the smart-card to record the user's interactions in a hidden log file that could be later retrieved. Moreover, when using a card fully controlled by the issuer to show his credentials, the user does not know if the card is using some covert channels to reveal more information about his identity than he is willing to allow.

To take advantage of the best of the two ends of the spectrum, Chaum and Pedersen proposed a new paradigm [CP92] called the "Wallet-with-Observer" setting. The idea behind this paradigm is to allow the user to be in charge of managing his credentials, while preventing credential misuse. This is done through the adoption of a special electronic wallet containing an issuer-built tamper-proof module, called *Observer*. The wallet is such that the cooperation of the observer and a user-controlled computing module is necessary for any transaction to succeed. Since the wallet's observer is tamper-proof, only transactions complying with the *legal usage rules*, initially fixed by the issuer, are supposed to take place.

2.2 Privacy-preserving credentials: A historical timeline

The area of privacy-preserving credentials has been an intensely active area of research since the 80's, and has resulted in *numerous* elegant and brilliant contributions. Given the extent and richness of this area we do not claim to exhaustively describe every single contribution that was made. Rather we attempt here to survey the state of the art in privacy-preserving credential systems, and highlight some of the key systems that marked the development of this area. We give a detailed discussion of Chaum's seminal work on blind signatures [Cha82, Cha87, CFN88], the Chaum and

Pedersen signatures [CP92], two variants of Brands' discrete logarithm-based credentials [Bra00], and two credential systems proposed by Camenisch and Lysyanskaya [CL02b, CL04]. We also provide a brief description of some key intermediary credential systems, such as those of Damgård [Dam88], Chen [Che95], Verheul [Ver01], and Visconti and Persioano [PV04], and indicate, in each case, how they relate to the other systems, and how they contributed to advancing the field. In the following, we briefly introduce each of the above systems.

Chaum's blind signatures

The notion of blind signatures has been invented by Chaum [Cha82], and later extended in [Cha87, CFN88]. This notion has been highly influential, and many researchers in the field agree that it was seminal in the development of privacy as a discipline. The main goal of blind signatures is to allow a user to receive a signature on a message of his choice, without the signer learning anything about either the message to be signed, or the obtained signature. Blind signatures can be useful in many applications where the privacy of users needs to be protected. For instance, in the context of anonymous authentication, a user can obtain a digital credential in the form a digital signature on a self-chosen pseudonym. Let t denote the user's pseudonym, and $\sigma_{\mathrm{CA}}(t)$ the certification authority's signature on it. To authenticate himself, the user shows the pair $(t, \sigma_{\mathrm{CA}}(t))$ to a service provider. Later the service provider may deposit the pair $(t, \sigma_{\mathrm{CA}}(t))$ at the CA, for instance to get paid for his service. Since $(t, \sigma_{\mathrm{CA}}(t))$ has been obtained through a blind signature mechanism, the CA is unable to link the deposited pseudonym-signature pair to the instance of the issuing protocol that generated it, and thus to the identity of the user.

It is worth noting that credentials obtained through Chaum's blind signatures are *not* universal; each credential is issued for a specific purpose (determined by the signature's verification key), and can only be used for that purpose.

Chaum-Pedersen signatures

In [CP92], Chaum and Pedersen proposed a credential system based on blind signatures and the wallet-with-observer paradigm. The wallet in [CP92] consists of two modules: (1) a tamper-proof module, called *observer*, that cannot be probed or controlled by any third party, and (2) a user-controlled computer, fully controlled by the user of the wallet. To send certified information to the outside world, the user's computer needs the help of the wallet's observer who holds a secret cryptographic key. The user-controlled computer, on the other hand, inspects all flows of data between the wallet's observer and the outside world, and filters out flows of data deviating from the protocol specification.

The wallet allows a user to obtain a pseudonym from the issuer along with a validity proof, called "validation information". The pair consisting of the pseudonym and validation information represents the user's credential. The user can show his credential both in an interactive protocol, or non-interactively by means of a signature. The verifier will not be able to link the showing of a credential to the instance of the issuing protocol that generated it, even if it colludes with the issuer. A verifier is capable, on the other hand, of linking different showings of the same credential to each other.

Similar to Chaum's blind signatures, credentials obtained through the Chaum-Pedersen signature mechanism are not universal.

Damgård credentials

In [Dam88] Damgård constructs a credential scheme based on general multiparty computations and bit commitments. The proposed scheme relies on a trusted center to make sure that each user holds at most one pseudonym with each organization. The system in [Dam88] allows users to obtain credentials in a blind way, and to show them unlinkably multiple times. The identify of credential holders is information-theoretically protected. The system in [Dam88] is mainly a proof of feasibility; more practical constructions inspired from Damgård's system were given later (cf. [Che95, LRSW00].)

Brands credentials

In [Bra95b, Bra97, Bra00] and earlier papers, Brands presents a credential system with a number of features. It provides (1) credential certificates that allow for the encoding of multiple user attributes in the same token. This feature makes the credentials universal, i.e., usable in any application context; the credential holder just needs to prove that his attributes satisfy a certain predicate determined by the context in question, (2) a framework for proving a class of linear predicates on the attributes, and (3) a credential issuing procedure based on the notion of restrictive blind signatures [Bra00, Chap 4]. Restrictive blind signatures is a primitive allowing a certification authority to issue a credential to a user U, such that U can contribute some of the attributes embedded in the credential. The issuing is such that the user-chosen attributes need not be known to the CA, but have to satisfy a certain "well-formedness" property dictated by the CA. This property is called a "blinding invariant". To make sure it does not certify incorrect (user-chosen) attributes, the CA may require the user to prove certain additional properties on those attributes. The user can prove those properties without revealing any extra information about his attributes, beyond the veracity of the assertion being proved. The credential itself consists of (1) a public key encoding the user attributes and (2) a special CA-supplied signature on it. At the end of the issuing protocol, the credential that the user has obtained is perfectly hidden from the issuer.

Later user U can show his credential to a verifier. Showing a credential does not necessarily require the revealing of the attributes encoded in it. A user can selectively and verifiably disclose any information he wishes about his attributes, which may include revealing the actual values of all or some of the attributes, or just proving a predicate on them. In [Bra00], Brands shows how to make proofs for a class of linear predicates.

At a later stage, and depending on the application, the verifier may want to deposit the credential showing transcript at the issuer. This step could be thought of as a cheque deposit in the context of e-banking. The deposited transcript is unlinkable to the instance of the issuing protocol that generated the credential. Although the transcript deposit is done offline, the issuer is capable of detecting instances of double-spending, and identifying the culprits.

Brands [Bra00] proposed two credential systems: the first is based on the discrete logarithm representation problem (DL-REP), and the second on the RSA representation problem (RSA-REP). For the purpose of this survey, we focus on the former, and describe two constructions [Bra00, Sections 4.4.2 and 4.5.2].

Chen's pseudonyms

In [Che95], Chen presented a discrete-logarithm-based system where a user can register a self-chosen pseudonym with an organization and obtain a credential certificate on it. Let U be a user known to organizations O_1 and O_2 under pseudonyms $P_{u,1}$ and

$P_{u,2}$ respectively. Let $Cred_{O_1}(P_{u,1})$ denote a credential certificate issued by O_1 to U on pseudonym $P_{u,1}$.

The main innovation in Chen's system is a *credential transfer procedure* that allows U to transform the initially obtained credential $Cred_{O_1}(P_{u,1})$ into a credential under his second pseudonym $P_{u,2}$. The transfer procedure requires help from the issuing organization O_1 and is conducted in a blind way; i.e., O_1 does not learn $P_{u,2}$ nor the resulting credential certificate on $P_{u,2}$. In the basic scheme [Che95], the transfer procedure is performed online and requires the availability of the issuing organization O_1 whenever a transfer is needed. Chen succeeded in making the transfer offline, but the new method requires the user to obtain, in advance, an additional "blank" copy of the original certificate for every planned transfer. If the same "blank" copy is used for transfers with two organization O_j and O_k, the latter will be able to link the user's pseudonyms $P_{u,j}$ and $P_{u,k}$. In that sense, Chen's certificates are considered as one-time show credentials; each certificate can be shown only once without the showings being linked. It is also worth noting that Chen's certificates do not encode any attributes in them.

Chen's system relies on a trusted center to validate the users' pseudonyms. The center's role is to make sure that the pseudonyms of a user are well formed; i.e., they all have the user's master secret key encoded in them. The trusted center does not participate in any of the subsequent credential procedures (issuing, showing, and transfer.) He is assumed however to behave honestly. For instance, the center is trusted not to impersonate users, create fake pseudonyms for himself, or collude with dishonest users and create multiple identities (master key pairs) for them. It is also worth noting that Chen's system does not provide any measures to discourage users from sharing their pseudonyms or credential certificates.

The LRSW pseudonym system

In [LRSW00], Lysyanskaya, Rivest, Sahai, and Wolf presented a pseudonym system, where users are able to register pseudonyms with organizations, and obtain credential certificates under their pseudonyms. Users can later show their credentials both to the issuing organization, or to a different one, in which case the showing is called credential transfer. The authors of [LRSW00] presented two main constructions. The first is mainly theoretical, and is based on the existence of trapdoor one-way functions [Gol01] (more precisely secure public key encryption, commitment schemes, and zero-knowledge proofs). The credentials can be shown and transferred unlinkably multiple times. The second construction is more practical, and describes a one-time show credential system. The security of this construction is based on the hardness of the discrete logarithm problem, and a related, but less common computational problem (see [LRSW00].)

Similar to Chen's system, the construction in [LRSW00] relies on a trusted third party (TTP) whose role is to make sure that each user has a unique master key pair (all of a user's pseudonyms are well-formed and have the same master secret key encoded in them.) The user obtains a credential certificate in a blind way, and can show it anonymously once, either to the issuing organization or other organizations. The authors do also sketch a variant system where a credential certificate can be shown multiple times unlinkably, but this new construction requires the cooperation of the issuing organization for every additional showing.

Similar to Chen's system, it is assumed that the TTP will not participate in the credential operations. However, unlike Chen's system, the work in [LRSW00] does not require the assumption that the TTP will refrain from impersonating users. Therefore, it can be stated that the LRSW system makes weaker trust assumptions than Chen's.

Verheul's self-blindable credential certificates

In [Ver01], Verheul introduces the notion of self-blindable credential certificates. The certificates in [Ver01] are of the form

$$\begin{aligned} P_u \quad || \quad & \mathit{Sig}_{\mathit{NymCA}}(P_u) \,||\, \mathit{Cert}(\mathit{PK}_{\mathit{NymCA}}, \text{“NymCA statement”}) \\ || \quad & \mathit{Sig}_{\mathit{CredCA}}(P_u) \,||\, \mathit{Cert}(\mathit{PK}_{\mathit{CredCA}}, \text{“Cred attribute”}) \end{aligned}$$

where P_u denotes the user's public key (or pseudonym), and *NymCA* and *CredCA* denote the pseudonym and credential certification authorities respectively. The notations $\mathit{Sig}_{\mathit{NymCA}}(P_u)$ and $\mathit{Sig}_{\mathit{CredCA}}(P_u)$ denote signatures on P_u, by the pseudonym and credential authorities respectively. The term $\mathit{Cert}(\mathit{PK}_{CA}, \text{“statement”})$ denotes a conventional public-key certificate on the public key of certification authority *CA*, while “statement” represents a cleartext description of a property satisfied by the holders of a signature from *CA*.

One of the main features of Verheul's scheme is that a user holding a credential of the form:

$$P_u || \mathit{Sig}_{\mathit{NymCA}}(P_u) \ldots || \mathit{Sig}_{\mathit{CredCA}}(P_u) \ldots,$$

can generate a new blinded version $P'_u || \mathit{Sig}_{\mathit{NymCA}}(P'_u) \ldots || \mathit{Sig}_{\mathit{CredCA}}(P'_u) \ldots$, of *the same* credential without help from any of the issuing *CAs*. Here P'_u denotes a user-blinded version of the initial pseudonym P_u. The two credentials are unlinkable to each other, and as a result the credential certificates can be shown multiple times without the showings being linked to each other. The credential transformation process above, also means that Verheul's scheme allows a user to aggregate multiple credential certificates, possibly issued by different *CAs* to different public keys $P_{u_1}, \cdots, P_{u_n}$, into a single large credential certificate with a public key P_u.

One of the main shortcomings of Verheul's credentials, on the other hand, is that the attributes (denoted by the term "statement" above) are encoded in cleartext as in a conventional X.509 public key certificate. This means that unlike the schemes of Brands (and Camenisch-Lysyanskaya, which will be presented shortly), Verheul's self-blindable certificates do not offer the selective disclosure feature, which allows a credential holder to prove predicates about his attributes without revealing them.

Finally, we note that Verheul suggested an idea for improving the efficiency of credential showings. The idea is to build credential systems in a bilinear group setting, where the DDH problem is easy, while the CDH and DL problems are hard. This choice improves the showing efficiency, since the validity of Schnorr-like signatures (e.g., [CP92, Bra00]) is easy to check for anyone knowing the signer's public key. This means that the credential holder does not have to send a validity proof for his certificate with each showing. The validity of the certificate can be easily verified using the publicly available signature verification key of the *CA*. The idea of using bilinear groups to improve protocol performance has been also seen in the context of identity-based encryption (e.g., [BF01]).

Techniques from [Che95, LRSW00, Ver01] have been reused and further expanded in the more elaborate systems of [CL02b, CL04]. To avoid redundancy, we describe all those techniques simultaneously when we introduce the Camenisch-Lysyanskaya systems [CL02b, CL04].

SRSA-based Camenisch-Lysyanskaya credentials

In [CL02b], Camenisch and Lysyanskaya proposed a credential system based on a new paradigm, where the user's credential is never revealed to verifiers. Instead the user only proves possession of a valid credential from a certain issuer. By doing so, the user can show his credential indefinitely-many times without the showings being linked to each other.[1] The system in [CL02b] relies on an escrow party however to revoke the user anonymity in case of abusive behaviour.

The system initially presented in [CL02b] did not allow for the encoding of multiple attributes, and therefore lacked the *universality* feature. Later the same system was generalized in [BCL04] to enable the encoding of multiple attributes.

The system proposed in [CL02b] is based on the Strong RSA assumption (SRSA), and is denoted by CL-SRSA in the rest of this document. The CL-SRSA system allows the encoding of multiple attributes, a subset of which can be chosen by the user, and hidden from the issuer, while the remaining attributes are known both to the issuer and the user. The CL-SRSA system allows a user to selectively disclose information about his attributes. For instance, a user can prove that the attributes underlying two different credentials (possibly issued by two different issuers) are consistent with one another.

Persiano-Visconti credentials

In [PV04], Persiano and Visconti propose a credential system *relatively similar* to the Brands system, except that the credentials in their system can be shown multiple times without the showings being linked to each other. To enable the multi-show feature, Persiano and Visconti use a similar approach to the one in [Ver01, CL02b], whereby the user (1) computes, for each showing, a new blinded version of his credential, (2) commits to the blinded credential, and (3) sends the resulting commitment to the verifier, along with a zero-knowledge proof that the latter commitment is well-formed and that the underlying credential is valid.

[1] This is similar in spirit to the credential transfer and pseudonym self-blinding procedures of [Che95, LRSW00, Ver01]

It is worth noting however that the Persiano-Visconti system achieves the above using a *special* computational assumption (see [PV04].)

Because the Persiano-Visconti credential system combines ideas from the Brands and Camenisch-Lysyanskaya systems, we do not discuss its details any further.

DL-based Camenisch-Lysyanskaya credentials

Camenisch and Lysyanskaya proposed in [CL04] a credential system based on the discrete logarithm assumption, in the context of bilinear groups. The DL-based version of Camenisch-Lysyanskaya credentials, denoted CL-DL, allows the embedding of multiple attributes in the same credential. A subset of the encoded attributes can be chosen by the user, and hidden from the issuer, while the remaining attributes are known both to the issuer and the user. The CL-DL system allows a user to selectively disclose information about his attributes. For instance, a user can prove that the attributes underlying two different credentials (possibly issued by two different issuers) are consistent with one another.

As in the CL-SRSA system, the user's credential is never revealed to verifiers. The user only proves possession of a valid credential from a certain issuer. By doing so, the user can show his credential multiple times without the showings being linked to each other. The CL-DL credential system relies on an escrow party to revoke the user anonymity in case of abusive behaviour.

Chapter organization

First, we start in Section 2.3 by defining a number of criteria that will be used to compare the different credential systems we survey here. In Section 2.4, we describe Chaum's seminal work on blind signatures [Cha82, Cha87, CFN88]. We then discuss the Chaum-Pedersen signatures [CP92] in Section 2.5. In Section 2.6, we examine two of Brands' DL-based credentials [Bra00]. In Sections 2.7 and 2.8, we describe the two credential systems proposed by Camenisch and Lysyanskaya: the Strong RSA-based CL-SRSA [CL02b] and the discrete logarithm-based CL-DL [CL04]. Table A.1 in appendix A summarizes our comparison of the six credential systems mentioned above. In Section 2.9, we highlight the main differences between the studied systems. We also provide in the appendix a more detailed description of the protocols associated with each system.

2.3 Comparison criteria

This section provides short definitions for some of the comparison criteria considered in this survey, and used in Table A.1. Comparison criteria with obvious definitions are left out.

Token type. Three types of tokens can be found in the literature:

1. **Signed messages** : A token of this type is simply a user-chosen number signed by the issuer. The early systems proposed by Chaum [Cha87] and Chaum, Fiat, and Naor [CFN88] are examples of systems with such tokens.
2. **Public-key certificates** : They consist of two parts; a public key and a issuer-supplied signature on it. The public key corresponds to a secret key, which could be just a random number (e.g., [CP92]), or a set of user attributes along with some randomness (e.g., [Bra00, Sections 4.5.2].) The issuer of a public-key certificate need not know the secret key corresponding to the certificate's public key.

3. **Secret-key certificates** : Similar to public-key certificates, secret-key certificates [Bra95c] also consist of a public key (corresponding to a secret key), and an issuer-supplied signature on it. The main difference between secret-key and public-key certificates is that, unlike public-key certificates, secret-key certificates can be generated by anyone, according to a distribution indistinguishable from that generated by the real issuer. That is, a verifier that sees a secret-key certificate $(pk, \sigma_{CA}(pk))$ cannot decide whether the latter (more precisely $\sigma_{CA}(pk)$) has been generated by the user or by a real issuer. The validity of the secret-key certificate can be verified only after the user proves that he knows the secret key corresponding to the certificate's public key.

Selective disclosure. In conventional credential systems (e.g., X.509 certificates), a user has to choose between (1) showing his credential and thus revealing all the attributes therein, and (2) not showing anything. In the more recent credential systems, a user has the ability to convince verifiers of the validity of his credential, while selectively choosing which information he wants to reveal or hide about his attributes. Revealing information about one's identity attributes may include the disclosure of the values of some or all of the attributes, or any properties about them.

User-signed transcripts. This feature allows a pair of interacting parties (a user and a verifier) to collect a user-signed transcript of their interaction. Being unforgeable, the resulting signed transcript can be used to enforce integrity and non-repudiation.

Signature with token. This feature allows a user to sign messages of his choice using the secret key corresponding to his credential's public key. The user hands in the resulting signature to a verifier, who may in turn show it to a third party. This feature can be useful in scenarios where more than two parties are involved, or just for evidence gathering as part of a signed audit trail.

Deployability in the Wallet-with-Observer setting. This indicates whether the credential system being considered has been deployed (or is deployable) in the "wallet-with-observer" setting introduced in the previous section. To be deployable in the wallet-with-observer setting, a credential system has to be sufficiently efficient to fit on state-of-the-art resource-limited portable devices.

In/Outflow control. In a wallet-with-observer setting, the observer may try to covertly communicate information to the outside world without the user's knowledge. This is called an outflow channel. Such a channel may transmit sensitive information about the user's transactions. The outflow control feature blocks this channel. Similarly, parties interacting with the wallet, may try to send covert — possibly harmful — instructions to the wallet's observer without the user's knowledge. This is called an inflow channel. The inflow control feature blocks this channel.

Token untraceability. This feature is concerned with the ability to hide the relationship between a credential and the identity of its holder. More precisely, the *untraceability* property states that showing a token to a verifier (possibly the issuer of that token) does not help him associate the token with the identity of its holder. Untraceability should hold unconditionally, even in cases where the issuer and verifier collude. There are weaker forms of untraceability, where the relationship between the token and the identity of its holder is hidden only in a statistical or computational sense.

Tokens unlinkability. This property states that the showings of different tokens held by the same user are not linkable to each other, unconditionally. There are weaker forms of unlinkability, namely statistical and computational unlinkability.

Multi-show unlinkability. This feature allows a certification authority to set up, for credentials it issues, a limit on the maximum number of times a credential can be shown before the showings become linkable to each other. Prior to the maximum limit, the showings are unconditionally unlinkable. Again, weaker variants of this property offer either statistical or computational unlinkability for showings performed before the maximum limit is reached.

Recertification. This feature allows a certification authority to issue new certificates for previously certified tokens. Recertification may be needed following the updating of some or all of the attributes encoded in an old token. The updating may or may not require the issuer to know the values of the attributes encoded in the old token.

Limited-show tokens. For a constant k, a k-show token is a token that remains unconditionally untraceable as long as it is shown a number of times less or equal to k. As soon as the token is shown a $(k+1)^{\text{st}}$ time, it becomes fully traceable. When k is infinite we fall back into the case of "purely" untraceable tokens. Note that, when a k-show token is used beyond the allowed k times, not only the showings become linkable to each other (which can be achieved by the multi-show unlinkability property alone), but they also become traceable to the identity of the token holder.

Blacklist proofs.[2] Often, expired and revoked credentials are added to a blacklist to prevent them from being used again. In some cases, the identity of the owner of the revoked credentials is unveiled and added to the blacklist. A blacklist proof allows innocent users to prove that they are not on the blacklist without revealing their identity. It is infeasible however for users whose credentials are on the blacklist to prove the opposite.

In the following sections, we briefly describe the credential systems that we compare in this survey.

2.4 Chaum's blind signatures

2.4.1 Main idea

The main goal of Chaum's blind signatures is to allow a user to obtain a digital signature on a message of his choice, without the signer learning the message that was signed or the signature received. The signer is considered as a certification authority (CA). The user can prove to a verifier that he has a valid credential by showing a correct message-signature pair. On the other hand, the verifier cannot trace a shown credential, back to the protocol instance that generated it, and thus to the identity of the user. This remains true even if the verifier colludes with the CA. This untraceability feature is a direct result of the *blinding* property of the issuing protocol. More details are given in the following sections.

[2] The terminology *Blacklist* proof may seem inadequate here since the goal is to prove that someone is *not* member of a blacklist. In that respect, the terminology Off-Blacklist proofs might be a better fit. For the sake of conformity with the literature however, we have decided to keep the usual "Blacklist proof" terminology.

2.4.2 Setting and Assumptions

Chaum's blind signatures operate in the RSA [RSA78] setting. The signer first chooses two large primes p and q, and computes the RSA modulus $n = pq$. He then chooses a public key e co-prime to $\phi(n)$, and computes private key $d = e^{-1} \mod \phi(n)$. The signer then publishes n and e, and keeps p, q, and d private. In addition, the signer chooses a collision-resistant one-way function f and makes it public. The security of Chaum's blind signatures is based on the hardness of the RSA problem, and the collision resistance of the function f.

In a more general setting, the signer may have several signing key pairs $(sk_1, pk_1), \cdots,$ (sk_ℓ, pk_ℓ) each corresponding to a well known attribute from $(A_1, \cdots, A_\ell)$. Holding a token signed under key (sk_i, pk_i), implies that the token holder satisfies attribute A_i.

2.4.3 Credential format

The following format has been originally proposed in [Cha87], and improved in [CFN88]. The recipient obtains a credential of the form:

$$(m, f(m)^d \mod n)$$

where m is a secret random number chosen by the recipient in $\mathbb{Z}_n^*$. The signer does not learn any information about the issued credential $(m, f(m)^d \mod n)$. Given the RSA public key (n, e), a verifier can check the validity of a credential (a, b) by verifying that a, b are in $\mathbb{Z}_n^*$, and that the following relation holds.

$$b^e = f(a) \mod n$$

2.4.4 Summary of security and privacy properties

Chaum's blind signatures provide a number of privacy and security features ranging from unforgeability and untraceability to security against oracle attacks. A more detailed description of these properties is given in the appendix in section B.1.1.

Chaum's blind signatures are considered to be the first incarnation of the concept of digital credentials, and like most emerging technologies they do have a number of limitations. For instance, it is not possible to encode multiple attributes in a token[3]. Moreover, tokens obtained through Chaum's blind signature scheme, do not contain information tied to the identity of the user, which makes them untraceable. In other words, even when such credentials are used in an abusive way, capturing the showing transcripts does not help in identifying their owner. Furthermore, because of the blind issuing process, Chaum's credentials are not revocable, which could pose a serious security problem. A summary of the features and limitations of Chaum's blind signatures is given in Table A.1.

2.5 Chaum-Pedersen signatures

2.5.1 Main idea

The credential system proposed by Chaum and Pedersen in [CP92], combines the Chaum-Pedersen signatures with Chaum's "signature blinding" technique. The Chaum-Pedersen credential system operates in the wallet-with-observer setting. First the wallet's observer and the user-controlled computer jointly compute a public key, whose corresponding private key is known only to the wallet-observer. The user-

[3] The only way to encode different attributes, using Chaum's blind signatures, is to issue different signatures with respect to different keys, each corresponding to a different attribute.

controlled computer and the wallet's observer, then conduct a blind signature issuing protocol with the issuer to obtain a blind certificate on the previously computed public key. This interaction with the issuer can be performed only with the cooperation of both the wallet's observer and the user-controlled computer. To show his credential, the user computes a blinded version of the initially obtained certificate, before showing it to a verifier. The credential showing can be performed only if the wallet's observer and user-controlled computer approve it.

2.5.2 Setting and Assumptions

The Chaum-Pedersen signatures operate in the discrete logarithm setting. Let p and q be large primes such that $q|(p-1)$. Let G_q be the unique subgroup of $\mathbb{Z}_p^*$ of order q. Let g be a generator of G_q. The parameters $p,\ q,$ and g are all public. The size of q is chosen such that computing discrete logarithms in G_q is hard. Let $H : \mathbb{Z}_p^4 \rightarrow \mathbb{Z}_q^*$ be a public one-way collision-resistant hash function.

There are two types of participants in the system: organizations and users. Each organization Z has a secret key $x_Z \in \mathbb{Z}_q$ of its choice, and a corresponding public key $h_Z := (g^{x_Z} \mod p)$. An organization can play the role of an issuer or a verifier. Each user is represented by a wallet, consisting of a tamper-proof module T (playing the role of an observer) and a user-controlled computing module C. The observer is assumed to have a secret key $x_T \in \mathbb{Z}_q$, and a corresponding public key $h_T := (g^{x_T} \mod p)$.

2.5.3 Credential format

First, the observer T and user-controlled computing module C jointly choose a random secret key $x \in \mathbb{Z}_q$, and compute the corresponding public key $h := (g^x \mod p)$.

The secret x is only known to T. Next, T and C engage in a blind signature protocol with the certification authority *CA* to obtain a signature $\sigma_{CA}(h)$ on their public key h. The issued credential is of the form:

$$(h, \sigma_{CA}(h) := (z, r, c))$$

where $z = h^{x_{CA}} \mod n$, and (r, c) is such that the equality $c = H(h, z, h_{CA}^{-c} g^r, z^{-c} h^r)$ holds. The signer does not learn anything about the issued credential (h, z, r, c). The obtained credential (h, z, r, c) is a public-key certificate. More details about the issuing and showing protocols of Chaum-Pedersen credentials are given in appendix B.2.

2.5.4 Summary of security and privacy properties

The Chaum-Pedersen signatures bear many similarities with Chaum's blind signatures. In addition to the properties offered by Chaum's blind signatures, they provide features such as impersonation resistance, whereby an attacker is unable to forge signatures on behalf of the wallet's observer. A more detailed description of these properties is given in appendix B.2.3.

Security and privacy limitations. It is worth mentioning that the security of the Chaum-Pedersen credential system relies in part on the hardness of the discrete logarithm problem, but most importantly on the tamper-resistance of the wallet. In particular, an attacker that succeeds in extracting the secret key of the wallet's observer, will be able to impersonate the wallet, withdraw new credentials on its behalf, and show any credential previously obtained by the wallet. Moreover, even when such compromised credentials are detected, it is not possible to trace them back to the wallet that owns them. This is due to the use of blind signatures. As a result, the Chaum-Pedersen credentials are completely irrevocable.

As noted above, the unlinkability of the card's transactions relies on the assumption that no party gets access to the content of the tamper-proof module (observer) even after the wallet is expired. This includes the issuer who may recollect the wallets at the end of their usage cycle. This is called the "hermiticity assumption". This assumption however does not always hold true in practice, and an issuer who gains access to the wallet's log, may easily trace down the wallet's transactions, by combining that log with information collected by verifiers. To fix this limitation, Cramer and Pedersen proposed a wallet with improved privacy [CP94], where access to the content of the tamper-proof module of the card does not help an issuer (possibly colluding with verifiers) link different credential showings to each other, or to the identity of the wallet holder. Although the protocol in [CP94] managed to circumvent the hermiticity assumption, it remains relatively costly in terms of round complexity. A summary of the features and limitations of the Chaum-Pedersen credentials is given in Table A.1.

2.6 Brands credentials

2.6.1 Main idea

Brands credentials [Bra00] are based on the notion of *restrictive* blind signatures. Informally, restrictive blind signatures are a special kind of signature satisfying the following two properties:

1. the user can obtain from the issuer a blinded signature on his public key, without the issuer learning any information about the user's public key or the issued signature,

2. the user can choose a subset of the attributes encoded in the public key to be signed, and keep their values hidden from the CA. The user may be required however to prove that the values of these attributes satisfy a certain agreed-upon predicate. This predicate is referred to in [Bra00] as the "blinding invariant". A complete definition of restrictive blind certificates can be found in [Bra00, Section 4.1].

To show his credential, a user reveals his public key accompanied by the issuer's signature on it. The user proves in addition that he knows the values of the attributes underlying the public key. He may also be required by the verifier to prove that the encoded attributes satisfy a certain agreed-upon predicate. Because the certificate was obtained through a blind issuing protocol the verifier cannot trace the revealed certificate back to the identity of the user, even if it colludes with the issuer.

Brands proposed two main types of credentials: One is based on the discrete logarithm representation problem (DL-REP), and the second on the RSA representation problem (RSA-REP). We do focus here on the discrete-logarithm-based version, and describe two constructions [Bra00, Sections 4.4.2 and 4.5.2] with different properties and computational efficiencies.

2.6.2 Setting and Assumptions

Brands credentials operate in the discrete logarithm setting. Let G_q be a group of prime order q and let g_0 be one of its generators. G_q is chosen such that computing discrete logarithm representations in it is hard. One possible construction is to choose G_q as the unique subgroup of $\mathbb{Z}_p^*$ of order q, where p and q are primes such that $p = 2q + 1$. Let $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ be a public one-way collision-resistant hash function.

In the setup phase, the certification authority randomly chooses $y_1, y_2, \cdots, y_\ell \in_R \mathbb{Z}_q$ and $x_0 \in_R \mathbb{Z}_q^*$, and computes $(g_1, g_2, \cdots, g_\ell, h_0) := (g_0^{y_1}, g_0^{y_2}, \cdots, g_0^{y_\ell}, g_0^{x_0}) \mod p$. The parameters $(g_1, g_2, \cdots, g_\ell, h_0)$ are then made public along with g_0, q, and a description of the group G_q and the hash function H.

The Discrete Logarithm Representation Problem : Let $(\varepsilon_1, \cdots, \varepsilon_\ell)$ be a set of elements from $\mathbb{Z}_q$, and let $\gamma := g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} \mod p$. We say that $(\varepsilon_1, \cdots, \varepsilon_\ell)$ is a discrete logarithm representation of γ with respect to the basis $(g_1, g_2, \cdots, g_\ell)$. In [Bra00, Section 2.3.2], Brands shows that given a random $\gamma \in G_q \setminus \{1\}$, and a basis $(g_1, g_2, \cdots, g_\ell)$, finding a discrete logarithm representation of γ with respect to $(g_1, g_2, \cdots, g_\ell)$ is at least as hard as breaking the discrete logarithm problem in G_q. Also, given γ and a representation $(\varepsilon_1, \cdots, \varepsilon_\ell)$ of γ with respect to basis $(g_1, g_2, \cdots, g_\ell)$, finding another representation $(\mu_1, \cdots, \mu_\ell)$ of γ with respect to the same basis is at least as hard as breaking the discrete logarithm problem in G_q.

2.6.3 Credential format I

To obtain a credential, a user first convinces the issuer that he fulfills a set of application-specific requirements necessary to receive that credential. The issuer then encodes ℓ user attributes in the credential. It is also possible for the user to encode a subset of the attributes which will remain hidden from the issuer. The user may be required, instead, to prove a certain property of this subset of attributes without revealing them. Let $x_1, \cdots, x_\ell$ denote all of the user's attributes to be encoded. The issued credential has the form:

$$(h, \sigma_{CA}(h) := (c'_0, r'_0, z')),$$

where

- $h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^\alpha$ denotes the credential's public key, with α a secret blinding factor randomly chosen by the user in $\mathbb{Z}_q^*$,
- $(c'_0, r'_0, z') \in \mathbb{Z}_q^2 \times G_q$ denotes the issuer's signature on h. The tuple (c'_0, r'_0, z') is such that:

$$c'_0 = H(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0} z'^{-c'_0})$$

At the end of the issuing protocol (See Appendix B.3.1), the issuer knows neither h, nor the signature (c'_0, r'_0, z'). The obtained token is of type public-key certificate. The next paragraph describes a secret-key certificate version of Brands' DL-based credentials.

2.6.4 Credential format II

The same setting introduced in section 2.6.2 applies to the credential format presented here. Let $x_1, \cdots, x_\ell$ denote all the user attributes that will be encoded in the credential. The attributes here are known to both the user and the *CA*. The *CA* issues a credential of the form:

$$(h', \sigma_{CA}(h') := (r'_0, c'_0)),$$

where

- $h' := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{\alpha_1}$ denotes the credential's public key, with α_1 a secret blinding factor randomly chosen by the user in $\mathbb{Z}_q^*$,

- $(r'_0, c'_0) \in \mathbb{Z}_q^2$ denotes the *CA*'s signature on h'. The pair (r'_0, c'_0) is such that:

$$c'_0 = H(h', g_0^{c'_0}(h')^{r'_0})$$

The obtained token is of type secret-key certificate. More details about the issuing, showing, and depositing protocols of Brands DL-based credentials, are given in appendix B.3.

2.6.5 Summary of security and privacy properties

Brands credentials (both format I and II) provide a multitude of features. Among these we note features such as credential's untraceability and unforgeability, selective disclosure, impersonation resistance, and security against false proofs. The latter is a soundness property that prevents users from proving claims (predicates) not satisfied by their attributes. A more detailed description of the security and privacy properties offered by Brands credentials is given in the appendix in section B.3.4.

It is worth noting here that Brands credentials are one-time show credentials; two showings of the same credential are linkable to each other. A summary of the features and limitations of Brands DL-based credentials is given in Table A.1.

2.7 SRSA-based Camenisch-Lysyanskaya credentials

2.7.1 Main idea

The credential systems proposed by Camenisch and Lysyanskaya [CL02b, CL04] rely on the idea that a user does not have to *reveal* his credential in order to prove that he has a valid one. Instead, the user just needs to *prove knowledge* of a valid credential issued by a recognized certification authority.

The credential system proposed in [CL02b] is based on the Strong RSA assumption, and offers a variety of state-of-the-art features. For example, it is possible to embed multiple attributes in the same credential, such that some of them may be chosen by the user and hidden from the *CA*. The CL-SRSA system also allows the user to selectively disclose information about his attributes. Also, since the initially obtained credential is never revealed to a verifier, the user can show his credential multiple times without the showings being linked to each other. The identity of the user can be unveiled however with the help of an escrow party. Moreover, [CL02b] presents a one-time show variant where credentials become traceable to the identity of their owners if shown more than once.

2.7.2 Setting and Assumptions

The CL-SRSA system operates in the RSA setting. Let $n := pq$ be an ℓ_n-sized RSA modulus chosen by the issuer, such that $p = 2p' + 1$, $q = 2q' + 1$, p', and q' are primes of similar size. The issuer also chooses uniformly at random $S \in_R QR_n$, and $R_1, \cdots, R_\ell$ and $Z \in_R \langle S \rangle$. To prove the well-formedness of these parameters, the issuer provides a proof of knowledge of the discrete logarithm of $R_1, \cdots, R_\ell$ and Z to the base S. Let ℓ_m, ℓ_c, ℓ_s, and $\ell_e = \ell_m + 3$ be system parameters. Let $\pm\{0,1\}^{\ell_m}$ be the domain from which the credential attributes are chosen.

2.7.3 Credential format

Let $x_1, \cdots, x_\ell$ denote the set of user attributes to be encoded in the credential. For $\ell' < \ell$, let $x_1, \cdots, x_{\ell'}$ be the subset of attributes chosen by the user, and hidden from the issuer. The user may be required to prove a property of these attributes to the *CA*. At the end of the issuing protocol, the user obtains a credential of the form:

$$((x_1, \cdots, x_\ell), \sigma_{CA}(x_1, \cdots, x_\ell) := (A, e, v)),$$

where

- A is an element of $\mathbb{Z}_n$
- e is a prime in the range $]2^{\ell_e+\ell_s+\ell_c+1}, 2^{\ell_e+\ell_s+\ell_c+2}[$
- v is an integer in the range $]0, 2^{\ell_e+\ell_c+\ell_s+1} + 2^{\ell_n+\ell_s}[$

and the tuple (A, e, v) satisfies the relation

$$Z = R_1^{x_1} \cdots R_\ell^{x_\ell} S^v (A)^e \mod n.$$

To show his credential, the user computes $\tilde{A} := AS^{r_A} \mod n$, a blinded version of A for a randomly chosen r_A, and proves to the verifier that he knows $x_1, \cdots, x_\ell$, and (e, v'), such that

$$Z = R_1^{x_1} \cdots R_\ell^{x_\ell} S^{v'} (\tilde{A})^e \mod n.$$

This makes it possible to show the credential unlinkably, an arbitrary number of times. The CL-SRSA system allows credentials to be used in the one-time show mode as well. The tokens obtained in the CL-SRSA system are simulatable, and are therefore considered of type secret-key certificate. In other words, there exists a polynomial-time simulator that can generate triplets (A', e', v') according to a distribution that looks indistinguishable from that of signature triplets (A, e, v) generated by the real *CA*. More details about the issuing, showing, and depositing protocols of the CL-SRSA system are given in appendix B.4. An exhaustive treatment can be found in [CL02b, BCL04].

2.7.4 Summary of security and privacy properties

The CL-SRSA system builds on the previous credential systems, and offers the multi-show unlinkability feature. This feature allows a user to show his credential indefinitely-many times without the showings being linked to each other. The CL-SRSA credentials are also untraceable (i.e., the showings cannot be linked to the instance of the issuing protocol that generate them, and thus to the identity of the user.) The main limitation of the CL-SRSA system is that the credential revocation procedure relies on a trusted third party (a key escrow party.)

A summary of the features and limitations of the CL-SRSA system is given in the appendix in Table A.1, and section B.4.3.

2.8 DL-based Camenisch-Lysyanskaya credentials

2.8.1 Main idea

The system in [CL04] follows the same approach as in [CL02b] and adopts the idea that a user does not have to reveal his credential to prove that he has a valid one. Instead, the user just needs to prove knowledge of a valid credential issued by a recognized certification authority.

The DL-based Camenisch-Lysyanskaya credential system [CL04], like its SRSA-based counterpart, allows the embedding of multiple attributes in the same credential, where some of the attributes can be chosen by the user and hidden from the issuer. In the showing phase, the user proves that he knows a valid credential from a recognized issuer, and that the attributes underlying this credential satisfy a certain agreed-upon predicate.

2.8.2 Setting and Assumptions

The CL-DL system is based on the discrete logarithm problem, and operates in a bilinear group setting. Let $G = \langle g \rangle$ and $\mathsf{G} = \langle \mathsf{g} \rangle$ be two groups of prime order q, and let $e : G \times G \to \mathsf{G}$ be an admissible bilinear map with the following properties:

- Bilinear: $\forall P, Q \in G, \forall a, b \in \mathbb{Z},\ e(P^a, Q^b) = e(P, Q)^{ab}$.
- Non-degenerate: $\exists P, Q \in G$ such that $e(P, Q) \neq 1_{\mathsf{G}}$.
- Efficient: There exists an efficient algorithm to compute e.

Note that whenever g is a generator of G, the group element $\mathsf{g} := e(g, g)$ is a generator of G. The *CA* first generates the parameters $(q, g, G, \mathsf{g}, \mathsf{G}, e)$ as specified above. Next, the *CA* chooses random $x, y, z_1, \cdots, z_\ell$ in $\mathbb{Z}_q$, and sets its secret key to $sk = (x, y, z_1, \cdots, z_\ell)$. The issuer then computes $X := g^x$, $Y := g^y$, and $Z_i := g^{z_i}$ for $1 \leq i \leq \ell$. Finally the *CA* publishes $(q, G, \mathsf{G}, g, \mathsf{g}, e, X, Y, Z_1, \cdots, Z_\ell)$ as its public key.

The CL-DL scheme is based on a discrete-logarithm-related assumption, introduced in [LRSW00] and referred to as the LRSW assumption.

The LRSW assumption. Suppose $G = \langle g \rangle$ is a group chosen according to the setting above. Let x, y be elements of $\mathbb{Z}_q$, and let $X, Y \in G$ be such that $X = g^x,\ Y = g^y$. Let $O_{X,Y}(.)$ be an oracle that on input $m \in \mathbb{Z}_q$, outputs a triplet $A = (a, a^y, a^{x+mxy})$ for a randomly chosen $a \in G$. Let k be a security parameter. The LRSW assumption states that given X, Y, and oracle access to $O_{X,Y}(.)$, it is hard to compute new triplets of the form (a, a^y, a^{x+mxy}) for a non-previously queried m. More formally, the LRSW assumption states that, for all probabilistic polynomial time adversary $\mathcal{A}dv$, the following probability is negligible in the security parameter k:

$$\mathit{Prob}_{(\mathcal{M},\mathcal{A},\mathcal{B},\mathcal{C})}\left[m \notin Q \wedge m \in \mathbb{Z}_q \wedge m \neq 0 \wedge a \in G \wedge b = a^y \wedge c = a^{x+mxy}\right]$$

where :

- Q is the set of queries that adversary $\mathcal{A}\mathbf{dv}$ has previously made to the oracle $O_{X,Y}(.)$, and
- $(\mathcal{M},\mathcal{A},\mathcal{B},\mathcal{C})$ is a tuple of random variables, whose value (m,a,b,c) is obtained through the following ordered sequence of events: $(q, G, \mathsf{G}, g, \mathsf{g}, e) \leftarrow Setup(1^k)$, $x \leftarrow \mathbb{Z}_q$, $y \leftarrow \mathbb{Z}_q$, $X = g^x$, $Y = g^y$, $(m,a,b,c) \leftarrow \mathcal{A}\mathbf{dv}^{O_{X,Y}}(q, G, \mathsf{G}, g, \mathsf{g}, e, X, Y)$.

2.8.3 Credential format

Let $\{m^{(i)}\}_{(1\leq i\leq \ell)}$ denote the set of user attributes to be encoded in the credential. Without loss of generality, we assume that for $\ell' \leq \ell$, the attributes $\{m^{(i)}\}_{(1\leq i\leq \ell')}$ are chosen by the user and hidden from the issuer. At the end of the issuing protocol, the user obtains a credential of the form:

$$(m^{(1)}, \cdots, m^{(\ell)}), \sigma_{CA}(m^{(1)}, \cdots, m^{(\ell)}) := (v, a, b, c, \{A_i, B_i\}_{1\leq i\leq \ell})$$

where v is a secret number randomly chosen by the user in $\mathbb{Z}_q$. The rest of the certificate, i.e., the tuple $(a, b, c, \{A_i, B_i\}_{1\leq i\leq \ell})$ is generated by the certification authority, and satisfies the equations:

$$\forall 1 \leq i \leq \ell,\ e(a, Z_i) = e(g, A_i) \text{ and } e(A_i, Y) = e(g, B_i)$$

$$e(a, Y) = e(g, b) \text{ and } e(g, c) = e(X, a)e(X, b)^v \prod_{i=1}^{\ell} e(X, B_i)^{m^{(i)}}$$

To show his credential, the user computes a blinded version of the initial certificate σ_{CA} as follows. First he randomly chooses $r, r' \in_R \mathbb{Z}_q$. Next he computes the tuple $\tilde{\sigma} := (a^{r'}, \{A_i^{r'}\}, b^{r'}, \{B_i^{r'}\}, c^{rr'})$. The latter is denoted $\tilde{\sigma} := (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c})$. The user then sends $\tilde{\sigma}$ to the verifier, and proves that he knows $(v, m^{(1)}, \cdots, m^{(\ell)})$ consistent with $(\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c})$. The user can also prove other predicates on his attributes.

Showing a blinded version of the credential instead of revealing the original one, makes it possible to show the credential unlinkably an unlimited number of times. The CL-DL system provides credentials for the one-time show mode as well. The credential certificates obtained in the CL-DL system are simulatable, and are therefore considered of type secret-key certificate. More details about the issuing, showing, and depositing protocols of the CL-DL system are given in appendix B.5. An exhaustive treatment can be found in [CL04, BCL04].

2.8.4 Summary of security and privacy properties

The CL-DL system provides the same security and privacy features as the CL-SRSA system. The main difference between the two systems is that CL-DL operates in a bilinear groups setting, and its security is based on the discrete logarithm assumption. A detailed description of the security and privacy properties offered by CL-DL is given in the appendix in section B.5.3.

A summary of the features and limitations of the CL-DL credentials is given in Table A.1.

2.9 Discussion

In this section we highlight the main features and weaknesses of the credential systems described in this chapter.

First we briefly state the paradigm used in each system. Chaum's blind signatures as well as the Chaum-Pedersen credential system are both based on the notion of blind issuing. The Brands I and Brands II systems are based on the notion of restrictive blind issuing. Finally, the CL-SRSA and CL-DL systems are based on the idea that a user need not reveal the original version of his credential and that instead he can just prove that he possesses one. As we shall see shortly, these conceptual distinctions do lead to major differences between the six studied credential systems.

While all six systems allow the encoding of some information in the issued credential, only Brands I, Brands II, CL-SRSA, and CL-DL allow the encoding of multiple attributes in a single credential. The same systems also make it possible for users to choose a subset of the encoded attributes, and keep them hidden from the CA. Similarly, all six systems allow for selective disclosure[4].

With respect to protecting the secrecy of credential attributes, we note that Chaum's blind signatures, Chaum-Pedersen signatures, Brands I, Brands II, and CL-DL systems, all hide the attributes underlying a given credential unconditionally. The same holds only statistically for the CL-SRSA system. Furthermore, efficient solutions to prove negative predicates have only been proposed for the Brands systems. It is not known if similar efficient solutions exist for the other systems.

[4] This is not offered in Chaum's basic blind signatures. It could be made possible however for special instances of the one-way hash function f.

With respect to untraceability, all six systems prevent issuers from tracing a showing transcript back to the identity of the user. The untraceability however is unconditional for Chaum's blind signatures, Chaum-Pedersen signatures, Brands I, and Brands II systems, while it is only statistical for the CL-SRSA and CL-DL[5] systems.

The CL-SRSA and CL-DL systems offer multi-show unlinkability, where multiple showings of the same credential are unlinkable to each other. Credentials in all the other systems are linkable if shown more than once.

With respect to limited-show capabilities, only Brands I, Brands II, CL-SRSA, and CL-DL offer this feature. Both Brands I and Brands II support one-time show credentials. The CL-SRSA, and CL-DL also support one-time show credentials, but unlike the Brands systems where the identity of an abusing user can be unveiled by any verifier, in the CL-SRSA, and CL-DL systems, an abuser's identity can only be revealed with the help of an escrow party. A more recent scheme by Camenisch, Hohenberger, Kohlweiss, Lysyanskaya, and Meyerovich [CHK+06] offers a k-time show capability, within a pre-defined time interval. That is, the credential can be shown a maximum of k times during a time period of pre-defined length, without the showings being linked or traced to the identity of the user.

Among all six systems, only Brands I and Brands II, offer showing transcript censoring capabilities. This allows a verifier to whom a credential was shown, to censor part of the predicate that was proved in the original showing, before relaying the

[5] In the CL-DL case, this is due to the statistical security of the verifiable encryption scheme [CS98] used in the showing protocol.

showing transcript to another party.

With respect to recertification, only Brands I, CL-SRSA, and CL-DL offer this feature without the user having to reveal the attributes underlying his credential. In the Brands I case however, this requires the revocation of the old credential. Similarly, recertification with selective attribute updating is only supported by the Brands I and CL-SRSA systems. In the CL-SRSA case, the updating is linkable to the issuing protocol instance that generated the original credential.

In terms of issuer's signature size, Brands II has the smallest, followed by Brands I, and CL-SRSA. CL-DL however has a signature size linear in the number of attributes embedded in the credential.

Finally, the Chaum-Pedersen credential system, as well as Brands I, and Brands II have already been implemented in the wallet-with-observer setting (e.g., on a mobile phone). We have not found any record indicating that the CL-SRSA and CL-DL systems have been deployed in such a setting.

Chapter 3

Accredited Symmetrically Private Information Retrieval (ASPIR)

In Chapter 2, we have described privacy-preserving credentials— a technology that allows users to selectively disclose information about data lying under their control. In this Chapter, we consider the problem of protecting access to personal information stored on a remote database lying outside the user's control. Our goal is to control access to this information according to privacy policies defined by the owners of the data. More specifically, we propose a new primitive allowing users to authorize access to their remotely-stored data according to a self-chosen access policy, without the storage server learning information about the access pattern, or even the index of the records being retrieved. The proposed solution, which we call Accredited Symmetrically Private Information Retrieval *(ASPIR), utilises symmetrically private information retrieval and privacy-preserving digital credentials. We present three constructions based on the discrete logarithm and RSA problems.*

3.1 Introduction

Access control has always been an interesting and challenging discipline of information security, and its use has reached many areas of our daily life (e.g., banking, business, healthcare). In order to access a service or a resource, users are often required to show their identity or prove possession of a set of qualifications and privileges. In many cases, this forces individuals into leaving identity trails behind, which can be used for criminal activities such as unlawful monitoring and identity theft. The data collected from such interactions — which is often rich in personal information — is in most cases stored in databases lying outside the control of the individual who owns it. The latter is also referred to as the data subject. Various techniques have been proposed in the literature to strengthen users' privacy and help protect their personal information. Among these, we note privacy-preserving digital credentials [Cha85, Bra00, CL02b], and symmetrically private information retrieval protocols [GIKM98, CMO00, KO97, AIR01, Lip05].

In a symmetrically private information retrieval (SPIR) system, there are generally two players: a Sender and a Receiver. The Sender has a database DB of records, and the Receiver submits a query Q to the Sender in order to retrieve a particular record. The main requirement in a SPIR system is privacy for both the Sender and the Receiver. That is, on the one hand, the Sender should not learn any information about the index of the record the Receiver is interested in, and on the other hand, the Receiver should not learn any information about the database, beyond the content of the record defined in the query Q, and what is already publicly known. In particular, the Receiver should not be able to learn information about more than one record per query. For instance, the Receiver should not be able to learn, through one query, the value of any function on a set of more than one record. SPIR systems have many real-world applications; for instance, consider a scenario where the inventor of a new drug needs information on a number of chemical components that will constitute his final product. This information can be accessed for a fee at some central database. This database could be managed, however, by parties with possibly competitive interests, and the inventor might be concerned that his intellectual property (IP) could be compromised. It is therefore natural, that the inventor might want the content of his queries to remain concealed from the database manager. The latter, on the other hand, wants to be paid for all information retrieved from his database. It is clear that the SPIR system described above, can be a solution to this pair of conflicting requirements.

There are similar applications however, that are closely related to the example above, which cannot be solved by a SPIR primitive. Consider for example the following e-health scenario where three types of participants are involved: (1) a patient, (2) a medical database containing the health records of patients, and (3) a doctor querying the medical database on patients' health records. The medical database and the doctor can be thought of as the Sender and Receiver, respectively, in a traditional SPIR setting. The requirements in the e-health application are as follows:

1. **Privacy for the Receiver**: The Receiver (doctor) wants to retrieve records from the medical database, without the Sender (DB) learning the index of those records, and thus the identity of his patient.

2. **Privacy for the Sender**: The Sender (DB) wants to be sure that, for each query, the Receiver (doctor) learns information only on one record (defined in the query) and nothing about the other records.

3. **Privacy for the data subject**: In order to comply with privacy legislation, the Sender wants to be sure that the Receiver has a valid reading authorization from the owner of the targeted record (i.e., the patient). We call the latter, *an Authorizer*. Notice that the Sender should not be able to learn the Authorizer's identity, otherwise the first requirement will be violated.

Another example where existing SPIR systems are insufficient, is that of credit history check-ups. Often when applying for a loan, or even a credit card, customers are asked by their would-be lenders for a permission to check their credit history. This credit history is generally stored in some large database managed by a government agency, or a private organization (possibly a business competitor), and is available for viewing for a fee. We call the manager of this database a Sender. We also denote the lender and customer, by Receiver and Authorizer, respectively. The requirements of the application can be stated as follows:

1. The Receiver wants to retrieve the credit history of the customer without disclosing the latter's identity to the Sender,
2. The Sender, who charges viewing fees per record retrieved, wants to be sure that the Receiver obtains information only on one record at a time, and nothing else about the other records.
3. In order to comply with privacy legislation, the Sender wants to be sure that the Receiver has obtained explicit viewing consent from the owner of the target record. This should be done without the Sender learning the identity of the owner of the target record.

The two examples above show typical scenarios where plain SPIR primitives fall short of protecting the interests of the Sender, the Receiver, and the Authorizer at the same time. The solution we provide here, addresses the interests of all three parties, and solves the problems described above. We call the presented solution: *Accredited SPIR*. In the remainder of this chapter, we sometimes refer to the latter set of requirements, namely privacy for the data-subject, the Sender, and the Receiver, as the *Accredited SPIR problem*.

Solution highlight

Let Q denote the query the Receiver submits to the Sender, and let R be the Sender's response. The Receiver recovers the answer to his query from R. Our main goal here is to make sure that before processing the query Q, the Sender is convinced that the Receiver has obtained an explicit consent from the owner of the record defined in Q, *without* revealing the identity of this owner (i.e., the Authorizer). The solution we propose, utilises three cryptographic primitives: privacy-preserving digital credentials, homomorphic encryption, and SPIR systems. We discussed privacy-preserving credentials in chapter 2. For the purpose of our solution, we assume that we have a credential system such as those in [Bra00, CL02b, CL04], where users are able to show their credentials unlinkably, and to selectively disclose information about their attributes.

More specifically, we assume that the Authorizer has a *CA*-issued identity credential (*Cred*) containing a set of attributes (ID,Age,$\cdots$). The idea is to first make the Authorizer and Receiver jointly compute the query Q, and then have the Authorizer produce a signed proof of knowledge of the secret attributes embedded in *Cred*. Also included in the proof, is a predicate stating that the value of the ID attribute embedded in *Cred* is the same as the one encoded in the query Q. The Receiver then deposits the signed proof along with the query to the Sender. The Sender first checks the validity of the proof. If accepted it carries on with the SPIR protocol and processes the query, otherwise it rejects.

As mentioned earlier, the signed proof does not reveal any information about the credential holder, and yet guarantees that the content of the query is consistent with the secret identity attribute embedded in the credential. Furthermore, owing to the fact that it is hard for a computationally bounded adversary to forge credentials, or make proofs about credentials he does not own, the Sender can be sure that the Receiver has indeed obtained an explicit consent from the targeted record's owner.

We present three constructions to solve the accredited SPIR problem. The first is based on a modified version of one of Brands DL-based credentials [Bra00, Section 4.5.2], the ElGamal cryptosystem, and a SPIR system proposed by Lipmaa in [Lip05]. The second and third constructions are variants of the first, and they use an RSA-based version of Brands credentials [Bra00, Section 4.2.2], in combination with the ElGamal, and the Okamoto-Uchiyama [OU98] cryptosystems, respectively.

Chapter organization

The remainder of this chapter is organized as follows. In Section 3.2, we review previous results and related work available in the literature. In Section 3.3, we introduce the main building blocks used in the first construction as well as throughout the chapter. Section 3.4 presents a DL-based accredited SPIR construction. Sections 3.6 and 3.7 highlight the security and privacy features, as well as the performance of the first construction. Section 3.8.2 contains a second construction based on a RSA version of Brands credentials, and Section 3.8.4 presents a third variant based on the Okamoto-Uchiyama cryptosystem. Finally, Section 3.9 highlights our conclusions.

3.2 Related work

Much research has gone into the problem of managing personal data in accordance with a user-defined privacy policy. In [GMM06a], for instance, Golle, McSherry, and Mironov, propose a mechanism by which data collectors can be caught and penalized if they violate an agreed-upon policy, and disclose sensitive data about a data-subject. The main idea in [GMM06a] is that a data-collector would place a *bounty*, which it must forfeit if a privacy violation is uncovered. The bounty could be explicit in the form of a bond, or implicit in the form of penalties imposed if privacy is violated. This technique however is geared towards violation detection after the fact, and assumes the existence of *active* bounty hunters who seek to induce dishonest data collectors into committing unlawful disclosures.

Another related approach is that of policy-based encryption by Bagga and Molva [BM05, BM06]. Policy-based encryption allows a user to encrypt a message with respect to an access policy formalized as a monotone Boolean expression. The encryption is such that only a user having access to a qualified set of credentials, complying with the policy, is able to successfully decrypt the message. The context in [BM05, BM06], however, is different from the one we consider in this chapter, since the authors' goal there is to allow the user to send a secret message to a designated set of players defined by a policy. Whereas, in our context, the target data is already stored in a database, and the goal is to allow parties authorized by the data owners to retrieve this data, without the database manager learning which data has been retrieved or the identity of the data owners.

In [SWP00], Song, Wagner, and Perrig, present a scheme allowing keyword search on encrypted data. Their setting consists of a user, and a server storing encrypted data owned by the user. The server can process search queries on the user's stored ciphertext, only if given proper authorization from the user. The scheme in [SWP00] also supports hidden user queries, where the server conducts the search without learning anything about the content of the query. Although the context of [SWP00] is related to the settings we address, it is not clear how the fore-mentioned work can be applied to the problem we describe in this chapter, since delegating querying capabilities to a third party (e.g., a Receiver) may require the user to reveal his encryption key, and thus share all of his past and future secrets. Besides, it is not clear how the scheme in [SWP00] can hide the identity of the data-owner from the server, or how it can impose restrictions (e.g., wrt. time or usage) on the search capabilities delegated to a third party.

Finally, Aiello, Ishai, and Reingold [AIR01] consider a scenario where users privately retrieve data from a database containing a set of priced data items. The proposed protocol is called priced oblivious transfer, and it allows a user U, who made an initial deposit, to buy different data items — subject to the condition that U's balance contains sufficient funds — without the database manager learning which items U is buying. We believe the construction in [AIR01] is the first to consider imposing additional requirements on oblivious transfer protocols. While interesting in their own right, the added requirements do not address the issue of protecting the identity of the data owners.

3.3 Building Blocks for the DL-based construction

3.3.1 Brands credentials

For the purpose of the ASPIR protocol we consider the discrete-logarithm-based Brands credential system with format I (Br-DL-I). We have already introduced this system in

Section 2.6.3. In the following, we briefly recall the setting and basic operations of the Br-DL-I system.

System setting. On input the security parameter κ, the *CA* randomly chooses κ-sized primes p and q such that $2q|(p-1)$. Let G_q be the unique subgroup of $\mathbb{Z}_p^*$ of order q, and let g_0 be one of its generators. The *CA* also chooses $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, a public collision-resistant hash function. In the setup phase, the certification authority randomly chooses $y_1, y_2, \cdots, y_\ell$ and $x_0 \in_R \mathbb{Z}_q^*$, and computes the basis $(g_1, g_2, \cdots, g_\ell, h_0) := (g_0^{y_1}, g_0^{y_2}, \cdots, g_0^{y_\ell}, g_0^{x_0}) \mod p$. The parameters $(g_1, g_2, \cdots, g_\ell, h_0)$ are then made public along with g_0, q, G_q, and H.

Credential issuing. To obtain a credential, a user first convinces the certification authority that he fulfills a set of application-specific requirements necessary to receive that credential. The certification authority then encodes a set of ℓ user attributes in the credential. It is also possible for the user to keep a subset of those attributes hidden from the certification authority. Let $x_1, \cdots, x_\ell$ denote all the attributes to be encoded. The credential's public key is then computed as $h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^\alpha$, where α is a secret blinding factor randomly chosen in $\mathbb{Z}_q^*$ by the user. The certification authority's signature on the credential is a triplet $(c'_0, r'_0, z') \in \mathbb{Z}_q^2 \times G_q$, satisfying the relation $c'_0 = H(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0} z'^{-c'_0})$. At the end of the issuing protocol, the certification authority knows neither h nor the signature (c'_0, r'_0, z'). Figure 3.1 summarizes the issuing protocol.

Credential showing. In order to have access to a service, user U can show his credential without the verifying party being able to do the following : (1) learn informa-

User U	Public Info	CA
	$(g_i)_{0\leq i\leq\ell}, h_0, (g_i^{x_0})_{0\leq i\leq\ell}, h_0^{x_0}, q, G_q, H$	
	$\xleftrightarrow{\text{agree on attributes } (x_1,\cdots,x_\ell)}$	$w_0 \in_R \mathbb{Z}_q$
		$a_0 := g_0^{w_0}$
$\alpha_1 \in_R \mathbb{Z}_q^*,$	$\xleftarrow{a_0,b_0}$	$b_0 := (g_1^{x_1}\cdots g_\ell^{x_\ell}h_0)^{w_0}$
$\alpha_2, \alpha_3 \in_R \mathbb{Z}_q,$		
$h := (g_1^{x_1}\cdots g_\ell^{x_\ell}h_0)^{\alpha_1}$		
$z := (g_1^{x_1}\cdots g_\ell^{x_\ell}h_0)^{x_0}$		
$z' := z^{\alpha_1}$		
$a_0' := h_0^{\alpha_2}g_0^{\alpha_3}a_0$		
$b_0' := (z')^{\alpha_2}h^{\alpha_3}b_0^{\alpha_1}$		
$c_0' := H(h, z', a_0', b_0')$		
$c_0 := c_0' + \alpha_2 \mod q$	$\xrightarrow{c_0}$	
	$\xleftarrow{r_0}$	$r_0 := c_0x_0 + w_0 \mod q$
$r_0' := r_0 + \alpha_3 \mod q$		
accept iff $a_0'b_0' = (g_0h)^{r_0'}(h_0z')^{-c_0'}$		
store $h, \sigma_{\text{CA}}(h) := (z', r_0', c_0')$		

Figure 3.1: Br-DL-I Credential Issuing protocol

tion about the encoded attributes beyond what U willingly discloses, or (2) link the credential to the user's identity even if it colludes with the certification authority.

In practice, to show his credential to a verifying party, user U reveals the following : (1) the credential's public key h along with a signature $\sigma_{\text{CA}(h)} := (z', c_0', r_0')$, and (2) a signed proof of knowledge of a representation of h, with respect to basis $(g_1, g_2, \cdots, g_\ell, h_0)$. This signature is performed on a challenge m chosen by the verifier. The verifier checks the validity of the credential by making sure the relation

$$c_0' \stackrel{?}{=} H(h, z', g_0^{r_0'}h_0^{-c_0'}, h^{r_0'}z'^{-c_0'})$$

holds. If the credential is valid, the verifier moves on to check the validity of the signed proof of knowledge. Figure 3.2 sketches the basic showing protocol of the Br-DL-I system.

User U	Public Info	Verifier
	$(g_i)_{0\leq i\leq \ell}, h_0, (g_i^{x_0})_{0\leq i\leq\ell}, h_0^{x_0}, q, G_q, H$	
$w_1, \cdots, w_{\ell+1} \in_R \mathbb{Z}_q$	$\xleftarrow{\quad m \quad}$	$m :=$ *nonce*$\|\|$*optional description*
$a := g_1^{w_1} \cdots g_\ell^{w_\ell} h_0^{w_{\ell+1}}$		*of the purpose of the interaction*
$c := H(h, a, m)$		
$r_1 := \alpha c x_1 + w_1$		
$\vdots$		
$r_\ell := \alpha c x_\ell + w_\ell$		
$r_{\ell+1} := \alpha c + w_{\ell+1}$	$\xrightarrow{h,\sigma_{\mathrm{CA}}(h):=(z',c'_0,r'_0),(a,r_1,\cdots,r_{\ell+1})}$	
		accept iff
		$c'_0 = H(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0} z'^{-c'_0})$,
		and for $c := H(h, a, m)$,
		$h^c a = g_1^{r_1} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$ holds.

Figure 3.2: Br-DL-I basic showing protocol – signed proof of knowledge

The signed proof of Figure 3.2 can be computed, for example, with respect to a predicate $\mathcal{P}$ of the form "$x_i = x_j$" for some $(i, j) \in [1, \ell]^2$. In this case, we have $h = (g_1^{x_1} \cdots (g_i g_j)^{x_i} \cdots g_\ell^{x_\ell} h_0)^{\alpha_1}$, and the proof of knowledge is carried out as follows. The user first computes a challenge $c := H(h, a, \mathcal{P}, m)$, using a description of the predicate $\mathcal{P}$ as a parameter to the hash function. The user then proves knowledge of a representation of h with respect to basis $(g_1, \cdots, g_{i-1}, g_i g_j, g_{i+1}, \cdots, g_{j-1}, g_{j+1}, \cdots, g_\ell, h_0)$, using the challenge c. The above method is commonly referred to as the Fiat-Shamir heuristic [FS86] in the literature. User U can also prove a much wider class of predicates in a similar fashion [Bra00, Section 3.6].

3.3.2 ElGamal homomorphic encryption

Our DL-based accredited SPIR construction relies on the ElGamal encryption scheme because of its homomorphic properties and because it fits well the setting of the Br-DL-I credentials. In the following we recall the settings of the ElGamal cryptosystem.

Settings. Let p, q, and G_q, be the public parameters chosen by the *CA* in the setup of the Br-DL-I credential system. User U randomly chooses g_{ElG}, a generator of G_q, and $x_u \in_R \mathbb{Z}_q^*$, and computes $y_{\mathsf{ElG}} := g_{\mathsf{ElG}}^{x_u} \mod p$. User U then publishes his ElGamal public key $(G_q, g_{\mathsf{ElG}}, y_{\mathsf{ElG}})$, and keeps his private key x_u secret. A message $m \in G_q$, can be encrypted by choosing a random $r \in_R \mathbb{Z}_q^*$, and computing $c = (g_{\mathsf{ElG}}^r, y_{\mathsf{ElG}}^r m) = (c_1, c_2)$. Using U's private key, the plaintext can be recovered as $m = c_2 / c_1^{x_u}$. Given a constant α, and encryptions of m and m', it is easy to compute randomized encryptions of $m \times m'$ and m^α.

3.3.3 AIR-based Lipmaa $\binom{n}{1}$-OT

In [Lip05], Lipmaa proposes a SPIR scheme based on ideas from a construction by Aiello, Ishai, and Reingold [AIR01]. Lipmaa's SPIR scheme is computationally private for the Receiver and perfectly private for the Sender. Its security relies on the hardness of the *Decisional Diffie-Hellman* and the *Decisional Composite Residuosity* problems [Lip05]. The SPIR scheme in [Lip05] has a log-squared communication complexity (in the size of the database).

Settings. Let DB denote the Sender's private database. Let $\Pi_{\mathsf{hom}} := (\mathsf{Gen}_{\mathsf{hom}}, \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}, \mathsf{D}_{\mathsf{sk}_{\mathsf{hom}}})$ and $\pi := (HomGen, HomEnc_{pk}, HomDec_{sk})$ be two homomorphic public-key cryptosystems such that (1) the ciphertext space of $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$ is contained in the message space of $HomEnc_{pk}$, and (2) the entries $\mathsf{DB}[j]$ belong to $\mathcal{M}_{\mathsf{hom}}$, the message space of $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$. In other words, we have for all $0 \leq j \leq n-1$, $\mathsf{DB}[j] \in \mathcal{M}_{\mathsf{hom}}$. We also require that $|\mathsf{DB}| := n \leq |\mathcal{M}_{\mathsf{hom}}|$.

Main idea. Let DB denote the Sender's private database as above, and let s be the index of the record the Receiver is interested in. The receiver computes $\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(s)$, a homomorphic encryption of s, and sends it to the Sender. Using the homomorphic properties of the encryption scheme $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$, the Sender computes for each record $\mathsf{DB}[j]$ in the database, $\mathsf{DB}'[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s-j)+\mathsf{DB}[j])$, where δ_j is a blinding factor, chosen by the Sender, uniformly at random in $\mathcal{M}_{\mathsf{hom}}$ – the message space of $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$.[1] All the encrypted records $\mathsf{DB}'[j]$ are then sent back to the Receiver, who will be able to retrieve something meaningful only from $\mathsf{DB}'[s] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\mathsf{DB}[s])$. For $j \neq s$, $\mathsf{DB}'[j]$ will decrypt to randomness. The construction in [Lip05] follows a similar methodology to the above, except that the Sender returns one single ciphertext to the Receiver. To achieve this, the Sender uses an extra loop of superposed encryptions that leads to a randomized ciphertext of $\mathsf{DB}'[s]$. Only the latter is sent back to the Receiver. This is done as follows. The database $(\mathsf{DB}[1], \cdots, \mathsf{DB}[n])$ is arranged in an α-dimensional $\lambda_1 \times \cdots \times \lambda_\alpha$ hyper-rectangle for some pre-defined positive integers λ_j, such that $n = \prod_{j=1}^{\alpha} \lambda_j$.[2] Each record $\mathsf{DB}[i]$ is indexed by a tuple $(i_1, \cdots, i_\alpha)$ on this hyper-rectangle, where $i_j \in \mathbb{Z}_{\lambda_j}$. To retrieve a particular record $(s_1, \cdots, s_\alpha)$, the Receiver submits to the Sender a homomorphic encryption $\beta_{jt} := HomEnc_{pk}(b_{jt})$, for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, where $b_{jt} = 1$ if $t = s_j$, and $b_{jt} = 0$ otherwise. The Sender exploits the homomorphic properties of the encryption scheme $HomEnc_{pk}(.)$ to create a new $(\alpha-1)$-dimensional

[1] If $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$ is additively homomorphic, then $\mathsf{DB}'[j]$ can be computed as $(\mathsf{c} \times \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(-j))^{\delta_j} \times \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\mathsf{DB}[j])$.

[2] The values of α and the λ_j's are chosen in a way that minimizes the communication complexity of the protocol, while keeping the Sender's and Receiver's computations practical. In [Lip05], Lipmaa sets $\lambda_j = n^{1/\alpha}$ for all j, and shows that communication complexity is optimal for $\alpha = \log n$. Few other choices of α and the λ_j's, that optimize the protocol efficiency for cases where the database contains *long* (resp. *short*) documents, are also given [Lip05].

database DB$_1$, such that $\forall(i_2, \cdots, i_\alpha) \in \mathbb{Z}_{\lambda_2} \times \cdots \times \mathbb{Z}_{\lambda_\alpha}$, DB$_1(i_2, \cdots, i_\alpha)$ is equal to an encryption of DB$_0(s_1, i_2, \cdots, i_\alpha)$, where DB$_0$ is the Sender's original database DB. The same procedure is repeated, and at the j^{th} iteration, an $(\alpha - j)$-dimensional database DB$_j$ is obtained by the Sender, such that DB$_j(i_{j+1}, \cdots, i_\alpha)$ is equal to a j-times encryption of DB$_0(s_1, \cdots, s_j, i_{j+1}, \cdots, i_\alpha)$, where $(s_1, \cdots, s_j)$ are fixed as above, and $(i_{j+1}, \cdots, i_\alpha) \in \mathbb{Z}_{\lambda_{j+1}} \times \cdots \times \mathbb{Z}_{\lambda_\alpha}$. After α iterations, the Sender obtains DB$_\alpha$, an α-times encryption of the target record DB$_0(s_1, \cdots, s_\alpha)$. The Sender returns DB$_\alpha$ to the Receiver, who needs to decrypt it α times to recover DB$(s) \overset{\text{def}}{=}$ DB$_0(s_1, \cdots, s_\alpha)$. Notice that in the hyper-rectangle construction above, the Receiver can cheat by maliciously sending $\beta_{jt} := HomEnc_{pk}(1)$, for $t \neq s_j$. To stop such attacks, the Sender performs the repeated encryptions above, on DB$_0$ = DB$'$ rather than on the Sender's original DB. Recall that entries in DB$'$ are of the form DB$'[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s - j) + \text{DB}[j])$, where $\mathsf{pk}_{\mathsf{hom}}$ is the public key of the Receiver. Let s' be the index corresponding to the β_{jt}'s. At the end of the protocol, the Receiver obtains DB$_\alpha$, which he decrypts α times to recover DB$'[s'] = \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_{s'}(s - s') + \text{DB}[s'])$. Next, the Receiver decrypts DB$'[s']$ once again, and recovers something meaningful (DB$[s]$) only if $s' = s$. A summary of the whole protocol is given in Figure 3.3.

Length Flexible Additively Homomorphic Encryption

In [Lip05], Lipmaa requires the use of a *Length Flexible Additively Homomorphic* (LFAH) public-key cryptosystem (e.g., [DJ01, DJ03]) to implement $HomEnc_{pk}$. A LFAH public-key cryptosystem is a tuple $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where

- Gen is a key generation algorithm, that on input a security parameter k returns a secret key / public key pair $(\mathsf{sk}, \mathsf{pk})$.

```
Receiver (index $s$)                Public Info                Sender (Database DB)
                     $|\mathrm{DB}| := n \le |\mathcal{M}_{\mathsf{hom}}|, k, \mathcal{R}$

$(\mathsf{sk}_{\mathsf{hom}}, \mathsf{pk}_{\mathsf{hom}}) \leftarrow \mathsf{Gen}_{\mathsf{hom}}(1^k)$
$(sk, pk) \leftarrow HomGen(1^k)$
    } Publish $\mathsf{pk}_{\mathsf{hom}}, pk$

$\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(s)$
For $j := 1$ to $\alpha$ do :
    For $t := 0$ to $\lambda_j - 1$ do :
        $r_{jt} \in_R \mathcal{R}$
        $\beta_{jt} := HomEnc_{pk}(b_{jt}, r_{jt})$,
            where $b_{jt} := 1$ if $t = s_j$,
            and $b_{jt} := 0$ otherwise.     $\xrightarrow{\mathsf{c},\ \{\beta_{jt}\}_{1 \le j \le \alpha, 0 \le t < \lambda_j}}$

                                             Check validity of $\mathsf{pk}_{\mathsf{hom}}$ and $\mathsf{c}$.
                                             For $j := 1$ to $n$ do :
                                                 $\delta_j \in_R \mathcal{M}_{\mathsf{hom}}$
                                                 $\mathrm{DB}_0[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s - j) + \mathrm{DB}[j])$.
                                                     $:= (\mathsf{c} \times \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(-j))^{\delta_j} \times$
                                                         $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\mathrm{DB}[j])$.[1]

                                             For $j := 1$ to $\alpha - 1$ do :
                                                 For $i_{j+1} := 0$ to $\lambda_{j+1} - 1, \cdots,$
                                                         $i_\alpha := 0$ to $\lambda_\alpha - 1$ do :
                                                 $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha) :=$
                                                     $\prod_{t \in \mathbb{Z}_{\lambda_j}} (\beta_{jt})^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)}$
$\mathrm{DB}'_\alpha := \mathrm{DB}_\alpha$     $\xleftarrow{\mathrm{DB}_\alpha}$     $\mathrm{DB}_\alpha := \prod_{t \in \mathbb{Z}_{\lambda_\alpha}} (\beta_{\alpha t})^{\mathrm{DB}_{(\alpha-1)}(t)}$
For $j := \alpha$ downto $1$ do :
    $\mathrm{DB}'_{j-1} := HomDec_{sk}(\mathrm{DB}'_j)$
Output $\mathrm{DB}[s] := \mathsf{D}_{\mathsf{sk}_{\mathsf{hom}}}(\mathrm{DB}'_0)$
```

Figure 3.3: Lipmaa $\binom{n}{1}$-OT

- Enc is an encryption algorithm, that on input (pk, s, m, r) where pk is a public key, $s \in \mathbb{Z}^+$ is a length parameter, m is a plain text, and r is a random string, returns $\mathsf{Enc}^s_{\mathsf{pk}}(m; r)$

- Dec is a decryption algorithm that on input (sk, s, c), where sk is a secret key, s is a length parameter, and c is a ciphertext, returns a plaintext $\mathsf{Dec}^s_{\mathsf{sk}}(c)$.

For a length parameter $s \in \mathbb{Z}^+$, let $\mathcal{M}_s$ and $\mathcal{C}_s$ be the message space and ciphertext space of Enc. More precisely, for any $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^k)$ and for any $s \in \mathbb{Z}^+$, we should have $\mathsf{Enc}^s_{\mathsf{pk}} : \mathcal{M}_s \times \mathcal{R} \to \mathcal{C}_s$ and $\mathsf{Dec}^s_{\mathsf{sk}} : \mathcal{C}_s \to \mathcal{M}_s$, where $\mathcal{R}$ is a randomness space independent of s.

Lipmaa requires in [Lip05] that for some positive integer a, $\mathcal{C}_s \subseteq \mathcal{M}_{s+a}$ for every s, and denotes by ξ the smallest among those a's.

A LFAH cryptosystem is additively homomorphic if for any key pair $(\mathsf{sk}, \mathsf{pk})$, any length parameter s, any $m, m' \in \mathcal{M}_s := \mathbb{Z}_{\sharp \mathcal{M}_s}$, and any $r, r' \in \mathcal{R}$, $\mathsf{Enc}^s_{\mathsf{pk}}(m; r) \cdot \mathsf{Enc}^s_{\mathsf{pk}}(m'; r') = \mathsf{Enc}^s_{\mathsf{pk}}(m + m'; r \circ r')$ where $\cdot$ is a multiplicative group operation in $\mathcal{C}_s$, $+$ is addition in $\mathcal{M}_s$, and $\circ$ is a groupoid operation in $\mathcal{R}$. It is assumed in [Lip05] that $\log \sharp \mathcal{M}_1 = k$, $\sharp \mathcal{M}_s = (\sharp \mathcal{M}_1)^s$, and that $\sharp \mathcal{C}_s = \sharp \mathcal{M}_{s+\xi}$.

The use of a LFAH cryptosystem is particularly important in the context of the Lipmaa OT [Lip05], because for $2 \leq j \leq \alpha - 1$, the terms $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha)$ and DB_α (cf. Fig 3.3) are computed as encryptions of other *ciphertexts*. For example, for $2 \leq j \leq \alpha - 1$, the plaintext being encrypted in $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha)$ is in fact a $(j-1)$-times encryption of $\mathrm{DB}_0(s_1, \cdots, s_j, i_{j+1}, \cdots, i_\alpha)$.[3] Therefore, for these multiple encryptions to be correct, we need to make sure that, for all $1 \leq j \leq \alpha - 1$, the ciphertext spaces of the j^{th} encryption level is a subset of the message space of the $j + 1^{st}$ encryption level. This is achieved by implementing the homomorphic cryptosystem

[3] That is, $\mathsf{D}_{\mathsf{sk}_{\mathsf{hom}}}(\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha))$ is itself a $(j-1)$-times encryption of $\mathrm{DB}_0(s_1, \cdots, s_j, i_{j+1}, \cdots, i_\alpha)$

$(HomGen, HomEnc_{pk}, HomDec_{sk})$ of Fig. 3.3 as a LFAH cryptosystem $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. More precisely, for $1 \leq j \leq \alpha$ and $0 \leq t \leq \lambda_j - 1$, β_{jt} is computed as $\beta_{jt} := \mathsf{Enc}_{\mathsf{pk}}^{s+(j-1)\xi}(b_{jt}, r_{jt})$. Here $\beta_{jt} \in \mathcal{C}_{s+(j-1)\xi} = \mathcal{M}_{s+j\xi}$.

Additionally, it is assumed that the ciphertext space M of the cryptosystem $(\mathsf{Gen}_{\mathsf{hom}}, \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}, \mathsf{D}_{\mathsf{sk}_{\mathsf{hom}}})$ is contained in the plaintext space $\mathcal{M}_1$ of $\mathsf{Enc}_{\mathsf{pk}}^{s}$.

Following the observation that $\mathsf{Enc}_{\mathsf{pk}}^{s+\xi}(m_2; r_2)^{\mathsf{Enc}_{\mathsf{pk}}^{s}(m_1;r_1)} = \mathsf{Enc}_{\mathsf{pk}}^{s+\xi}(m_2 \cdot \mathsf{Enc}_{\mathsf{pk}}^{s}(m_1; r_1); r_3) \in \mathcal{M}_{s+2\xi}$ for any $m_1 \in \mathcal{M}_s, m_2 \in \mathcal{M}_{s+\xi}$ and $r_1, r_2 \in \mathcal{R}$, and for some $r_3 \in \mathcal{R}$, it is clear that the expressions :

$$\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha) := \prod_{t \in \mathbb{Z}_{\lambda_j}} (\beta_{jt})^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)}, 1 \leq j \leq \alpha - 1, \text{ and}$$

$$\mathrm{DB}_\alpha := \prod_{t \in \mathbb{Z}_{\lambda_\alpha}} (\beta_{\alpha t})^{\mathrm{DB}_{(\alpha-1)}(t)}$$

are well defined, and that $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha) \in \mathcal{C}_{s+(j-1)\xi} = \mathcal{M}_{s+j\xi}$, for all $j \in [1, \alpha - 1]$.

Remark. Both [Lip05] and [AIR01] propose the ElGamal cryptosystem to implement $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}$. It is worth noting however, that using plain ElGamal, it is not possible to compute $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(\delta_j(s-j) + \mathrm{DB}[j])$ given $\mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(s)$, since ElGamal is *only multiplicatively* homomorphic. We fix this problem in the next section.

3.4 Accredited SPIR based on the DL problem

The DL-based accredited SPIR scheme we propose, is achieved by combining the three building blocks above: Lipmaa's SPIR scheme, Brands's credentials, and the ElGamal cryptosystem.

Settings. The setting for the system we propose is the combination of the settings of each of the three building blocks. These were already introduced earlier in the chapter. In the following, we just highlight a few conditions (on the parameters of the building blocks) that need to be satisfied. Since we use the ElGamal cryptosystem to implement $\Pi_{\mathsf{hom}} := (\mathsf{Gen}_{\mathsf{hom}}, \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}, \mathsf{D}_{\mathsf{sk}_{\mathsf{hom}}})$, we require that:

- $G_q = \langle g_{\mathsf{ElG}} \rangle = \langle g_{db} \rangle = \langle g_i \rangle$, where $\{g_i | 0 \leq i \leq \ell\}$ is the discrete logarithm representation basis in the Brands scheme.
- $G_q \subseteq \mathcal{M}_{HomEnc_{pk}}$, where $\mathcal{M}_{HomEnc_{pk}}$ is the message space of $HomEnc_{pk}$.
- $\mathrm{DB}[j] \in G_q$, for $0 \leq j \leq |\mathrm{DB}| - 1$, and $n := |\mathrm{DB}| \leq q$.

Overview. We first give the main idea of the construction, before getting into the details. We assume the public parameters of the three building blocks are already known to all parties. Let $\mathrm{ID}_{\mathcal{A}}$ be an attribute, that uniquely identifies the Authorizer (e.g., an SSN). This $\mathrm{ID}_{\mathcal{A}}$ will determine the index by which the Receiver will query the Sender's database. Let us first assume that the Authorizer possesses a Br-DL-I credential of the form $(h, \sigma_{\mathrm{CA}}(h))$, where $h = (g_1^{\mathrm{ID}_{\mathcal{A}}} g_2^{x_2} \cdots g_\ell^{x_\ell} h_0)^\alpha$. The Authorizer computes $\mathsf{c} := \mathsf{E}_{\mathsf{pk}_{\mathsf{hom}}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}}) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}}) := (\mathsf{c}_1, \mathsf{c}_2)$, where $\mathsf{pk}_{\mathsf{ElG}}$ is the Receiver's ElGamal public key, and g_{db} is a public generator of G_q chosen by the Sender. Next, the Authorizer produces a signed proof of knowledge asserting that the logarithm, to the base g_{db}, of the plaintext encoded in c, is the same as the first attribute embedded in credential h. We call this last assertion an ID-consistency proof.

Notice that this last proof cannot be done in a straightforward way using the original Br-DL-I credentials, because h has the form $h := g_1^{\beta_1} \cdots g_\ell^{\beta_\ell} h_0^\alpha$, where $\beta_i = \alpha x_i \mod q$ for some random blinding factor α. Establishing ID-consistency in this case requires proving a non-linear predicate on secret exponents $(\alpha, \beta_1, \mathrm{ID}_{\mathcal{A}})$, defined by $\mathcal{P} \equiv$ "$\beta_1 = \alpha \times \mathrm{ID}_{\mathcal{A}} \mod q$", which cannot be done efficiently with available state-of-the-art techniques. To fix this problem we propose a modified version of the Br-DL-I credentials, with exactly the same security and privacy properties. In the modified version, the credential's public key h is computed as $h := (g_1^{x_1} \cdots g_{\ell-1}^{x_{\ell-1}} g_\ell^{\alpha} h_0)$, where $x_1, \cdots, x_{\ell-1}$ are identity attributes, and α is a secret random blinding factor chosen by the credential holder. This modification is of general interest, and can be used in other contexts as well. A summary of the modified credential system is outlined in Figure 3.4.

User U	Public Info	CA
	$(g_i)_{0\le i\le\ell}, h_0, (g_i^{x_0})_{0\le i\le\ell}, h_0^{x_0}, q, G_q, H$	
	$\xleftrightarrow{\text{agree on attributes } (x_1,\cdots,x_{\ell-1})}$	$w_0 \in_R \mathbb{Z}_q$
		$a_0 := g_0^{w_0}, a_\ell := g_\ell^{w_0}$
$\alpha_1 \in_R \mathbb{Z}_q^*,$	$\xleftarrow{a_0,b_0,a_\ell}$	$b_0 := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{w_0}$
$\alpha_2, \alpha_3 \in_R \mathbb{Z}_q,$		
$h := g_1^{x_1} \cdots g_{\ell-1}^{x_{\ell-1}} h_0 g_\ell^{\alpha_1}$		
$z' := (g_1^{x_1} \cdots g_{\ell-1}^{x_{\ell-1}} h_0 g_\ell^{\alpha_1})^{x_0}$		
$a'_0 := h_0^{\alpha_2} g_0^{\alpha_3} a_0$		
$b'_0 := (z')^{\alpha_2} h^{\alpha_3} (b_0 a_\ell^{\alpha_1})$		
$c'_0 := H(h, z', a'_0, b'_0)$		
$c_0 := c'_0 + \alpha_2 \mod q$	$\xrightarrow{c_0}$	
	$\xleftarrow{r_0}$	$r_0 := c_0 x_0 + w_0 \mod q$
$r'_0 := r_0 + \alpha_3 \mod q$		
accept iff $a'_0 b'_0 = (g_0 h)^{r'_0} (h_0 z')^{-c'_0}$		
store $h, \sigma_{\mathrm{CA}}(h) := (z', r'_0, c'_0)$		

Figure 3.4: Modified version of the Br-DL-I Credential Issuing protocol

In what follows, we assume the Authorizer possesses a credential of the new type. Let us denote the public key of the Authorizer's new credential by

$$h := (g_1^{\mathrm{ID}_{\mathcal{A}}} \cdots g_{\ell-1}^{x_{\ell-1}} g_\ell^{\alpha_1} h_0)$$

To prove ID-consistency between h and the SPIR query $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{EIG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$, it suffices for the Authorizer to produce a signed proof of knowledge of a DL-representation of $(h/\mathsf{c}_2 \mod p)$ with respect to basis $((g_1 g_{db}^{-1}), g_2, \cdots, g_\ell, h_0, g_{\mathsf{EIG}}, y_{\mathsf{EIG}})$. This is done using the same method of Figure 3.2. We denote the latter signed proof by

$$\mathrm{SPK}\{(\varepsilon_1, \cdots, \varepsilon_\ell, \mu, \nu) : h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0 \wedge \mathsf{c}_2 = y_{\mathsf{EIG}}^{\mu} g_{db}^{\nu} \wedge \varepsilon_1 = \nu\}(m)$$

As shown in Figure 3.2, the message m to be signed, can be a concatenation of several fields, including a fresh nonce. In addition, it may contain the identity of the Receiver, which will allow the Authorizer to exclusively tie the authorization to the Receiver, and discourage him from sharing it with other parties. The Authorizer may also include an expiry date in m to make sure his authorization remains valid only for the appropriate amount of time. More generally, the Authorizer may encode in m any application-specific policy he wants the Receiver to follow.

Running the SPIR. Now let us assume the signed proof above was accepted. The next step would be for the Receiver to compute the query messages as indicated in the OT scheme of Figure 3.3. First let $(\mathrm{ID}_{\mathcal{A}}^{(1)}, \cdots, \mathrm{ID}_{\mathcal{A}}^{(\alpha)})$ be the representation of the Authorizer's $\mathrm{ID}_{\mathcal{A}}$ in the α-dimensional hyper-rectangle $\lambda_1 \times \cdots \times \lambda_\alpha$ used by the Sender's database. The Receiver then computes, for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, the homomorphic encryptions $\beta_{jt} := HomEnc_{pk}(b_{jt})$, where $b_{jt} = 1$ if $t = \mathrm{ID}_{\mathcal{A}}^{(j)}$, and $b_{jt} = 0$ otherwise. Next, the Receiver submits the following to Sender:

- the credential $(h, \sigma_{\mathrm{CA}}(h))$,
- the first part of the query $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$,
- an ID-consistency proof SPK$\{(\varepsilon_1, \cdots, \varepsilon_\ell, \mu, \nu) : h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0 \wedge \mathsf{c}_2 = y_{\mathsf{ElG}}^{\mu} g_{db}^{\nu} \wedge \varepsilon_1 = \nu\}(m)$,
- the second part of the query consisting of the β_{jt}'s for $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$.

Note that there is no need for the Receiver to prove consistency between the β_{jt}'s and $(\mathsf{c}_1, \mathsf{c}_2)$. As we will show later, any attempt by the Receiver to incorrectly compute the β_{jt}'s, will prevent him from learning anything meaningful at the end of the SPIR protocol.

Once the ID-consistency check succeeds, the Sender starts processing the Receiver's query as explained in the following. But first, we make a few practical assumptions. We assume that n, the size of the Sender's database, is bounded above by q, the order of G_q. In practice, q is chosen to be at least 160-bit long, which means the Sender's database could have up to 2^{160} different records. Although we think this should be sufficient in practice, the size of q can always be increased if needed. Moreover, we assume that each record DB$[i]$ contains a field for storing $(g_{db}^{-i} \mod p)$, in addition to a large field containing application-specific data (e.g., health, financial data).

Next the query is processed as follows. Using $(\mathsf{c}_1, \mathsf{c}_2)$, and the *multiplicatively* homomorphic properties of ElGamal, the Sender computes, for $j \in [1, n]$, $\mathrm{DB}_0[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_j(\mathrm{ID}_{\mathcal{A}} - j)} \times \mathrm{DB}[j])$, where δ_j is a blinding factor, chosen by the Sender, uniformly at random in $[1, q-1]$. This is done as follows. Given $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}})$, the sender computes $\mathrm{DB}_0[j] := \big(\mathsf{c}_1{}^{\delta_j}, (\mathsf{c}_2 \times g_{db}^{-j})^{\delta_j} \times \mathrm{DB}[j]\big)$. Note that g_{db}^{-j} has already been precomputed and stored in with DB's j^{th} record.

Remark on Notation. We sometimes use a more compact notation to describe operations on the ciphertexts. For example, we use the $\otimes$ and $\odot$ operators as follows. For $m, m' \in G_q$, $x \in \mathbb{N}$, and for $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m_1)$, and $(\mathsf{c}'_1, \mathsf{c}'_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m')$, the expressions $\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m) \otimes m'$, $\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m) \odot \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m')$, and $(\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m))^x$ are interpreted respectively as:

$$\begin{aligned} \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m) \otimes m' &\overset{\text{def}}{=} (\mathsf{c}_1, \mathsf{c}_2 \times m'), \\ \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m) \odot \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m') &\overset{\text{def}}{=} (\mathsf{c}_1 \times \mathsf{c}'_1, \mathsf{c}_2 \times \mathsf{c}'_2), \text{ and} \\ (\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m))^x &\overset{\text{def}}{=} \underbrace{\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m) \odot \cdots \odot \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(m)}_{x \text{ times}} \end{aligned}$$

In particular, the encryption $\mathrm{DB}_0[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}\left((g_{db})^{\delta_j(\mathrm{ID}_{\mathcal{A}}-j)} \times \mathrm{DB}[j]\right)$ can be computed as:[4]

$$\mathrm{DB}_0[j] \quad = \quad ((\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}}) \otimes g_{db}^{-j})^{\delta_j} \otimes \mathrm{DB}[j])$$

◊

Next, the Sender proceeds with computing DB_α by repeated encryptions of the records of DB_0, as indicated in Figure 3.5. That is, for $j := 1$ to $\alpha - 1$, and for

[4] We use this shorthand notation in Figure 3.5

$(i_{j+1}, \cdots, i_\alpha) \in [0, \lambda_{j+1} - 1] \times \cdots \times [0, \lambda_\alpha - 1]$, the Sender computes

$$\begin{aligned}
\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha) &:= \prod_{t \in \mathbb{Z}_{\lambda_j}} (\beta_{jt})^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)} \\
&= \left(\prod_{t \in \mathbb{Z}_{\lambda_j}, t \neq \mathrm{ID}_{\mathcal{A}}^{(j)}} (\beta_{jt})^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)} \right) \times (\beta_{j\mathrm{ID}_{\mathcal{A}}^{(j)}})^{\mathrm{DB}_{j-1}(\mathrm{ID}_{\mathcal{A}}^{(j)}, i_{j+1}, \cdots, i_\alpha)} \\
&= \left(\prod_{t \in \mathbb{Z}_{\lambda_j}, t \neq \mathrm{ID}_{\mathcal{A}}^{(j)}} (HomEnc_{pk}(0))^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)} \right) \times \\
&\qquad (HomEnc_{pk}(1))^{\mathrm{DB}_{j-1}(\mathrm{ID}_{\mathcal{A}}^{(j)}, i_{j+1}, \cdots, i_\alpha)} \\
&= HomEnc_{pk} \left(\mathrm{DB}_{j-1}(\mathrm{ID}_{\mathcal{A}}^{(j)}, i_{j+1}, \cdots, i_\alpha) \right)
\end{aligned}$$

A consequence of the equation above is that, for $j := 1$ to $\alpha - 1$, DB_j is a $(\alpha - j)$-dimensional database, containing $\lambda_{j+1} \times \cdots \times \lambda_\alpha$ elements, and each of these elements $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha)$, for $(i_{j+1}, \cdots, i_\alpha) \in [0, \lambda_{j+1} - 1] \times \cdots \times [0, \lambda_\alpha - 1]$, is in fact a j-times encryption through $HomEnc_{pk}$ of $\mathrm{DB}_0(\mathrm{ID}_{\mathcal{A}}^{(1)}, \cdots, \mathrm{ID}_{\mathcal{A}}^{(j)}, i_{j+1}, \cdots, i_\alpha)$.

Similarly, DB_α which is computed as

$$\mathrm{DB}_\alpha := \prod_{t \in \mathbb{Z}_{\lambda_\alpha}} (\beta_{\alpha t})^{\mathrm{DB}_{(\alpha-1)}(t)} = HomEnc_{pk} \left(\mathrm{DB}_{\alpha-1}(\mathrm{ID}_{\mathcal{A}}^{(\alpha)}) \right)$$

is in fact an α-times encryption through $HomEnc_{pk}$ of $\mathrm{DB}_0(\mathrm{ID}_{\mathcal{A}}^{(1)}, \cdots, \mathrm{ID}_{\mathcal{A}}^{(\alpha)}) \overset{\text{def}}{=} \mathrm{DB}_0[\mathrm{ID}_{\mathcal{A}}]$. The Sender then sends DB_α to the Receiver, who recovers $\mathrm{DB}_0[\mathrm{ID}_{\mathcal{A}}]$ from it, by repeated decryptions $HomDec_{sk}(\cdot)$, where sk denotes the Receiver's secret key. Finally, the Receiver obtains the desired record $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$, by decrypting $\mathrm{DB}_0[\mathrm{ID}_{\mathcal{A}}]$ one more time using his ElGamal private key.[5] A summary of the protocol is given in Figure 3.5.

[5] Recall that $\mathrm{DB}_0[j] := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}\left((g_{db})^{\delta_j(\mathrm{ID}_{\mathcal{A}} - j)} \times \mathrm{DB}[j]\right)$, for $0 \leq j \leq |\mathrm{DB} - 1|$.

Public Info

$$p, q, (g_i)_{0 \le i \le \ell}, h_0, (g_i^{x_0})_{0 \le i \le \ell}, h_0^{x_0}, H, k, pk^{(\mathrm{R})}, \mathcal{R}, g_{db},$$

$$\mathsf{pk}_{\mathsf{ElG}}^{(\mathrm{R})} := (g_{\mathsf{ElG}}, y_{\mathsf{ElG}}), G_q := \langle g_i \rangle := \langle g_{\mathsf{ElG}} \rangle := \langle g_{db} \rangle, n := |\mathrm{DB}| \le q, \lambda_1, \cdots, \lambda_\alpha.$$

Authorizer **Receiver** **Sender (Database DB)**

Authorizer:

$$(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}^{(\mathrm{R})}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$$

$$\left.\xrightarrow[\substack{\mathrm{SPK}\{(\boldsymbol{\varepsilon}_1,\cdots,\boldsymbol{\varepsilon}_\ell,\boldsymbol{\mu},\boldsymbol{\nu}):\boldsymbol{h}=\boldsymbol{g}_1^{\boldsymbol{\varepsilon}_1}\cdots\boldsymbol{g}_\ell^{\boldsymbol{\varepsilon}_\ell}\boldsymbol{h}_0 \\ \wedge\mathsf{c}_2=\boldsymbol{y}_{\mathsf{ElG}}^{\boldsymbol{\mu}}\boldsymbol{g}_{db}^{\boldsymbol{\nu}}\wedge\boldsymbol{\varepsilon}_1=\boldsymbol{\nu}\}(\boldsymbol{m})}]{\boldsymbol{h}, \boldsymbol{\sigma}_{\mathrm{CA}}(\boldsymbol{h}) := (\boldsymbol{z}', \boldsymbol{r}_0', \boldsymbol{c}_0'), (\mathbf{c}_1, \mathbf{c}_2)}\right\} \textbf{Authorization}$$

Receiver:

```
For $j := 1$ to $\alpha$ do :
    For $t := 0$ to $\lambda_j - 1$ do :
        $r_{jt} \in_R \mathcal{R}$
        $\beta_{jt} := HomEnc_{pk^{(\mathrm{R})}}(b_{jt}, r_{jt})$,
            where $b_{jt} := 1$ if $t = \mathrm{ID}_{\mathcal{A}}^{(j)}$,
            and $b_{jt} := 0$ otherwise.
```

$$\xrightarrow{\textbf{Authorization}, \{\boldsymbol{\beta}_{jt}\}_{0 \le t < \lambda_j}^{1 \le j \le \alpha,}}$$

Sender (Database DB):

```
Check Authorization validity.
For $j := 1$ to $n$ do :
    $\delta_j \in_R [1, q-1]$
    $\mathrm{DB}_0[j] :=$
        $((\mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}^{(\mathrm{R})}}(g_{db}^{\mathrm{ID}_{\mathcal{A}}}) \otimes g_{db}^{-j})^{\delta_j} \otimes \mathrm{DB}[j])$.[4]
For $j := 1$ to $\alpha - 1$ do :
    For $i_{j+1} := 0$ to $\lambda_{j+1} - 1, \cdots,$
                    $i_\alpha := 0$ to $\lambda_\alpha - 1$ do :
    $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha) :=$
            $\prod_{t \in \mathbb{Z}_{\lambda_j}} (\beta_{jt})^{\mathrm{DB}_{j-1}(t, i_{j+1}, \cdots, i_\alpha)}$
$\mathrm{DB}_\alpha := \prod_{t \in \mathbb{Z}_{\lambda_\alpha}} (\beta_{\alpha t})^{\mathrm{DB}_{(\alpha-1)}(t)}$
```

$$\mathrm{DB}'_\alpha := \mathrm{DB}_\alpha \quad \xleftarrow{\quad \mathbf{DB}_{\boldsymbol{\alpha}} \quad}$$

Receiver:

```
For $j := \alpha$ downto $1$ do :
    $\mathrm{DB}'_{j-1} := HomDec_{sk^{(\mathrm{R})}}(\mathrm{DB}'_j)$
Output $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}] := \mathsf{D}_{\mathsf{sk}_{\mathsf{ElG}}^{(\mathrm{R})}}(\mathrm{DB}'_0)$
```

Figure 3.5: Accredited Symmetrically Private Information Retrieval : DL-based Construction

3.5 Security definitions

We first introduce the notion of *Ciphertext Indistinguishability under Chosen Plaintext Attacks*, or IND-CPA for short. IND-CPA is one of many definitions, available in the literature (e.g., [Gol01]), attempting to capture the meaning of security for an encryption scheme.[6]

More specifically, let us consider the following experiment defined for a public key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and adversary $\mathcal{B}$:

The CPA indistinguishability experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{B},\Pi}(k)$:

1. $\mathsf{Gen}(1^k)$ is run to obtain a public key/private key pair (pk, sk).
2. Adversary $\mathcal{B}$ is given pk as well as an oracle access to $\mathsf{Enc}_{pk}(\cdot)$.

   The adversary outputs a pair of messages m_0, m_1 of the same length, from the message space of Enc.
3. A bit $b \in_R \{0, 1\}$ is chosen uniformly at random, and then a ciphertext $c := \mathsf{Enc}_{pk}(m_b)$ is computed and given to $\mathcal{B}$. We call c the challenge ciphertext.
4. $\mathcal{B}$ continues to have access to $\mathsf{Enc}_{pk}(\cdot)$, and outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Definition 3.5.1 (Ciphertext Indistinguishability under Chosen Plaintext Attacks (IND-CPA))**.** *A public-key encryption scheme* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to have* indistinguishable encryptions under a chosen plaintext attack *(or is* IND-CPA secure*) if for*

[6] There are two main reasons why we are using this particular definition here: (1) because we are using the ElGamal public-key cryptosystem as a building block in our constructions, and *IND-CPA* is the minimum level of security one could expect a public-key cryptosystem to satisfy; and (2) because the ElGamal cryptosystem is known to satisfy IND-CPA security, but not stronger notions of security such as *IND-CCA* (*Ciphertext Indistinguishability under Chosen Ciphertext Attacks*) or *Non-Malleability* [Gol01] etc.

all probabilistic polynomial-time adversaries $\mathcal{B}$ there exists a negligible function $\mathsf{negl}(\cdot)$ *such that:*

$$\left| Prob[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{B},\Pi}(k) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(k).$$

Next we define the notion of α-IND-LFCPA security; the counterpart of IND-CPA security for LFAH cryptosystems. We first consider the following experiment defined for a LFAH public key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and adversary $\mathcal{B}$:

α-IND-LFCPA experiment $\mathsf{PubK}^{\mathsf{lfcpa}}_{\mathcal{B},\Pi}(k)$:

1. $\mathsf{Gen}(1^k)$ is run to obtain a public key/private key pair $(\mathsf{pk}, \mathsf{sk})$.
2. The Adversary $\mathcal{B}$ is given pk, and returns α length parameters $s_1, \cdots, s_\alpha$, and a pair of messages m_0, m_1 of the same length, from the message space $\mathcal{M}_{s_1}$ of $\mathsf{Enc}^{s_1}_{\mathsf{pk}}$. Note that for $s_1, \cdots, s_\alpha$ such that $s_1 \leq \cdots \leq s_\alpha$, we have $\mathcal{M}_{s_1} \subseteq \cdots \subseteq \mathcal{M}_{s_\alpha}$.
3. A bit $b \in_R \{0, 1\}$ is chosen uniformly at random, and then the ciphertexts $c_1 := \mathsf{Enc}^{s_1}_{\mathsf{pk}}(m_b; \mathcal{R}), \cdots, c_\alpha := \mathsf{Enc}^{s_\alpha}_{\mathsf{pk}}(m_b; \mathcal{R})$ are computed and given to $\mathcal{B}$. We call $c_1, \cdots, c_\alpha$ the challenge ciphertexts.
4. $\mathcal{B}$ outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Definition 3.5.2 (**α**-IND-LFCPA). *An LFAH public-key cryptosystem* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to be secure in the sense of α-IND-LFCPA, if for all probabilistic polynomial-time adversaries $\mathcal{B}$ there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\left| Prob[\mathsf{PubK}^{\mathsf{lfcpa}}_{\mathcal{B},\Pi}(k) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(k).$$

In [Lip05], Lipmaa gives arguments about the α-IND-LFCPA security of the Damgaard-Jurik cryptosystems [DJ01, DJ03], and suggests using them as possible implementations for the LFAH cryptosystem $\pi := (HomGen, HomEnc_{pk}, HomDec_{sk})$.

Now we define one last property related to ciphertext indistinguishability under Chosen Plaintext Attacks. We call this property *Strong α-IND-LFCPA security*. Informally, this property states that the juxtaposition, side by side, of an IND-CPA-secure cryptosystem, and an α-IND-LFCPA secure cryptosystem, results in a cryptosystem that *still* retains the IND-CPA property.

More precisely, let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key IND-CPA cryptosystem, and $\Pi^{\alpha} = (\mathsf{LFGen}, \mathsf{LFEnc}, \mathsf{LFDec})$ be a public key α-IND-LFCPA cryptosystem. Now let $\Pi^{s} = (\mathsf{SGen}, \mathsf{SEnc}, \mathsf{SDec})$ be the combination, by juxtaposition, side by side, of Π and Π^{α}. The resulting cryptosystem Π^{s} is such that $\mathsf{SGen} = (\mathsf{Gen}, \mathsf{LFGen})$, $\mathsf{SEnc} = (\mathsf{Enc}, \mathsf{LFEnc})$, and $\mathsf{SDec} = (\mathsf{Dec}, \mathsf{LFDec})$. Let us recall also that plaintext messages can be represented in an α-dimensional hyper-rectangle $\lambda_1 \times \cdots \times \lambda_{\alpha}$ (as described in the previous section). We consider the following game for the cryptosystem Π^{s} and an adversary $\mathcal{B}$.

Strong α-IND-LFCPA experiment $\mathsf{PubK}^{\mathsf{Strong\ lfcpa}}_{\mathcal{B},\Pi^{s}}(k)$:

1. $\mathsf{Gen}(1^k)$ and $\mathsf{LFGen}(1^k)$ are run to obtain public key/private key pairs $(\mathsf{pk}, \mathsf{sk})$ and $(\mathsf{pk}_{\alpha}, \mathsf{sk}_{\alpha})$ respectively.

2. The Adversary $\mathcal{B}$ is given $(\mathsf{pk}, \mathsf{pk}_{\alpha})$, and he returns a pair of messages m_0, m_1 of the same length, from the message space of Enc.

3. A bit $b \in_R \{0, 1\}$ is chosen uniformly at random, and then the ciphertext $c := \mathsf{Enc}_{pk}(m_b)$ is computed.

4. Let $(m_b^{(1)}, \cdots, m_b^{(\alpha)})$ denote the representation of m_b in the α-dimensional hyperrectangle $\lambda_1 \times \cdots \times \lambda_\alpha$. For $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, the encryptions $c_{jt} := \mathsf{LFEnc}_{\mathsf{pk}_\alpha}^{s_j}(a_{jt}; \mathcal{R})$ are computed, where $a_{jt} = 1$ if $t = m_b^{(j)}$, and $a_{jt} = 0$ otherwise.[7]

5. Next, the encryptions c as well as c_{jt}, $1 \leq j \leq \alpha, 0 \leq t < \lambda_j$, are all given to $\mathcal{B}$.

6. $\mathcal{B}$ outputs a bit b'.

7. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

Definition 3.5.3 (Strong $\boldsymbol{\alpha}$-IND-LFCPA)**.** *A public-key cryptosystem Π^s, consisting of a juxtaposition side-by-side of an IND-CPA secure cryptosystem Π and an α-IND-LFCPA secure cryptosystem Π^α, is said to be secure in the sense of* Strong α-IND-LFCPA, *if for all probabilistic polynomial-time adversaries $\mathcal{B}$ there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that:*

$$\left| Prob[\mathsf{PubK}_{\mathcal{B},\Pi^s}^{\mathsf{Strong\ lfcpa}}(k) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(k).$$

Finally we highlight *some* of the security properties that a privacy-preserving credential system should have, and point out how these properties apply to the case of the Br-DL system. We have already discussed the Br-DL system in Chapter 2. Here we only recall some of the security and privacy properties of the Br-DL scheme, that we need in our analysis of the ASPIR protocol. The analysis of the ASPIR protocol is given in section 3.6.

Some of the properties of a secure privacy-preserving credential system A secure and privacy-preserving credential system should (at least) satisfy the following properties:

[7] The s_j's are public length parameters corresponding to LFEnc. For $1 \leq j \leq \alpha$, $s_j := s + (j-1)\xi$, where s and ξ are public length parameters generated by LFGen. Details on these parameters have already been given on page 64.

1. **Credential unforgeability:** This property states that a computationally limited adversary can forge new credentials on behalf of the certification authority, only with negligible probability. In the case of the Br-DL credential scheme, Brands proves credential unforgeability under the DL assumption, in the Random Oracle Model [KL07, Chap 13]. The details of the proof can be found in [Bra00, Proposition 4.3.7].

2. **Selective disclosure and attribute hiding:** This property allows a credential holder to keep hidden the identity attributes in his credential, and to selectively disclose any predicate on those attributes. This is generally achieved using Zero-Knowledge proofs [Gol01]. Informally, in a Zero-Knowledge proof, any information that a verifier can learn by engaging with the prover, she can also learn it from merely the system parameters, the prover's public information, her a priori knowledge, and and the status of the formulae requested. In the context of the Br-DL credential system, Brands proves that the credential showing protocol of the Br-DL system is Honest-Verifier Zero-Knowledge under the DL assumption, in the Random Oracle Model. That is, the verifier is assumed to be semi-honest (or honest but curious). Brands also proves a weaker property; namely that the showing protocol is *witness-indistinguishable* (See [Gol01, Section 4.6] for a definition). The security proofs of the Brands system can be found in [Bra00, Propositions 3.3.4 and 3.3.5].

3. **Proof unforgeability:** This property states that it is hard for a computationally limited adversary to forge signed proofs of knowledge (i.e., credential showings) for a credential he does not own. Brands proves unforgeability under the DL assumption, in the Random Oracle Model. The proof is given in [Bra00, Propositions 3.3.6].

3.6 Security and privacy properties of the ASPIR protocol

3.6.1 Definitions

In order to define the security of the ASPIR scheme, we need to address three different aspects: privacy for the Receiver, privacy for the Sender, and security for the Data-subject (or Authorizer). Our first definition deals with *privacy for the Receiver*, and follows the same approach as Naor and Pinkas in [NP01, NP99].

Definition 3.6.1 (Privacy for the Receiver – Indistinguishability). *Let n denotes the size of the Sender's database DB. For any pair of indexes $i, j \in [0, n-1]$, and for any probabilistic polynomial time adversary $\mathcal{B}$ playing the Sender's role, the views that $\mathcal{B}$ sees in case the Receiver tries to obtain DB$[i]$, and in case he tries to obtain DB$[j]$, are computationally indistinguishable given DB$[0], \cdots,$ DB$[n-1]$.*

The Sender's privacy is slightly more elaborate than that of the Receiver. Since the Receiver gets some information at the end of the protocol, we want to make sure that he does not get more, or different, information from what he should. In particular, in the context of the ASPIR protocol, we want the Receiver to successfully retrieve the target record when he submits a valid[8] query to the Sender, and to obtain a uniformly random value otherwise. There are at least two approaches to formally express the notion of *privacy for the Sender*.

The first approach, used by Naor and Pinkas in [NP01, NP99] and mentioned by Canetti and Goldwasser in [CG99], consists in making a comparison between the view of an adversarial Receiver in a real execution of the ASPIR protocol, and

[8] A query is valid if it is computed as prescribed in the protocol of Fig 3.5.

the view of an adversary emulating the Receiver in an *ideal implementation*. The ideal implementation uses a trusted third party TTP that receives the Sender's input DB$[0], \cdots,$ DB$[n-1]$, and the Receiver's query i, and returns DB$[i]$ to the Receiver. The precise requirement can be formulated as follows:

Definition 3.6.2 (Privacy for the Sender – Comparison with the Ideal Model)**.** *For every probabilistic polynomial time adversary $\mathcal{B}$ substituting the Receiver, there exists a probabilistic polynomial time machine $\mathcal{B}'$ that plays the Receiver's role in the ideal model such that the outputs of $\mathcal{B}$ and $\mathcal{B}'$ are statistically indistinguishable to a distinguisher that is given $DB[0], \cdots, DB[n-1]$.*

The second approach, to defining the Sender's privacy, simply states that any malicious Receiver that does not follow the protocol of Fig 3.5 and submits an invalid query, can at best recover a uniformly random value. More precisely, the requirement is as follows:

Definition 3.6.3 (Privacy for the Sender – Uniformly Random Output for Bad Queries)**.** *For every adversary $\mathcal{B}$ playing the Receiver's role, if $\mathcal{B}$ submits an* invalid *query to the Sender, then $\mathcal{B}$'s output is statistically indistinguishable from the uniform distribution.*

In our analysis of privacy for the Sender, we use the second approach (Def. 3.6.3). The first approach may seem more general but we could not use it in our context. In [NP99], Naor and Pinkas were able to use a security definition based on the real-vs-ideal model comparison (ie., similar to Def. 3.6.2), for the following reasons. In their protocol (an OT$^2_1$), the Receiver gets, in response to his query, two ciphertexts: $c_0 = H(K_0) \oplus M_0$ and $c_1 = H(K_1) \oplus M_1$, such that he can only compute one of the K_i's. The terms M_0 and M_1 denote the Sender's messages, and $H(\cdot)$ denotes a public hash function. Because [NP99] assumes $H(\cdot)$ a random oracle, the authors there are able to build, for any real-world adversary $\mathcal{B}$, an adversary $\mathcal{B}'$ operating in the ideal model, such that their outputs are indistinguishable from each other. This is done as follows.

$\mathcal{B}'$ plays the role of the Sender in the real world, and that of the Receiver in the ideal world. In response to a query from $\mathcal{B}$, $\mathcal{B}'$ returns two random values α_0 and α_1 (as if they were c_0 and c_1). Since we are in the random oracle model, $\mathcal{B}'$ is able to monitor $\mathcal{B}'$s queries to $H(\cdot)$. When $\mathcal{B}$ queries $H(\cdot)$ on K_b (for $b \in \{0, 1\}$), $\mathcal{B}'$ queries the TTP on M_b, sets $H(K_b) := \alpha_b \oplus M_b$ and returns it to $\mathcal{B}$; otherwise $\mathcal{B}'$ returns a random value. Naor and Pinkas show in [NP99] that an adversary $\mathcal{B}'$ constructed this was has an output that is indistinguishable from that of $\mathcal{B}$.

Unfortunately, this method does not seem to be suitable for our context. For example, we cannot take advantage of the random oracle monitoring trick because we do not use a hash function to encrypt responses to the Receiver's queries. The method of Naor and Pinkas could be used in our context however, if we assume that $HomEnc_{pk}(\cdot)$ and $\mathsf{D}_{\mathsf{sk}^{(\mathrm{R})}_{\mathsf{ElG}}}(\cdot)$ are decryption oracles, but this may not be a reasonable assumption.

Finally, the definition of security for the data-subject follows straightforwardly from the informal description given earlier in the chapter.

Definition 3.6.4 (Security for the Data-subject – Authorization Unforgeability)**.** *Assuming the Sender honestly follows the prescribed protocol, and does not collude with the Receiver, then a probabilistic polynomial-time adversarial Receiver should not, with a non-negligible probability of success, be able to get $DB[ID_{\mathcal{A}}]$ without an authorization from the data-subject $ID_{\mathcal{A}}$.*

Definition 3.6.5 (Secure Accredited SPIR). *An Accredited SPIR scheme is said to be secure if it satisfies the requirements of definitions 3.6.1, 3.6.3, and 3.6.4.*

3.6.2 Analysis

Theorem 3.1 (Privacy for the Receiver). *Assuming the Br-DL credential system is secure, the juxtaposition of ElGamal and the LFAH cryptosystem $\pi := (HomGen, HomEnc_{pk}, HomDec_{sk})$) is secure in the sense of Strong α-IND-LFCPA, and that the Authorizer (Data-subject) does not collude with the Sender, the protocol of Figure 3.5 provides computational privacy for the Receiver.*

Proof arguments. Intuitively, the Receiver's privacy is ensured because of the following. The Sender's view consists of (1) a credential, (2) an ElGamal encryption, (3) a signed proof of knowledge of the attributes embedded in the credential, and (4) a homomorphic encryption of an α-dimensional coordinate. Because the proof of knowledge is *Honest-Verifier Zero-Knowledge*[9] for any distribution of the attributes (cf. [Bra00, Prop. 3.3.4]), seeing the signed proof of knowledge, together with the credential, does not leak any extra information to the Sender (or rather, give him more computational ability) about the content of the Receiver's query. The Sender could only hope to extract information from the ElGamal ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ and the LFAH homomorphic encryptions β_{jt}'s. But if this were possible, then we would be breaking the Strong α-IND-LFCPA security of the combination of ElGamal and the LFAH cryptosystem π, which contradicts our assumption.

[9] When the verifier is *not* honest, the proof of knowledge cannot be proven Zero-Knowledge anymore. If *Zero-Knowledgeness* is necessary then one could use *fully* Zero-knowledge proofs of knowledge (e.g., [FO97, Bou00]) instead of Schnorr-like protocols, but the former tend to be less efficient. So far, there are no major attacks known against the Schnorr protocol[Sch91] or similar ones (e.g., Br-DL [Bra00]), and therefore it continues to be widely accepted as secure in practice, mainly because of its simplicity and efficiency.

Note that the ElGamal cryptosystem is already known to be IND-CPA secure [KL07] under the DDH assumption, and that the LFAH cryptosystem π (implemented in [Lip05] using the Damgaard-Jurik cryptosystems [DJ01, DJ03]) is argued to be α-IND-LFCPA secure [Lip05]. Although we could not prove it formally, but there are strong indications that the juxtaposition of these two cryptosystems is indeed Strong α-IND-LFCPA secure. So far we did not manage to find any evidence to the contrary.

Making a formal proof about the security of the combination may not be an easy task; we could not find any formal proofs in the literature about the security of a similar combinations of cryptosystems.

Notice also that the ciphertexts:

- $\mathrm{DB}_0[j],\ j \in [1, n]$,
- $\mathrm{DB}_j(i_{j+1}, \cdots, i_\alpha)$, $j \in [1, \alpha - 1]$, $(i_{j+1}, \cdots, i_\alpha) \in \mathbb{Z}_{\lambda_{j+1}} \times \cdots \times \mathbb{Z}_{\lambda_\alpha}$, and DB_α,

computed by the sender when processing a query, contain plaintexts that are different from those in $(\mathsf{c}_1, \mathsf{c}_2)$ or the β_{jt}'s, and therefore they dont weaken the security of the latter. ∎

Theorem 3.2 (Privacy for the Sender)**.** *Assuming that the Sender has a uniform source of randomness, and that the cryptosystem* $\pi := (HomGen,\ HomEnc_{pk},\ HomDec_{sk})$*) is Additively* Homomorphic, *the protocol of Figure 3.5 provides perfect privacy for the Sender.*

Proof. Recall that the Receiver (by construction) knows the Authorizer's identity, and the corresponding DB index $\mathrm{ID}_{\mathcal{A}}$. In particular, given a valid authorization from the data-subject (Authorizer), he can recover $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$ (this follows clearly from the correctness of the ASPIR protocol).

Let us assume now that an adversarial Receiver wants to gain more information about the content of DB, other than $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$. Let s be the index that the Receiver uses in the second part of the query (the first part of the query has been generated by an honest, non-colluding Authorizer, and contains the index $\mathrm{ID}_{\mathcal{A}}$). This implies that the value DB_{α} computed by the Sender and returned to the Receiver, is an α-time encryption, under $HomEnc_{pk}(.)$, of $\mathrm{DB}_0[s] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)} \times \mathrm{DB}[s])$, where δ_s is a secret random blinding factor chosen by the Sender. We have two cases here:

Case 1 (s $=$ ID$_{\mathcal{A}}$) : The only output the Receiver gets is an α-time encryption, under $HomEnc_{pk}(.)$, of $\mathrm{DB}_0[s] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(\mathrm{DB}[s]) = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}(\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}])$. Therefore no other information about the other DB entries is revealed.

Case 2 (s $\neq$ ID$_{\mathcal{A}}$) : The only output the Receiver gets is an α-time encryption, under $HomEnc_{pk}(.)$, of $\mathrm{DB}_0[s] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)} \times \mathrm{DB}[s])$, where δ_s is a random blinding factor uniformly chosen by the Sender.

Since g_{db} is a generator of G_q (the message space of the ElGamal cryptosystem), and since δ_s is chosen uniformly at random in $\mathbb{Z}_q^*$, we conclude that for $s \neq \mathrm{ID}_{\mathcal{A}}$, the quantity $(g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)}$ is distributed uniformly at random in G_q. Since all the DB entries are elements of G_q, this implies in turn that $(g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)} \times \mathrm{DB}[s]$ is also uniformly distributed in G_q, if $s \neq \mathrm{ID}_{\mathcal{A}}$. In particular, given a fixed value of s, $\mathrm{ID}_{\mathcal{A}}$, and the rest of the public parameters, for every possible assignment of $\mathrm{DB}[s]$ in G_q, there exists a unique value of δ_s in $\mathbb{Z}_q^*$ that is consistent with the expression $\mathrm{DB}_0[s] = \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\delta_s(\mathrm{ID}_{\mathcal{A}}-s)} \times \mathrm{DB}[s])$. This means that the obtained $\mathrm{DB}_0[s]$ decrypts to a uniformly distributed random quantity, that reveals nothing about any $\mathrm{DB}[s]$, for $s \neq \mathrm{ID}_{\mathcal{A}}$. This implies perfect privacy for the Sender.

∎

Theorem 3.3 (Security for the Authorizer)**.** *Assuming the Br-DL credential system is secure, and that the Receiver does not collude with the Sender, the protocol of Figure 3.5 provides computational security for the Authorizer (data-subject).*

Proof. Let $\mathcal{B}$ be a PPT adversary, playing the Receiver's role, that breaks the Authorizer's security. We show that $\mathcal{B}$ can be used to break the security of the Br-DL system by forging new credentials containing the Authorizer's identity.

More precisely, $\mathcal{B}$ is by assumption able to retrieve a record $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$, without obtaining an authorization from the data-subject $\mathrm{ID}_{\mathcal{A}}$. Recall that data-subject's authorization consists of the following items:

- A Br-DL credential $\{h, \sigma_{\mathrm{CA}}(h)\}$,
- An encryption $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$ of the data-subject's identity, and
- A signed proof of knowledge

$$\mathrm{SPK}\{(\varepsilon_1, \cdots, \varepsilon_\ell, \mu, \nu) : h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0 \wedge \mathsf{c}_2 = y_{\mathsf{ElG}}^{\mu} g_{db}^{\nu} \wedge \varepsilon_1 = \nu\}(m)$$

  asserting that the first attribute of credential $\{h, \sigma_{\mathrm{CA}}(h)\}$ contains the same identity encrypted in $(\mathsf{c}_1, \mathsf{c}_2)$.

Since we assume that the Sender does not collude with adversary $\mathcal{B}$, it must be the case that the query submitted by $\mathcal{B}$, to retrieve $\mathrm{DB}[\mathrm{ID}_{\mathcal{A}}]$, is a valid one. In particular, it must be the case that the query contains a valid credential $\{h, \sigma_{\mathrm{CA}}(h)\}$, with $\mathrm{ID}_{\mathcal{A}}$ as its first attribute. In other words, $\mathcal{B}$ must have succeeded in forging a Br-DL credential.

∎

Theorem 3.4. *Assuming the Br-DL credential system is secure, the juxtaposition of ElGamal and the LFAH cryptosystem* $\pi := (HomGen,\ HomEnc_{pk},\ HomDec_{sk}))$ *is secure in the* *sense of Strong α-IND-LFCPA, and that none of the three parties colludes with the other, the protocol of Figure 3.5 is a secure ASPIR scheme in the sense of Definition 3.6.5.*

Theorem 3.4 follows immediately from Theorems 3.1, 3.2, and 3.3.

3.6.3 Additional privacy for the Authorizer

User-centricity and policy enforcement. When issuing an authorization, the credential holder could specify in the message of the signed proof of knowledge, a set of rules that he wants the recipient of the authorization to comply with. For instance, he could specify an expiry date, an upper bound on the number of times the authorization is used, or any other usage policy. The Sender is *trusted* to refuse processing queries from a Receiver who does not satisfy the usage policy specified in the signature. Note that the above trust assumption is legitimate in this context, because in our threat model the Sender is assumed to *only* mount attacks that benefit him personally (e.g., attacks that allow him to learn the access pattern to the database, or the content of a SPIR query etc.) In particular, the Sender and Receiver roles are assumed to be adversarial with respect to each other, and therefore those parties are assumed not to collude with one another in our setting. This assumption is legitimate because otherwise a malicious Sender colluding with the Receiver can just give away the whole database to the latter; in that case *no cryptographic solution* can achieve the *policy enforcement* property, unless the records are stored in encrypted form on the database.

Revocability. The Authorizer may decide to revoke a previously issued authorization. This can be done anonymously as follows. The Authorizer first needs to prove knowledge of a representation of the credential used in the authorization to be revoked. To prevent the leakage of personal network information, this proof should be conducted over a physically anonymized channel (e.g., a MixNet [Cha81]). Once the proof of knowledge is accepted, the Authorizer requires the Sender (DB manager) to add the credential in question to a black list of revoked authorizations. Later, it is easy for the Sender to check whether the credential submitted in a query is on the black list or not. This can be done efficiently using hash tables for instance. Note that an authorization can be revoked *only* by its issuer, since it is infeasible for a computationally bounded adversary to find a discrete-log representation for a given credential public key.

Authenticated personal information retrieval. In the special case, where the Authorizer and Receiver are the same entity, the construction we propose provides the data-subject with a mechanism to retrieve *his own* personal data anonymously. In other words, the construction we propose ensures that the stored data can be retrieved only by its owner. The channel between the Receiver and Sender in this case has to be physically anonymized.

3.7 Performance analysis

The accredited SPIR construction of Figure 3.5 does not lead to a significant increase in computation and communication complexity, compared to the underlying SPIR scheme [Lip05]. If we assume the Authorizer has a credential with $(\ell - 1)$ attributes, then the added computation complexity can be summarized as follows. The Authorizer needs to make $(\ell + 6)$ offline exponentiations (all precomputable), while the Receiver and Sender, both need to make $(\ell + 8)$ online exponentiations. The complexity of these computations is negligible compared to the complexity of the underlying SPIR scheme which is linear in n, where n is the size of the database. In practice, ℓ is in the order of 20, whereas n is much larger ($n \approx 2^{160}$). In terms of communication complexity, both the Authorizer and Receiver need to send $(\ell{+}8)\log(n){+}5$ extra bits to the Receiver and Sender respectively. Again, this does not change the overall $\mathcal{O}\big(\log^2(n)\big)$ asymptotic communication complexity of the underlying SPIR scheme [Lip05].

3.8 Accredited SPIR based on RSA

The constructions we present in this section use a version of the Brands credentials based on the RSA representation problem [Bra00, Section 4.2.2]. We refer to this type of credentials as the Br-RSA credentials. For the sake of completeness, we briefly introduce them in the following subsections.

3.8.1 RSA-based Brands credentials

Settings. On input the security parameter κ, the credential issuer chooses the following :

- two κ-sized primes P and Q, and computes $N := PQ$.
- a prime v smaller than N, and co-prime to $\phi(N)$.
- random elements $(g_1, \cdots, g_\ell) \in_R (\mathbb{Z}_N^*)^\ell$
- a one-way hash function $\mathcal{H}(.) : \{0,1\}^* \rightarrow \mathbb{Z}_s$, for some s super-polynomial in κ.

The credential issuer makes the parameters $N, v, (g_1, \cdots, g_\ell)$, and $\mathcal{H}$ public, and keeps P and Q secret. In addition, the issuer chooses $x_0 \in_R \mathbb{Z}_v^*$, such that given $h_0 := x_0^v \mod N$, computing the v^{th} root of h_0 is hard. The issuer then publishes h_0 and keeps x_0 secret.

The RSA representation problem : Let $(\varepsilon_1, \cdots, \varepsilon_\ell)$ be a set of elements in $\mathbb{Z}_v$, and $\varepsilon_{\ell+1}$ be an element in $\mathbb{Z}_N^*$. Let $\gamma := g_1^{\varepsilon_1} \cdots, g_\ell^{\varepsilon_\ell} \varepsilon_{\ell+1}^v \mod N$. We say that $(\varepsilon_1, \cdots, \varepsilon_\ell, \varepsilon_{\ell+1})$ is an RSA representation of γ with respect to the basis $(g_1, g_2, \cdots, g_\ell, v)$. Brands shows in [Bra00, Section 2.3.3] that given a random $\gamma \in \mathbb{Z}_N^* \setminus \{1\}$, and a basis $(g_1, g_2, \cdots, g_\ell, v)$, the task of finding an RSA representation of γ with respect to $(g_1, g_2, \cdots, g_\ell, v)$ is at least as hard as breaking the corresponding instance of the RSA problem, i.e., finding v^{th} roots in $\mathbb{Z}_N^*$. Brands shows also, that given γ and a representation $(\varepsilon_1, \cdots, \varepsilon_\ell, \varepsilon_{\ell+1})$ of γ with respect to basis $(g_1, g_2, \cdots, g_\ell, v)$, the task of finding a second representation $(\mu_1, \cdots, \mu_\ell, \mu_{\ell+1})$ of γ with respect to the same basis is at least as hard as breaking the corresponding instance of the RSA problem.

Credential issuing. Assume after making the necessary identity checks, the certification authority accepts to issue a credential to the user. Let $(x_1, \cdots, x_\ell) \in (\mathbb{Z}_v^*)^\ell$ be the attributes the *CA* wants to encode in the credential, and let $h := g_1^{x_1} \cdots g_\ell^{x_\ell} \mod N$. The x_i's are known to both the user and the *CA*. The user then chooses a random blinding factor $\alpha_1 \in \mathbb{Z}_N^*$ and computes the credential's public key $h' := h\alpha_1^v$. The certification authority's digital signature on the credential is a pair $(c_0', r_0') \in \mathbb{Z}_s \times \mathbb{Z}_N^*$, satisfying the relation $c_0' = \mathcal{H}(h', {r_0'}^v (h_0 h')^{-c_0'})$. At the end of the issuing protocol, the certification authority knows neither h' nor the signature (c_0', r_0'). Figure 3.6 describes the Br-RSA issuing protocol.

Credential showing. Similar to the Br-DL-I system, a user can show his credential to a verifier, by first revealing the credential's public key h' and a corresponding *CA*-issued signature (c_0', r_0'). The verifier checks if the validity relation $c_0' \stackrel{?}{=} \mathcal{H}(h', {r_0'}^v (h_0 h')^{-c_0'})$ holds. Once the validity check succeeds, the user produces a signed proof of knowledge of an RSA representation of h' with respect to basis the $(g_1, \cdots, g_\ell, v)$. We denote

User U	Public Info	CA
	$N, v, h_0, (g_i)_{1 \leq i \leq \ell}, \mathcal{H}, s$	
$h := g_1^{x_1} \cdots g_\ell^{x_\ell} \mod N$	$\xleftrightarrow{\text{agree on attributes } (x_1, \cdots, x_\ell)}$	$h := g_1^{x_1} \cdots g_\ell^{x_\ell} \mod N$
$\alpha_1, \alpha_2 \in_R \mathbb{Z}_N^*,$	$\xleftarrow{a_0}$	$a_0 \in_R \mathbb{Z}_N^*$
$\alpha_3 \in_R \mathbb{Z}_v,$		
$h' := h{\alpha_1}^v \mod N$		
$c_0' := \mathcal{H}(h', {\alpha_2}^v (h_0 h)^{\alpha_3} a_0)$		
$c_0 := c_0' + \alpha_3 \mod v$	$\xrightarrow{c_0}$	
$r_0^v (h_0 h)^{-c_0} \overset{?}{=} a_0 \mod N$	$\xleftarrow{r_0}$	$r_0 := ((h_0 h)^{c_0} a_0)^{1/v} \mod N$
$r_0' := r_0 \alpha_2 \alpha_1^{c_0'} (h_0 h)^{(c_0' + \alpha_3) \text{ div } v} \mod N$		
store $h', \sigma_{\text{CA}}(h') := (c_0', r_0')$		

Figure 3.6: Br-RSA Credential Issuing protocol

this representation by $(x_1, \cdots, x_\ell, \alpha_1)$. The signed proof can also be computed with respect to a predicate $\mathcal{P}$ on exponents $(x_1, \cdots, x_\ell)$, agreed-upon by the user and the verifier at the time of the showing. Figure 3.7 sketches the basic Br-RSA credential showing protocol.

3.8.2 Combining ElGamal with Brands RSA-based Credentials

We assume the Authorizer holds a Br-RSA credential h', and certificate $\sigma_{\text{CA}}(h') := (c_0', r_0')$, with $h' = (g_1^{\text{ID}_\mathcal{A}} g_2^{x_2} \cdots g_\ell^{x_\ell} \alpha^v) \mod N$. Recall that $\text{ID}_\mathcal{A}, x_2 \cdots x_\ell$ are elements of $\mathbb{Z}_v$, and that $\text{ID}_\mathcal{A}$ represents the index of the Authorizer's record in the Sender's DB. In order to accommodate all possible DB indexes (spanning over $[1, n]$), the range $[1, v]$ of the attributes in the Br-RSA system, is chosen such that it contains $[1, n]$. In other words, the prime v should be greater than n. Moreover, our solution requires the Authorizer to send an encryption of the index of the targeted DB record, along

User U	Public Info	Verifier
	$N, v, h_0, (g_i)_{1\leq i\leq\ell}, \mathcal{H}, H, s$	
$w_1, \cdots, w_\ell \in_R \mathbb{Z}_v$		
$w_{\ell+1} \in_R \mathbb{Z}_N^*$		
$a := g_1^{w_1} \cdots g_\ell^{w_\ell} w_{\ell+1}^v \mod N$	$\xleftarrow{m}$	$m := nonce\Vert$*optional description*
$c := H(h', a, m)$		*of the purpose of the interaction*
$\forall\, 1 \leq i \leq \ell, r_i := cx_i + w_i \mod v$		
$r_{\ell+1} := \prod_{j=1}^{\ell} g_j^{(cx_j+w_j \text{ div } v)} x_{\ell+1}^c w_{\ell+1} \mod N$		
	$\xrightarrow{h',\sigma_{\text{CA}}(h'):=(c_0',r_0'),a,r_1,\cdots,r_{\ell+1}}$	accept iff
		$c_0' = \mathcal{H}(h', {r_0'}^v (h_0 h')^{-c_0'})$,
		and for $c := H(h', a, m)$,
		$\prod_{i=1}^{\ell} g_i^{r_i} r_{\ell+1}^v (h')^{-c} = a \mod N$ holds.

Figure 3.7: Br-RSA basic Credential Showing protocol

with the issued authorization. In the construction we propose, this encryption is performed using an ElGamal cryptosystem with a message space spanning a group G_q of prime order q (see Section 3.3.2 for details). In order to accommodate all possible DB indexes (ranging over $[1, n]$), we choose q to be grater than n. Finally, the factors P and Q of the modulus N in the Br-RSA system, are chosen to be greater than p.

To summarize, we have :

- Br-RSA parameters: $N = PQ$, with P and Q primes. Next v is chosen in $\mathbb{Z}_N^*$ with $gcd(v, \phi(N)) = 1$
- ElGamal parameters: q and $p = 2q + 1$ primes
- Conditions: Let $n := |\text{DB}|$, we should have $P > p$, $Q > p$, $v > n$, and $q > n$.

As in the first construction based on Br-DL-I credentials, the Authorizer (data subject) and Receiver use the ElGamal cryptosystem to compute the SPIR query. Assuming the same setting for the Sender and Receiver as in the first construction of Section 3.4, the Authorizer computes $(\mathsf{c}_1, \mathsf{c}_2) := \mathsf{E}_{\mathsf{pk}_{\mathsf{ElG}}}((g_{db})^{\mathrm{ID}_{\mathcal{A}}})$. To prove ID-consistency between h' and the SPIR query, it suffices for the Authorizer to produce a signed proof of knowledge of an RSA-representation of $(h' \times (\mathsf{c}_2 \mod p)^{-1} \mod N)$ with respect to basis $((g_1 g_{db}^{-1}), g_2, \cdots, g_\ell, y_{\mathsf{ElG}}^{-1}, v)$. In the following, we make a few observations to show that the parameter choices above are sound:

1. By construction $\mathsf{c}_2 \in G_q$ a subgroup of $\mathbb{Z}_p^*$, and thus $gcd(\mathsf{c}_2, p) = 1$. Moreover, primes P and Q are greater than p, which implies that $gcd(\mathsf{c}_2, P) = gcd(\mathsf{c}_2, Q) = 1$. Since $N = PQ$, we have that $gcd(\mathsf{c}_2, N) = 1$, and therefore $((\mathsf{c}_2 \mod p)^{-1} \mod N)$ exists.
2. For the same reasons, g_{db}, g_{ElG}, and y_{ElG} all have inverses modulo N.

The observations above clearly indicate that we are again in the settings of the Br-RSA credential system.

Putting the pieces together. As in the first construction in Section 3.4, the Authorizer proves ID-consistency between his credential and the generated SPIR query, by sending to the Receiver a signed proof of knowledge stating that the prover knows the attributes embedded in the credential, and that the value of the first attribute is the same as the index encoded in the attached SPIR query. More specifically, the Authorizer computes the following signed proof of knowledge

$$\mathsf{SPK}\{(\varepsilon_1, \cdots, \varepsilon_{\ell+1}, \mu) : h'{\mathsf{c}_2}^{-1} = (g_1 g_{db}^{-1})^{\varepsilon_1} g_2^{\varepsilon_2} \cdots g_\ell^{\varepsilon_\ell} (y_{\mathsf{ElG}}^{-1})^{\varepsilon_{\ell+1}} \mu^v \mod N\}(m)$$

This is done using the protocol in Figure 3.7. The Authorizer can use the message m to encode any usage policy he wants the Receiver to follow. For instance, the Authorizer may include in m the identity of the Receiver to exclusively tie the authorization to the latter, or an expiry date to make sure the authorization remains valid only for the desired amount of time.

After accepting the signed proof, the Receiver proceeds with the SPIR protocol in Figure 3.5 without any further changes.

3.8.3 Security and privacy properties

Theorem 3.5. *Assuming the Br-RSA credential system is secure, the juxtaposition of ElGamal and the LFAH cryptosystem* $\pi := (HomGen, HomEnc_{pk}, HomDec_{sk})$*) is secure in the sense of Strong* α*-IND-LFCPA, and that none of the three parties colludes with the other, the protocol of Section 3.8.2 is a secure ASPIR scheme in the sense of Definition 3.6.5.*

The proof of Theorem 3.5 relies on roughly the same arguments as Theorem 3.4, except that we are dealing here with the RSA representation problem, instead of the DL representation problem.

In addition to Theorem 3.5, the properties of *user-centricity*, *revocability*, and *authenticated PIR* described in Section 3.6.3, also apply for the new scheme.

3.8.4 Variant based on the Okamoto-Uchiyama cryptosystem

The construction of Section 3.8.2, can be modified by using the Okamoto-Uchiyama cryptosystem [OU98] instead of the ElGamal cryptosystem. The Okamoto-Uchiyama cryptosystem is a probabilistic public key cryptosystem whose security[10] is equivalent to the hardness factoring moduli of the form $n' = p'^2q'$, for primes p' and q'. The Okamoto-Uchiyama cryptosystem is additively homomorphic.

[10] The Okamoto-Uchiyama cryptosystem is *semantically* secure against *passive adversaries*. By definition, an encryption scheme Enc is *semantically* secure against *passive adversaries* if for all probabilistic polynomial time adversary $\mathcal{A}$, for sufficiently large k, there exists a negligible function negl such that $\left|\text{Prob}[\mathcal{A}(k, \mathsf{pk}, m_0, m_1, \mathsf{C} := \mathsf{Enc}_{\mathsf{pk}}(m_b)) = b] - \frac{1}{2}\right| \leq \mathsf{negl}(k)$, where public key pk and messages m_0, m_1 are chosen by a probabilistic generator, and b is chosen uniformly at random in $\{0, 1\}$.

Setting of the Okamoto-Uchiyama cryptosystem. Given security parameter κ, choose κ-sized primes p' and q', and let $n' = (p')^2q'$. The primes p' and q' are such that $gcd(p', q'-1) = gcd(q', p'-1) = 1$. Choose random $g \in \mathbb{Z}^*_{n'}$, and let $h = g^{n'} \mod n'$. The tuple (n', g, h, κ) is published as the public key, while (p', q') are kept secret. To encrypt a message $0 < m < 2^{\kappa-1}$, select random $r \in \mathbb{Z}_{n'}$, and compute $\mathsf{E}_{\mathsf{pk_{OU}}}(m, r) := g^m h^r \mod n'$. The decryption function of the Okamoto-Uchiyama cryptosystem uses a special "logarithmic function" to recover the plaintext from a ciphertext. More details are given in [OU98].

Putting the pieces together. The Authorizer uses the Receiver's public key (in this case the Okamoto-Uchiyama public key) to produce the SPIR query and prove ID-consistency between the latter and an Authorizer's credential. Let $\mathsf{c} := \mathsf{E}_{\mathsf{pk_{OU}}}(\mathrm{ID}_{\mathcal{A}}, r) = g^{\mathrm{ID}_{\mathcal{A}}} h^r \mod n'$ be a randomized encryption of the Authorizer's ID. Moreover, let the pair $(h'$, $\sigma_{\mathrm{CA}}(h'))$ be the Authorizer's Br-RSA credential, with $h' = (g_1^{\mathrm{ID}_{\mathcal{A}}} g_2^{x_2} \cdots g_\ell^{x_\ell} \alpha^v) \mod N$. Since the attributes in the Brands credential have to be in $\mathbb{Z}^*_v$, and since the DB index $\mathrm{ID}_{\mathcal{A}}$ can be in the range $[1, n]$, we need to take $v > n$ in order to accommodate all possible indexes in the database. Moreover, we also choose P and Q, the co-factors of N, to be greater than the Okamoto-Uchiyama modulus n'. The latter choice is made to make sure that $\mathsf{c} := \mathsf{E}_{\mathsf{pk_{OU}}}(\mathrm{ID}_{\mathcal{A}}, r) = g^{\mathrm{ID}_{\mathcal{A}}} h^r \mod n'$ has an inverse modulo N. Also, since the Okamoto-Uchiyama cryptosystem encrypts messages in the $[0, 2^{\kappa-1}]$ range, where κ is the bit size of p' and q', then we need $2^{\kappa-1}$ to be greater then $n := |\mathrm{DB}|$. In other words, we need $n' > n^3$.

To summarize, we have :

- Br-RSA parameters: $N = PQ$, with P and Q primes. Next v is chosen in $\mathbb{Z}^*_N$ with $gcd(v, \phi(N)) = 1$
- Okamoto-Uchiyama parameters: $n' = (p')^2q'$ with p' and q' primes

- Conditions: Let $n := |\mathrm{DB}|$, we should have $P > n'$, $Q > n'$, $v > n$, and $n' > n^3$.

The Authorizer now computes $h' \times ((\mathsf{c} \mod n')^{-1} \mod N) = ((g_1 g^{-1})^{\mathrm{ID}_{\mathcal{A}}} g_2^{x_2} \cdots g_\ell^{x_\ell} h^{-r}\alpha^v) \mod N$ and produces a signed proof of knowledge of an RSA representation of $h' \times ((\mathsf{c} \mod n')^{-1} \mod N)$ with respect to basis $((g_1 g^{-1}), g_2, \cdots, g_\ell, h^{-1}, v)$. Once the signed proof is accepted, the Receiver submits the SPIR query together with the signed proof to the Sender. The Sender in turn checks the validity of the credential, and the signed proof, and proceeds with the remaining steps of the original SPIR scheme of Figure 3.3.

3.9 Conclusion

We described a new access control scheme, where access policies are defined by the data subjects. More specifically, the proposed scheme allows database managers to be convinced that each of their stored records is being retrieved according to the policies of the data subjects, without the querier leaking information regarding the identity of the record that has been retrieved or the identity of the data owner. We present three constructions: the first is based on the discrete logarithm problem, while the other two are based on the RSA problem. The constructions we propose rely on anonymous authorizations, and they utilize SPIR systems and privacy-preserving digital credentials. The authorizations contain non-modifiable, unforgeable user-defined policies governing their use. Moreover, authorizations can be anonymously revoked by their issuers whenever the need arises. Compared to the complexity of the underlying SPIR scheme, the increase in complexity incurred by our solution is negligible.

This work can be extended in a number of ways. For example, it would be interesting to add a mechanism to support "authorized and anonymous editing of records".

One could also try to improve efficiency, and propose additional constructions based on other building blocks and assumptions.

Chapter 4

Efficient Multi-Authorizer ASPIR

In Chapter 3, we presented a technique allowing users to authorize access to their remotely stored data according to a self-defined privacy policy, without the database manager ascertaining the access pattern to their records. In this Chapter, we generalize the above technique to a setting where each record on the database is co-owned by a number of parties instead of a single one. The protocol we propose is such that the storage server answers a query only if convinced that the Receiver party holds a valid authorization from the owners of the target record. This is achieved without the storage server learning any information about the identity of the target record. We provide a first construction that allows a Receiver to retrieve a DB record only if he has authorizations from all owners of the target record, and a second construction where the Receiver needs authorizations only from a subset S of the owners of size greater than a given threshold. We also provide a construction where owners of the same record do not have equal ownership rights, and the record in question is retrieved using a set of authorizations consistent with a general access structure. The proposed constructions are efficient and use a pairing-based signature scheme. Their security depends on the Bilinear Diffie-Hellman assumption.

4.1 Introduction

The mishandling of personal user data, and the lack of credible assurances about the way this data will be protected, are thought to be some of the biggest obstacles facing electronic services today. The main goal of this work is to give users increased control over their data, in particular when the data is stored on a remote server.

Research geared towards enhancing users' control over their data, received a significant attention in the past (e.g., [GMM06b, AdM02, YXFD04, Bra00, CL02b, CL04]). In Chapter 3, we have proposed a partial solution that contributes to reinforcing user's control over their data. This solution, called accredited symmetrically private information retrieval [Lay07], or ASPIR for short, assumes a setting where sensitive information belonging to users (data-subjects) is stored on a remote database DB managed by a party called a *Sender*. The setting includes an additional party called a *Receiver* who retrieves records from the database. The construction in [Lay07], allows a Receiver to retrieve data owned by the user (data-subject), from a database DB managed by the Sender, such that the following three requirements are satisfied: (1) *Privacy for the data-subject:* the Receiver can retrieve a data record only if he has a valid authorization to do so from the record owner, (2) *Privacy for the Receiver:* the Sender is convinced that the Receiver's query is authorized by the owner of the target DB record, without learning any information about the content of the query, or the identity of the record owner, and (3) *Privacy for the Sender:* the Receiver cannot retrieve information about more than one record per query. For example, the Receiver cannot use an authorization from user U to learn information about database records not belonging to U.

The constructions in Chapter 3, cover a setting where each record in the database is owned by a single user. In many applications, data records are the property of several parties simultaneously rather than a single one. For example, in the healthcare domain, a medical procedure is performed by a *doctor* on a *patient* within the premises of a *hospital*. It may be natural in some jurisdictions that all three parties, namely the patient, the doctor, and the hospital, have a right to the database record documenting the medical procedure. As a result, a Receiver (e.g., a second doctor) who wants to have access to the above record, needs an authorization from all three record owners. With the obtained authorizations, the Receiver should be able to retrieve the target record subject to the following conditions: (1) the Receiver can retrieve the record in question only if he has the approval of all record owners, (2) the Sender is convinced that the Receiver's query is approved by the owners of the target data, without learning any information about the index of the target data, or the identity of the authorizers, and (3) the Receiver cannot retrieve information about records other than the one defined in the submitted query.

The ASPIR constructions of Chapter 3 rely on privacy-preserving digital credentials [Bra00] to protect the anonymity of the authorizer with respect to the Sender. The digital credential primitive has been used in addition to hide the index of the retrieved record, and to guarantee the unforgeability of the issued authorizations. While highly versatile, the digital credentials of [Bra00] do require a certain amount of computations from the different participants, especially the authorizers. Moreover, the construction in Chapter 3 assumes that each record owner possesses a digital credential of the type in [Bra00], and that he is willing to use it to issue authorizations.

In this work, we extend the ASPIR protocol of chapter 3 to a context where each database record can have multiple owners. The protocol we present in this chapter has a neater and more generic design, and uses SPIR primitives in a black-box fashion, unlike the construction in chapter 3 which works specifically for Lipmaa's SPIR scheme [Lip05]. The construction we present here is more efficient than the one in chapter 3, and uses a lightweight pairing-based signature scheme similar to that in [BLS01] instead of digital credentials. In this chapter, we also propose a t-out-of-n threshold multi-authorizer ASPIR variant, where records can be privately retrieved by a Receiver as long as he has authorizations from t out of the n owners of the target record.

Finally, we consider a setting where the owners' rights to a record are not necessarily equal. For example one could imagine a setting where an authorization from the patient is sufficient to access his medical record, while authorizations from *both* the doctor and hospital are necessary to access the same record. The latter could be useful in cases of emergency where the patient is unable to grant an authorization.

Summary of contribution and chapter organization

This chapter presents the following original contributions:

- A multi-authorizer accredited SPIR scheme where data records stored on a Sender's database can be retrieved by a Receiver (1) only if the latter has authorizations to do so from the target record owners, and (2) without the Sender ascertaining information about the index of the retrieved record or the identity of any of the record owners.
- The proposed scheme allows record owners to encode, in the issued authorizations, any privacy policy they want to enforce on their data, including the Receiver's identity, an expiry date, etc.
- We also propose a variant scheme for t-out-of-n threshold access, where a Receiver is able to retrieve a data record only if it has authorizations from at least t out of the n owners of the record.

- We finally consider a setting where owners of a record have unequal rights. In this setting, records are retrieved in accordance with a general access structure reflecting the non-uniformity of owners' rights.

The rest of the chapter is organized as follows. Section 4.2 starts with some definitions, and a description of the SPIR primitive that will be used as a building block. In section 4.3, we present our main multi-authorizer ASPIR construction. Sections 4.4 and 4.5 highlight the security and privacy features of the proposed scheme, as well as its performance. In Section 4.6, we briefly describe an extension to our t-out-of-n threshold access scheme. Section 4.7 addresses the more general case where owners have unequal ownership rights. Finally, in section 4.9 we provide our concluding remarks.

4.2 Preliminaries

The construction we present uses a pairing-based signature scheme similar to [BLS01], and relies on the hardness of the Bilinear Diffie-Hellman Problem (BDH). We first introduce bilinear maps, and the BDH problem; following this we describe the pairing-based signature and the SPIR building blocks.

Definition 4.2.1 (Admissible bilinear pairings)**.** *Let* $(\mathbb{G}_1, \times)$ *and* $(\mathbb{G}_2, \times)$ *be multiplicative groups of the same prime order* q*. Assume that the discrete logarithm problem in* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *is hard, an admissible bilinear pairing is a map* $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ *satisfying the following properties:*

- *Bilinearity: For all* $P, Q \in \mathbb{G}_1$*, and* $\alpha, \beta \in \mathbb{Z}_q^*$*,* $e(P^\alpha, Q^\beta) = e(P, Q)^{\alpha\beta}$*.*
- *Non-degeneracy: There exists* $P, Q \in \mathbb{G}_1$ *such that* $e(P, Q) \neq 1_{\mathbb{G}_2}$*.*
- *Computability: Given* $P, Q \in \mathbb{G}_1$*, there is an efficient algorithm to compute* $e(P, Q)$*.*

Definition 4.2.2 (Bilinear Diffie-Hellman Problem). *Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map, and let P be a generator of $\mathbb{G}_1$. For $a, b, c \in \mathbb{Z}_q^*$, given the tuple (P, P^a, P^b, P^c) output $e(P, P)^{abc}$.*

4.2.1 Pairing-based signature scheme

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map, and let P be a generator of $\mathbb{G}_1$. Assume the signer has a private key $sk := x \in \mathbb{Z}_q^*$, and a corresponding public key $pk := P^x$. To sign a message m, the signer computes $\sigma := H(m)^x$, where $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is a public collision-resistant one-way function. The verifier accepts σ' as a valid signature on m' with respect to pk, only if the equation $e(\sigma', P) = e(H(m'), pk)$ holds.

4.2.2 Symmetrically private information retrieval

A private information retrieval scheme or PIR for short, involves two players: a Sender and a Receiver. The Sender manages a database DB, and answers queries on DB submitted by the Receiver. The main goal of PIR schemes is to allow the Receiver to retrieve a DB record of his choice without the Sender learning the content of his query, and without resorting to the trivial and inefficient method where the Sender just returns the whole database back to the Receiver. The property of hiding the content of the Receiver's query from the Sender is called *Privacy for the Receiver*.

PIR schemes are mainly concerned with providing *Privacy for the Receiver*. There are settings however, where the Sender too is interested in controlling access to his database. For example, the Sender could be a multimedia provider with a business model based on charging a fee for every piece of content accessed in his database.

A solution to this type of settings can be obtained by using Symmetrically Private Information Retrieval schemes or SPIR for short.

A SPIR scheme allows a Receiver to efficiently retrieve records from the Sender's database such that the following two properties are assured:

- *Privacy for the Receiver*: the Sender does not learn any information about the index of the target record. That is, even after seeing the query of the Receiver, the distribution of the view of the Sender remains indistinguishable from the uniform random distribution over the interval $[1, |\mathrm{DB}|]$.
- *Privacy for the Sender*: the Receiver does not learn any information on the database content, other than the target record. In other words, seeing the Sender's response to a given query, does not help the Receiver guess the content of the database that is not targeted in the query. We say that the distribution of the view of the Receiver on $\overline{\mathrm{DB}}$, the non queried part of the database, remains indistinguishable from the uniform random distribution on the space of $\overline{\mathrm{DB}}$.

Depending on whether the indistinguishability is perfect, statistical, or computational, the above properties, namely Privacy for the Receiver, and Privacy for the Sender can be either perfect, statistical, or computational. For example, Lipmaa proposes in [Lip05] a SPIR scheme that is computationally private for the Receiver and perfectly private for the Sender.

A significant number of PIR and SPIR schemes can be found in the literature (e.g., [CKGS98, KO97, CMS99, Lip05, GR05]) with various performance levels, and a multitude of features such as :

- Single-DB (e.g., [KO97]) vs. multiple-DB Senders (e.g., [CKGS98].)

- Use of algebraic properties (e.g., homomorphic encryption [Lip05] and ϕ-assumption [CMS99]) vs. non-algebraic properties (e.g., existence of one-way trapdoor permutation [KO97].)

- Index-based (e.g., [Lip05, CMS99]) vs. keyword-based queries (e.g., [CGN98].)

More information on these and other differences can be found in [OS07, Gas04]. For the purpose of this chapter however, we do not discuss these features any further, and use SPIR schemes in a *black-box* fashion.

Notations. In the remainder of this chapter we assume that we have a SPIR scheme denoted SPIR. Let s be the secret index of the record the Receiver is interested in. The Receiver uses the public information, and possibly his private information to compute a SPIR query encoding s. We denote by $\mathsf{Q}_{\mathsf{SPIR}}$ the query the Receiver submits to the Sender. Let $\mathsf{R}_{\mathsf{SPIR}}$ be the Sender's answer to the Receiver's query. The Receiver then uses his private information and s, to recover DB$[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$.

4.3 Protocol description

The multi-authorizer accredited SPIR protocol we propose relies on the two building blocks described above. We start by describing a first construction in section 4.3.2, and then present a more efficient one in section 4.3.3. We assume the public parameters of the above building blocks are already known to all parties: the Sender, the Receiver, and the Authorizers.

4.3.1 Settings

We assume that multiple parties play the Authorizer role, as opposed to one single party as in [Lay07]. Without loss of generality, we assume that we have *three* types of Authorizers $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$. For example, $\mathcal{A}$ could represent the Patients, $\mathcal{B}$ the Doctors, and $\mathcal{C}$ the Hospitals. In addition, our setting contains a database DB of size N managed by the Sender. Each record in DB belongs to a triplet of parties (A, B, C) from the set $\mathcal{A} \times \mathcal{B} \times \mathcal{C}$. The owners (A, B, C) of a given record may or may not have the same rights (depending on the privacy laws in place.) Section 4.7 treats the case where owners have unequal rights.

Next, we assume that each party has an identifier *ID*, and that each record in the database is labeled with the identity of its owners, e.g., $(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$. We also assume the existence of a publicly known one-to-one mapping between *ID* triplets and the indices of DB's records. The latter is denoted: *index* : $\mathcal{A} \times \mathcal{B} \times \mathcal{C} \rightarrow [1, N]$. Finally, we assume that each DB record indexed by j, and corresponding to identity triplet $(\mathit{ID}_{j,1}, \mathit{ID}_{j,2}, \mathit{ID}_{j,3})$, contains a field with the owners' public keys $(pk_{j,1}, pk_{j,2}, pk_{j,3}) := (P^{x_{j,1}}, P^{x_{j,2}}, P^{x_{j,3}})$ stored in it.

4.3.2 First construction

Let (A, B, C) be a tuple of owners who are willing to authorize a Receiver *RecID*, to retrieve their record indexed by $s := \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$, according to a usage policy $\mathcal{P}$. Each of the owners first provides the Receiver with a signature $\sigma_i(P_m) := (P_m)^{x_i}$, for $P_m := H(s, \mathit{RecID}, \mathcal{P})$. Next, the Receiver prepares a SPIR query $\mathsf{Q}_{\mathsf{SPIR}}$ for index s, and submits *RecID*, $\mathcal{P}$, and $\mathsf{Q}_{\mathsf{SPIR}}$ to the Sender. Upon receiving this information, the Sender first authenticates[1] *RecID* and verifies that the submitted query is compliant with usage policy $\mathcal{P}$.[2] If one of these checks fails the Sender aborts, else it proceeds with the query. Next, for every Authorizer type[3], the Sender chooses a random blinding factor $\delta_i \in \mathbb{Z}_q^*$, where $i \in [1,3]$ for the purpose of our description. For each record DB$[j]$, the Sender computes $P_{mj} := H\left(j, \mathit{RecID}, \mathcal{P}\right)$ and DB$'[j] :=$ DB$[j] \times \left(\prod_{i=1}^{3} e((P_{mj})^{\delta_i}, pk_{j,i})\right)$.

The Sender then executes the SPIR scheme on $\mathsf{Q}_{\mathsf{SPIR}}$ and DB$'$, and returns the response $\mathsf{R}_{\mathsf{SPIR}}$ to the Receiver along with P^{δ_1}, P^{δ_2}, and P^{δ_3}. The Receiver first recovers DB$'[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$, and then computes

$$\begin{aligned}
\mathrm{DB}_0[s] &= \mathrm{DB}'[s] \,/\, \prod_{i=1}^{3} e\left(\sigma_i(P_m), P^{\delta_i}\right) \\
&= \mathrm{DB}[s] \times \prod_{i=1}^{3} e\left((P_{m,s})^{\delta_i}, pk_{s,i}\right) / \prod_{i=1}^{3} e\left((P_m)^{x_i}, P^{\delta_i}\right) \\
&= \mathrm{DB}[s] \times \left(\prod_{i=1}^{3} e\left((P_{m,s})^{\delta_i}, P^{x_{s,i}}\right) / \, e\left((P_m)^{x_i}, P^{\delta_i}\right) \right) \\
&\overset{(*)}{=} \mathrm{DB}[s] \times \left(\prod_{i=1}^{3} e\left((P_m)^{\delta_i}, P^{x_i}\right) / \, e\left((P_m)^{x_i}, P^{\delta_i}\right) \right) \\
&= \mathrm{DB}[s]
\end{aligned}$$

$(*)$: the equality holds because for $s = \mathit{index}(ID_A, ID_B, ID_C)$, the keys $x_{s,i}$ are no other than the secret keys x_i of owners (A, B, C). Similarly $P_{m,s} = P_m$.

[1] The receiver can be authenticated using conventional X.509 public key certificates for example. In case the identity of the receiver needs to be protected, then privacy-preserving credential systems (e.g., [Bra00, CL02b, CL04]) can be used instead.

[2] The policy $\mathcal{P}$ can be any Boolean statement of the form: "Receiver should be a practicing surgeon accredited by the College of Physicians **AND** Retrieval date prior to 31 July 2009" for instance. The policy can be encoded using state of the art XML format for example.

[3] As noted earlier, to keep the description simple we assumed *three* types $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$.

In the above solution, the Sender is required to compute a number of pairings linear in the number of authorizer types (to compute each $e\left((P_{mj})^{\delta_i}, pk_{j,i}\right)$, $i \in [1, n]$), and to return P^{δ_i} for each authorizer type. This clearly results in computational and communication complexities that are linear in the number of authorizer types. We improve these complexities in the next section.

4.3.3 Improved construction

Let (A, B, C) be a tuple of owners who are willing to authorize a Receiver *RecID*, to retrieve their record indexed by $s :=$ *index*(ID_A, ID_B, ID_C), according to a usage policy $\mathcal{P}$. Each of the owners first provides the Receiver with a signature $\sigma_i(P_m) := (P_m)^{x_i}$, for $P_m := H\left(s, RecID, \mathcal{P}\right)$. The Receiver aggregates the σ_i's into one single signature $Sig(P_m) := \prod_{u\in\{A,B,C\}}\sigma_u(P_m)$. He then prepares a SPIR query $\mathsf{Q}_{\mathsf{SPIR}}$ for index s, and submits *RecID*, $\mathcal{P}$, and $\mathsf{Q}_{\mathsf{SPIR}}$ to the Sender as in the first construction. The Sender processes the Receiver's query as in the first construction, except that here it chooses a single random blinding factor $\delta \in \mathbb{Z}_q^*$, and for each $1 \leq j \leq N$, computes DB$'[j] :=$ DB$[j] \times e\left(P_{mj}, \prod_{u=1}^{3} pk_{j,u}\right)^{\delta}$. The use of a single blinding factor δ, for all types of Authorizers, reduces the Sender's computational overhead for each database record, from linear in the number of Authorizer types to constant. A similar reduction is achieved in the size of the Sender's response which drops from linear in the number of Authorizer types to constant.

Finally, the Sender executes the SPIR scheme on $\mathsf{Q}_{\mathsf{SPIR}}$ and DB$'$, and returns the response $\mathsf{R}_{\mathsf{SPIR}}$ to the Receiver along with δP. The Receiver then recovers DB$'[s]$ from $\mathsf{R}_{\mathsf{SPIR}}$, and computes DB$_0[s]$ = DB$'[s]\,/e(Sig(P_m), P^{\delta})$, thereby using the aggregate signature $Sig(P_m)$ as if it was a "decryption key". This approach of using signatures as decryption keys is of general interest, and could be useful in the wider context of access control. A summary of the whole protocol is given in Figure 4.1.

Receiver (*RecID*) | **Public Info** | **Sender (Database DB)**

$(P_m, \sigma_u(P_m)),\ u \in \{A, B, C\},$ for $P_m := H(s, \mathit{RecID}, \mathcal{P}),$
where $s := \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C),$ and $\mathcal{P} :=$ {usage policy}

$$\{\boldsymbol{pk_u}\}_{u\in\{A,B,C\}} \triangleq \{\boldsymbol{pk_{s,i}}\}_{1\le i\le 3},$$
$$\boldsymbol{e(\cdot,\cdot), P}, \mathbb{G}_1 = \langle \boldsymbol{P} \rangle, \mathbb{G}_2, \boldsymbol{q},\ \textsf{\textbf{SPIR scheme}}$$

$$\mathit{Sig}(P_m) = \prod_{u\in\{A,B,C\}} \sigma_u(P_m) = \prod_{u\in\{A,B,C\}} (P_m)^{x_u} = (P_m)^{\sum_u x_u}$$

$\mathsf{Q} = \mathsf{Q}_{\mathsf{SPIR}}(s)$ $\xrightarrow{\mathsf{Q},\ \boldsymbol{RecID},\ \boldsymbol{\mathcal{P}}}$

If $\mathcal{P}$ satisfied continue
else abort
Choose $\delta \in_R \mathbb{Z}_q^*$
For j := 1 to N do :
$P_{mj} = H(j, \mathit{RecID}, \mathcal{P})$
$\mathrm{DB}'[j] = \mathrm{DB}[j] \times e\left(P_{mj}, \prod_{u=1}^{3} pk_{j,u}\right)^{\delta}$

Execute SPIR on DB′ and Q

SPIR-recover DB′$[s]$ from Res $\xleftarrow{\mathsf{Res},\ \boldsymbol{P^{\delta}}}$ **Let Res = $\mathsf{R}_{\mathsf{SPIR}}(\mathsf{Q}, \mathrm{DB}')$**

Output DB$_0[s]$:= DB′$[s]$ / $\boldsymbol{e(Sig(P_m), P^{\delta})}$

Figure 4.1: Multi-Authorizer ASPIR (Improved Construction)

It can be easily checked that $\mathrm{DB}_0[s]$ computed by the Receiver is the desired record $\mathrm{DB}[s]$.

$$\begin{aligned}\mathrm{DB}_0[s] &= \mathrm{DB}'[s] \,/\, e(Sig(P_m), P^{\delta}) \\ &= \mathrm{DB}[s] \times e(P_m, \prod_{u=1}^{3} pk_u)^{\delta} \,/\, e((P_m)^{\sum_{u=1}^{3} x_u}, P^{\delta}) \\ &= \mathrm{DB}[s] \times e(P_m, P^{\sum_{u=1}^{3} x_u})^{\delta} \,/\, e((P_m)^{\sum_{u=1}^{3} x_u}, P)^{\delta} \\ &= \mathrm{DB}[s]\end{aligned}$$

Remark. The usage policy $\mathcal{P}$ encoded in P_m can be any privacy policy the owners want to enforce on their record. This may include usage limitations such as an expiry date, a description of what is considered an acceptable usage scenario etc. Note that by binding authorizations to a specific Receiver exclusively, the protocol is able to prevent pooling attacks[4].

4.4 Security and privacy evaluation

Definition 4.4.1 (Valid Authorization)**.** *Let (A, B, C) be the owners of a record in the Sender's DB, indexed by $s = index(ID_A, ID_B, ID_C)$. For a given usage policy $\mathcal{P}$, a Receiver is said to have a valid authorization under $\mathcal{P}$, from owner $O \in \{A, B, C\}$, if and only if the Receiver has a valid signature from O on $P_m = H\,(s, ReceiverID, \mathcal{P})$, and $\mathcal{P}$ is satisfied at the time the authorization is used.*

Definition 4.4.2 (Secure Multi-Authorizer ASPIR protocols)**.** *A multi-Authorizer ASPIR protocol (MASPIR) is said to be secure if the following hold: (1) the protocol satisfies the "privacy for Receiver" and "privacy for Sender" properties usually provided by conventional*

[4] Pooling attacks occur when different receivers combine their authorizations in order to gain access to records they were not able to get access to individually.

SPIR schemes, and (2) a Receiver cannot retrieve a given record with non-negligible probability unless he has authorizations from all owners of the given record. For the special case of threshold MASPIR (respectively, MASPIR with unequal ownership rights), we require the Receiver to have authorizations from a subset S of the owners, where the size of S is greater than a given threshold (respectively., a subset S of the owners that is part of a predefined General Monotonic Access Structure.)

Theorem 4.1. *Assuming the Bilinear Diffie-Hellman problem is hard and the* SPIR *primitive is secure, the protocol presented in Figure 4.1 is a secure MASPIR protocol.*

Note. At the time of writing this chapter, we were not able to prove Theorem 4.1. We do prove however the following weaker theorem.

Theorem 4.2. *Assuming the Bilinear Diffie-Hellman problem is hard and the* SPIR *primitive is secure, the protocol presented in Figure 4.1 is such that it is hard for an adversary to retrieve a given record, while missing as little as* one single authorization *from one of the owners of that record.*

Proof. The protocol illustrated in Figure 4.1 is, by construction, based on a secure SPIR primitive. By examining the exchange of messages, it is easy to see that the protocol of Figure 4.1 satisfies the "privacy for Receiver" and "privacy for Sender" properties already provided by the underlying SPIR primitive. In the following we examine the second security criterion of definition 4.4.2.

We show that if an Adversary $\mathcal{A}_{MASPIR}$ can retrieve a record, while missing *one single authorization* from one of the owners of that record, then the Bilinear Diffie-Hellman problem can be solved. In other words, we show how to construct an Adversary $\mathcal{A}_{BDH}$ that uses $\mathcal{A}_{MASPIR}$ to solve the Bilinear Diffie-Hellman problem.

Let s be the index of the record targeted by the Adversary $\mathcal{A}_{MASPIR}$ playing the role of a malicious Receiver. Let $(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$ be the identity tuple of the corresponding owners, i.e., $s = \mathit{index}(\mathit{ID}_A, \mathit{ID}_B, \mathit{ID}_C)$. The Adversary $\mathcal{A}_{MASPIR}$ submits a query and retrieves record DB$[s]$ from the Sender's response, while having valid signatures from only two (out of the three) owners (A, B, C).[5] Without loss of generality, assume he has signatures from A and B.

For any given instance $(P', (P')^a, (P')^b, (P')^c)$ of the BDH problem, the Adversary $\mathcal{A}_{BDH}$ obtains $(abc) \cdot e(P', P')$ by interacting with $\mathcal{A}_{MASPIR}$ and playing the role of the owners A and B, and the Sender as follows.

1. $\mathcal{A}_{BDH}$ chooses random elements x_A, x_B of $\mathbb{Z}_q^*$ and sets $P = P'$, $pk_A = (P')^{x_A}$, $pk_B = (P')^{x_B}$, and $pk_C = (P')^c$.

2. $\mathcal{A}_{BDH}$ gives P and $\{pk_i\}_{i \in \{A,B,C\}}$ to $\mathcal{A}_{MASPIR}$.

3. $\mathcal{A}_{BDH}$ sets $P_m = (P')^b$ for the parameters s, *RecID*, and $\mathcal{P}$ (the hash function H is assumed as a random oracle in this proof).

4. $\mathcal{A}_{BDH}$ computes signatures $\sigma_A(P_m) = (P_m)^{x_A}$ and $\sigma_B(P_m) = (P_m)^{x_B}$, and gives them to $\mathcal{A}_{MASPIR}$, along with s, *RecID*, and usage policy $\mathcal{P}$.

5. $\mathcal{A}_{MASPIR}$ submits $\mathsf{Q} := \mathsf{Q}_{\mathsf{SPIR}}(s)$, *RecID*, and usage policy $\mathcal{P}$ to $\mathcal{A}_{BDH}$.

6. $\mathcal{A}_{BDH}$ sets :

   - DB$_0[j] := e\left(P_m, ((P')^a)^{(x_A + x_B)}\right)$ for all j.
   - $P^\delta := (P')^a$

   $\mathcal{A}_{BDH}$ then executes SPIR on DB$_0$ and Q and returns $\mathsf{Res} = \mathsf{R}_{\mathsf{SPIR}} = \mathsf{SPIR}(\text{DB}_0, \mathsf{Q})$ and P^δ to $\mathcal{A}_{MASPIR}$.

[5] In addition to these signatures, $\mathcal{A}_{MASPIR}$ *might* have a *partial* knowledge about the signature of the third owner. However, $\mathcal{A}_{MASPIR}$ may not know the entire signature of the third owner, because he would otherwise become an *authorized* Receiver.

7. $\mathcal{A}_{MASPIR}$ computes (this step could be done earlier)

$$Sig(P_m) := \prod_{i \in \{A,B,C\}} \sigma_i(P_m) := (P')^{b(x_A+x_B+c)}$$

8. $\mathcal{A}_{MASPIR}$ recovers $\mathrm{DB}_0 = \mathrm{DB}_0[s]$ from Res and computes

$$\begin{aligned}
\mathrm{DB} &= \mathrm{DB}_0 \,/\, e(Sig(P_m), P^{\delta}) \\
&= e\left(P_m, (P')^{a(x_A+x_B)}\right) \,/\, e((P')^{b(x_A+x_B+c)}, (P')^{a}) \\
&= e\left((P')^{b}, (P')^{a(x_A+x_B)}\right) \,/\, e((P')^{b(x_A+x_B+c)}, (P')^{a}) \\
&= e\left(P', P'\right)^{ab(x_A+x_B)} \,/\, e(P', P')^{ab(x_A+x_B+c)} \\
&= e\left(P', P'\right)^{(ab(x_A+x_B)-ab(x_A+x_B+c))} \\
&= e\left(P', P'\right)^{-(abc)}
\end{aligned}$$

9. $\mathcal{A}_{BDH}$ outputs $\mathrm{DB}^{-1} = e\left(P', P'\right)^{abc}$

$\mathcal{A}_{BDH}$ can solve the BDH problem using $\mathcal{A}_{MASPIR}$. Therefore, assuming the BDH problem is hard, retrieving a record while missing one of the the required authorizations, is infeasible. ∎

The above proof can be easily generalized to the case where records belong to n owners, for any integer n. Similar theorems can be proved for the protocol variants of Sections 4.6, and 4.7.

4.5 Performance analysis

In this analysis we focus mainly on exponentiation operations; group operations such as multiplications are significantly cheaper. As noted by Boyen in [Boy06], a pairing operation can be reduced to a single exponentiation of size less than the group order. Following Boyen's observation, we count a pairing operation here as a regular exponentiation operation, similar to those used in Section 4.3.2 to compute signatures for example.

It is worth noting that all SPIR schemes require $\Omega(|\mathrm{DB}|)$ computations by the Sender; if this is not the case, then the Sender will not access at least one record in the database, and thus we can safely infer that the non accessed records are not being sought in the Receiver's query, thereby violating the Receiver's privacy. Based on this observation, the total computations of the Sender cannot be expected to drop below this linear lower bound.

Let n be the number of owners of each record in the Sender's database, and let N be the database size. In addition to the basic operations required by the underlying SPIR scheme, our protocol requires the following: (a) each owner of the target record must perform one exponentiation in $\mathbb{G}_1$, (b) the Receiver to perform a n-element multiplication in $\mathbb{G}_1$ (to compute $Sig(P_m)$ from the elementary signatures $Sig_u(P_m)$), and one pairing, and (c) the Sender to perform N exponentiations and N pairings. Note that the exponentiation and multiplication operations in items (a) and (b) are pre-computable, and can be performed in advance offline. Despite the increase in functionalities, the protocol we propose does not lead to higher computational cost compared to that of the underlying SPIR scheme, which is already linear in N. Similarly, our communication performance is equivalent to that of the underlying SPIR scheme, since we increase the amount of exchanged data only by a small constant. This increase is negligible, considering that the best known communication complexity achieved for SPIR to date, is $\mathcal{O}\big(\log^2(N)\big)$ [GR05, Lip05].

4.6 Extension to threshold access

In some applications it may be useful to provide a mechanism to allow a Receiver to privately recover a certain record as long as he has authorizations from t out of the n record owners. As in the basic case, the Sender should not learn the identity of the Authorizers or the index of the retrieved record. We do this using ideas similar to those in [Bol03]. In the following description, we highlight the changes we made to the protocol presented in section 4.3.

Assume the record owners jointly select a master secret key *MSK* $:= x \in \mathbb{Z}_q^*$, and distribute it among themselves using a verifiable (t, n)-secret sharing scheme. We note that there is no need for a third party in the secret sharing procedure. The n record owners can generate a secret key *MSK*, and privately distribute the shares among themselves without help from a trusted third party, using protocols such as [Ped91a, IS90]. The secret generation is such that no shareholder knows *MSK* individually. Let $x_u,\ u \in [1, n]$ be the n secret shares, and $(sk_u, pk_u) := (x_u, P^{x_u})$, $u \in [1, n]$, be the private/public key pairs of the record owners. The master secret key x can be written as a Lagrange interpolation of any subset of shares x_u, of size greater or equal to t. Let *MPK* $:= P^x$ be the corresponding master public key. For DB$[j]$, the j^{th} record in the database DB, let $(ID_{j,1}, \cdots, ID_{j,n})$ be the identity tuple of its owners, and let *MPK*$_j$ be their joint master public key. We assume that DB$[j]$ has a field with *MPK*$_j$ stored in it. Note that given the owners' public keys $(pk_{j,1}, \cdots, pk_{j,n})$, anyone can reconstruct the corresponding master public key *MPK*$_j$ by simple Lagrange interpolation.

A Receiver holding authorizations $(P_m, \sigma_u(P_m))$ from at least t record owners $\{u_1, \cdots, u_t\}$, can reconstruct a signature on P_m which can be verified with the master public key *MPK*, by computing $\mathit{Sig}(P_m) = \prod_{v=1}^{t} \sigma_{u_v}(P_m)^{L_{u_v}} = (P_m)^{\sum_{v=1}^{t} L_{u_v} x_{u_v}} = (P_m)^x$, where L_{u_v} denote the appropriate Lagrange coefficients[6]. The Receiver then proceeds with the protocol as in the basic case, and submits $\mathsf{Q}_{\mathsf{SPIR}}$, RecID, and usage policy $\mathcal{P}$ to the Sender.

The Sender checks the consistency of the submitted query with the Receiver's identity and usage setting, and chooses a random blinding factor $\delta \in \mathbb{Z}_q^*$. For each record in the database indexed by j, the Sender computes P_{mj}, and $\mathrm{DB}_0[j] = \mathrm{DB}[j] \times e\left(P_{mj}, \mathit{MPK}_j\right)^{\delta}$. The rest of the protocol is similar to the one outlined in Section 4.3.

4.7 Extension to authorizers with unequal rights

Up to this point, we have assumed that the owners of a given record all have equal rights. In other words, if a record belongs to (A, B, C) then an authorization from A is worth exactly the same as that from B or C. In some settings however, owners of a record do not have equal rights. For instance in the healthcare context, a medical record belonging to (patient A, doctor B, hospital C) should be accessible only if authorizations are provided, say from A alone, or B and C together. Authorizations from B or C alone are not sufficient. More generally, for a record R belonging to a set of owners $O = \{A_1, \cdots, A_n\}$, we denote by $\mathcal{A} \subset 2^O$, the subsets of O whose authorizations are sufficient to access R. The set $\mathcal{A}$ is called a *generalized access structure*. One of the main properties of a generalized access structure is *monotonicity*.

An access structure $\mathcal{A} \subset 2^O$ is said to be *monotone*, if for any subsets $B \subset O$ and $B' \subset O$, if $B \in \mathcal{A}$ and $B \subset B' \subset O$, then $B' \in \mathcal{A}$.

[6] The values of the L_{u_v}'s depend only on the values of the u_v's.

In the following we show how secret sharing with a generalized access structure [ISN89] can be used to realize multi-authorizer ASPIR in a context where owners have unequal rights to their record.

Consider a database record R, and assume R's owners agree on a generalized access structure $\mathcal{A}$. Using a method similar to that outlined in Section 4.6, R's owners jointly select a master secret key *MSK* $:= x \in \mathbb{Z}_q^*$, and split it into shares among themselves, according to the access structure $\mathcal{A}$. The secret generation and distribution are such that no shareholder knows *MSK* individually, and no help from a secret sharing dealer is needed. More details on how this is done are given in the example below. After the distribution of the shares, each owner ends up with a share of information on *MSK*, that he uses as a signing key. A Receiver obtains signatures from a subset of owners as in the threshold case. Next, the Receiver combines the partial signatures using Lagrange interpolation in order to recover a valid signature with respect to master key *MPK*. Note that the master public key *MPK* corresponding to *MSK* is stored in a field within record R, as in the threshold construction presented in Section 4.6. The signature will be valid only if the Receiver holds partial signatures from a set of owners that is part of the access structure $\mathcal{A}$.

Example. Let R be a record belonging to (A_1, A_2, A_3, A_4), who agree on an access structure $\mathcal{A} = \{\{A_1, A_2, A_3\}, \{A_1, A_4\}, \{A_2, A_4\}, \{A_3, A_4\}\}$. Let $x \in \mathbb{Z}_q^*$ be the master secret key (*MSK*) that (A_1, A_2, A_3, A_4) select jointly. Let (x_1, x_2, x_3, x_4) be shares of x in a $(4, 4)$-threshold secret sharing scheme. Assume we have a mechanism to securely distribute share tuples (x_2, x_4) to A_1, (x_3, x_4) to A_2, (x_1, x_4) to A_3, and (x_1, x_2, x_3) to A_4. It can be easily seen that the distributed share tuples satisfy the access structure $\mathcal{A}$, since combining them according any of the tuples in $\mathcal{A}$ is sufficient to obtain all four shares of x. Further details on how share tuples are determined in the general case, can be found in [ISN89].

The received x_i's are used by the owners as private signing keys to issue authorizations. For example, a Receiver authorized by $\{A_1, A_4\} \in \mathcal{A}$, obtains $(P_m, \sigma_2(P_m), \sigma_4(P_m))$ from A_1, and $(P_m, \sigma_1(P_m), \sigma_2(P_m), \sigma_3(P_m))$ from A_4, where $\sigma_i(P_m) = (P_m)^{x_i}$ for $1 \leq i \leq 4$. The Receiver then computes the signature on P_m with respect to master key *MPK*, by interpolating the σ_i's as follows : $\sigma(P_m) = \prod_{v=1}^{4}(\sigma_v(P_m))^{L_v}$, where L_v denote the appropriate Lagrange coefficients. The reconstructed signature is later used by the Receiver to "decrypt" DB$'[s]$ as in the original MASPIR protocol outlined in Section 4.3. The rest of the protocol remains the same as in the threshold case.

We now give a brief overview on how the master secret x is jointly selected by (A_1, A_2, A_3, A_4), and how the shares are generated and distributed. For $1 \leq i \leq 4$, owner A_i chooses $s_i \in_R \mathbb{Z}_q^*$, and generates a random 3rd-degree polynomial in $\mathbb{Z}_q$, $f_i(X) = s_i + \sum_{j=1}^{3} a_{ij} X^j$. Let $f(X) = \sum_{i=1}^{4} f_i(X)$. If we set $x = \sum_{i=1}^{4} s_i$, then $\{x_j = f(j),\ 1 \leq j \leq 4\}$ is a valid set of (4,4)-threshold shares of x. Note that x is uniquely determined at this point, and yet it is unknown to any of the individual A_i's. Next, the share tuples are distributed as follows. Consider for instance the share tuple (x_2, x_4) intended for A_1. For $2 \leq i \leq 4$, owner A_i sends $(f_i(2), f_i(4))$ to A_1. Next, A_1 obtains the desired shares by computing $x_j = \sum_{i=1}^{4} f_i(j)$, for $j \in \{2, 4\}$.[7] The remaining share tuples for A_2, A_3, and A_4 are distributed in the same fashion. The share distribution above can be made verifiable using the technique of [Ped91a].

[7] Note that the tuples $(f_i(2), f_i(4))$; $2 \leq i \leq 4$, received by A_1, do not reveal any information to A_1 about the secrets s_i, since $f_i(X) = s_i + \sum_{j=1}^{3} a_{ij} X^j$ are random 3rd-degree polynomials, and in order to learn any information about s_i, one needs to know the values of $f_i(X)$ in at least *four* points. Recall here that each A_i receives the values of $f_{j \neq i}(X)$ in at most three points, and that the A_i's are assumed not to collude and pool their shares. The same observation holds for the tuples received by A_2, A_3, and A_4.

4.8 The case of an owner tuple possessing multiple records.

So far, we have assumed that each tuple (A, B, C) could own at most one single record. In the following we briefly discuss the case where a tuple of owners may possess $k \geq 1$ records. The goal now is to allow these owners to issue an authorization to the Receiver so that he can retrieve their k records. One trivial way to do this is as follows. First, add one argument to the $index(\ldots)$ function, specifying the rank of a record. For example, the term $s_i = index(A, B, C, i)$ will now denote the index of the i^{th} record (among k) belonging to (A, B, C). The owners now give the Receiver an authorization for each DB$[s_i]$, and the retrieval proceeds as in the basic case.

To avoid the issuing of multiple authorizations, we can use the following method. The value of P_m, in the authorization issued to the Receiver, is now computed as $P_m = H(ID_A, ID_B, ID_C, RecID, \mathcal{P})$, and each of the owners provides the Receiver with a signature $\sigma_u(P_m) := (P_m)^{x_u}$, $u \in \{A, B, C\}$. The Receiver then aggregates the σ_u's into one single signature $Sig(P_m) := \prod_{u \in \{A,B,C\}} \sigma_u(P_m)$ as in section 4.3.3. A similar modification is required on the Sender's side as well. For $j \in [1, N]$, the Sender computes the P_{mj}'s as $P_{mj} = H(ID_{j,1}, ID_{j,2}, ID_{j,3}, RecID, \mathcal{P})$. Note that the identities $ID_{j,u}$, $u \in [1, 3]$, of the record owners are readily available to the Sender along with the corresponding public keys $pk_{j,u}$, $u \in [1, 3]$. The Sender then computes DB$'[j]$ from DB$[j]$ as in section 4.3.3 using the new value of P_{mj} instead. As a result of the above modifications, we note that for all indices $s_i = index(ID_{j,1}, ID_{j,2}, ID_{j,3}, i)$, $i \in [1, k]$, referencing the records belonging owner tuple $(ID_{j,1}, ID_{j,2}, ID_{j,3})$, the value of P_{mj} is the same, and the entries DB$[s_i]$ are all encrypted with the same "key": $e(P_{mj}, \prod_{u=1}^{3} pk_{j,u})^{\delta}$. The Receiver finally retrieves the entries DB$'[s_i]$ one by one, using a SPIR primitive, and decrypts them using his aggregate signature $Sig(P_m)$ as in the basic case.

In the above scheme, the Receiver retrieves the entries DB$'[s_i]$ separately. This can be improved using a method based on the hybrid encryption paradigm [Sho01]. First, we modify the setting to include two databases DB$_1$ and DB$_2$. Each entry in DB$_1$ is used to store a key corresponding to a triplet of owners. The database DB$_2$ on the other hand, is used to store the actual owners' records encrypted under the keys kept in DB$_1$. The encryption can be done using some data encapsulation mechanism (DEM)[8][Sho01]. DB$_2$ is such that the records belonging to a given tuple of owners are all encrypted under the *same* key. In order to grant access to their records, the owners (A, B, C) give the Receiver an authorization to retrieve their encryption key from DB$_1$ (using the construction outlined in section 4.3.3.) And using this key, the Receiver decrypts all the DB$_2$ records belonging to (A, B, C). Note that if DB$_2$ can be made public, the Receiver does not need to run the SPIR scheme again to retrieve the encrypted records.

4.9 Conclusion

We have presented a special access control protocol for databases containing sensitive personal data. In particular, the described constructions allow a Receiver to retrieve a record in the database, if and only if: (a) he has authorizations from all, or in the case of a threshold scheme a subset of, the owners of the target record, and (b) the context in which the database is queried, is consistent with a usage policy chosen by the owners of the target record, and embedded in authorizations issued to the Receiver. The solution we propose is such that the database manager is convinced that the above

[8] DEM could be any symmetric-key encryption scheme (e.g., AES.)

holds without being able to ascertain any information about the index of the target record or the identity of its owners. The security of the proposed construction relies on the Bilinear Diffie-Hellman assumption. We have also presented a construction for a setting where the owners of a record do not have equal ownership rights. Furthermore, we have provided a solution for the case where a tuple of owners can authorize access to multiple records at once. The protocol we propose in this chapter is more efficient than the one in chapter 3 and can be constructed with any SPIR primitive. Despite the increase in functionality, the presented protocol does not add significant cost to the complexity of the underlying SPIR.

4.9.1 Possible extensions

One way to extend the work described in this chapter is to consider a setting where each entry in the Sender's database is labelled by a number of keywords, and where the database is queried by keywords instead of indices. Furthermore, we assume that each database entry is entirely encrypted now (i.e., both the DB record and the attached keywords are encrypted). Given the above settings, our goal is to achieve the following requirements:

- The Data-subjects should be able to control *who* can search their records, *what* keywords can be queried, as well as the *terms and conditions* of the search.
- The Receiver can *only* retrieve records matching the *authorized search keywords*.
- The DB Manager (or Sender) should not be able to learn the *identity of the data-subjects* who issued the search authorizations, the *search keywords*, or the *search results*.

Chapter 5

Privacy in Practice: A Protocol for e-Health

In previous Chapters, we have described a number techniques to help users protect their privacy and exert more control over their data. The presented techniques can be used in settings where the information is under the user's control, as well as in settings where the information is stored on a remote server lying outside the user's control. In this Chapter we apply some of these techniques to a real world application: healthcare. Real world healthcare systems are generally large and overly complex systems. Designing privacy-friendly protocols for such systems is a challenging task. In this Chapter, we present a privacy-preserving protocol for the Belgian healthcare system. The proposed protocol protects the privacy of patients throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practise. The presented protocol relies on standard privacy-preserving credential systems, and verifiable public key encryption, which make it readily suited for implementation.

5.1 Introduction

Healthcare represents one of the main pillars reflecting the quality of public service in our society. Over the years, countries around the world have experimented with a multitude of technical choices and policies to improve the quality of their health service. One technical choice that seems to be turning into a trend is the migration from traditional paper-based healthcare to electronic healthcare. The latter has a number of advantages. Among them we note the greater convenience and speed to access health data, which translates into shorter treatment delays, less medical errors, better statistics, higher cost-efficiency, better fraud detection mechanisms, and shorter refund delays for patients covered by health insurance plans.

Despite all the above benefits, patients are still showing a certain reluctance and skepticism towards new electronic healthcare systems. The reason for this skepticism is mainly attributed to the lack of assurances about the way patient data is handled, and the implications that may result from a breach of the patients' privacy.

To help reduce this lack of trust one should design e-health protocols with both security and privacy in mind. Due to the sensitive nature of health data, such protocols should be based on well established cryptographic primitives, and should provide defences against possible user inadvertencies such as ID card losses.

In order to facilitate the adoption of any new protocol in practice, one should take into account the current procedures, practices, and existing infrastructures. Ignoring these elements may result in a very high adaptation cost (e.g., to change the existing infrastructure before the new system can be used). In some cases the proposed protocols require the elimination of entire parties. Sometimes these parties represent government agencies or ministries, and removing them is simply unrealistic.

In this work, we design a protocol that protects the privacy of patients throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practice [1]. The Belgian healthcare system is a large and complex system with many players who do not necessarily share the same interests. The e-health protocol we propose protects the following: (1) the privacy of patients by eliminating any information leak that may harm their interests, and (2) the privacy of doctors, their prescription habits, and their interactions with patients. Moreover, our protocol has mechanisms to handle disputes and retrace fraudulent activities, all without changing the structure of the current Belgian healthcare practice. The protocol can also be easily adapted to satisfy other healthcare systems that are similar to the Belgian system.

Chapter organization

The rest of the chapter is organized as follows. Section 5.2 surveys related works. Section 5.3 introduces the Belgian healthcare system. Section 5.4 states the security and privacy properties we require for our protocol. In sections 5.5 and 5.6, we present the building blocks, as well as the protocol we propose to satisfy the previous requirements. Section 5.7 contains an evaluation of the proposed protocol. In section 5.9, we summarize our contributions, and suggest some ideas to extend the protocol.

5.2 Related work

A significant amount of work related to e-health can be found in the literature. One of the focal points has been the quest for efficient ways to migrate services from the

[1] There are auxiliary procedures in the Belgian healthcare system that are not covered here. The proposed protocol can be slightly modified to include them.

paper-based to electronic setting. A great deal of work for instance has been dedicated to features such as semantic web and interoperability between various healthcare organizations [HL7a, HL7b, IHE, VGP05]. Other issues such as reliability, accessibility, availability, storage integrity, and fault-tolerance have also been addressed [KSN$^+$07, TPB$^+$06].

Privacy in healthcare is also receiving increased attention. For instance, Ateniese and de Medeiros proposed in [AdM02] an e-health protocol compatible with the healthcare system in the US. The proposed protocol provides *pseudonymous* privacy to the patients, and protects the identity as well as the prescription patterns of doctors. The patient's privacy relies on a tamper-resistant smartcard solution based on conventional public key certificates [IT05]. The doctors' privacy however is based on a group signature scheme, allowing them to issue prescriptions to patients on behalf of an accredited group of doctors. The doctors' anonymity can be revoked by an escrow party, and all the prescriptions issued by a given doctor are linkable to each other by the insurance company. Prescription linkability is an added feature in [AdM02], and is intended to allow insurance companies to gather statistics. The protocol we propose uses privacy-preserving credentials equipped with a selective disclosure feature, and provides stronger privacy guarantees for the patient and doctors. Moreover our protocol is computationally more efficient than that of [AdM02] owing to the higher performance of credential systems in comparison with group signatures.

Yang, Han, Bao, and Deng proposed in [YXFD04] a smartcard-enabled electronic prescription system compatible with the healthcare system in the US; this system is similar to that of [AdM02]. They also present a signature delegation feature that allows a patient to authorize a delegate (e.g., family member) to pick up prescribed medicines, and sign a reception pad on the patient's behalf, without the patient giving his signing key to the delegate. Unlike the construction of Ateniese and de Medeiros, the scheme in [YXFD04] advocates for storing all patient health data on the smartcard in order to facilitate patient mobility, and spare doctors the burden of querying remote medical databases through an unreliable network. The smartcard in [YXFD04] is also used to store patient signing keys and certificates, and to compute signatures. While the smartcard paradigm is interesting in many ways, the protocol as described in [YXFD04] makes the security and privacy of patients completely dependant on the tamper-resistance of the card. As argued in [Bra00, Section 6.1], relying uniquely on the tamper-resistance of the card is in general not a recommendable approach. Moreover, the construction in [YXFD04] is such that the identity of the pharmacist is fixed by the doctor at the time of issuing the prescription. This is clearly too restrictive from the patient's point of view, since no alternative is given if the patient cannot obtain all prescribed medicine from the designated pharmacist, or if he decides to fill his prescription at a non-predetermined location, while travelling for example. Moreover, allowing doctors to designate a particular pharmacist at prescription issuing time, may result in kickback schemes between doctors and pharmacists.

In [YDB06], Yang, Deng, and Bao presented a password-based authentication scheme for healthcare delivery systems. The rationale behind their scheme is to allow patients to authenticate to healthcare providers using long-term short passwords, as opposed to public-key certificates which assume the existence of a public key infrastructure. It is well known however that password-based authentication systems are vulnerable to dictionary attacks [BM93, GLNS93]. To protect against dictionary attacks, the authors in [YDB06] proposed a special network architecture with a front-end *service server* known to the users, and a back-end *control server* hidden from the users. To authenticate to the system, a user interacts with the service server, which in turn cooperates with the control server in order to validate the authentication request. The system in [YDB06] is purely for authentication purposes; it provides no privacy for the patient, and does not consider issues such as controlling access to health data.

5.3 Brief overview on the Belgian healthcare system

A typical workflow in the Belgian healthcare system involves a *doctor*, a *patient*, a *pharmacist*, a *Medical Prescription Administration (MPA)*, a *Health Insurance Institute (HII)*, a public safety organization, denoted *IFEB* [2], and a social security organization, denoted *RIZIV* [3]. Every patient is member of one of the existing HIIs. Every pharmacist is attached to one of the existing MPAs. The latter is called the pharmacist's local MPA. An MPA processes all the prescriptions filled by its client pharmacists, and plays the role of an intermediary between pharmacists and the patients' HIIs. The role of an MPA is similar to a router, in that it sorts out received prescriptions by HII, and then forwards them in batch to the right HIIs. Figure 5.1 gives a summary of the overall system.

A basic scenario in the Belgian healthcare system is as follows. The patient visits a doctor and receives a prescription. The patient then takes his prescription to a pharmacist. The pharmacist checks the validity of the prescription, and charges the patient a portion[4] of the cost. The remaining cost of the prescription will be paid for by the patient's Health Insurance Institute (HII). The pharmacist delivers the prescribed medicine to the patient, and forwards a copy of the prescription as well as an invoice to his local MPA. The MPA in turn processes the received data and forwards it to the patient's HII. The patient's HII checks the validity of the data, updates the patient's

[2] "Instituut voor Farmaco-Epidemiologie van België" (*Belgian Institute of Pharmaco-Epidemiology*)

[3] "RijksInstituut voor Ziekte- en InvaliditeitsVerzekering" (*National Institute for Health and Disability Insurance*)

[4] The size of this portion is determined by the patient's social security status.

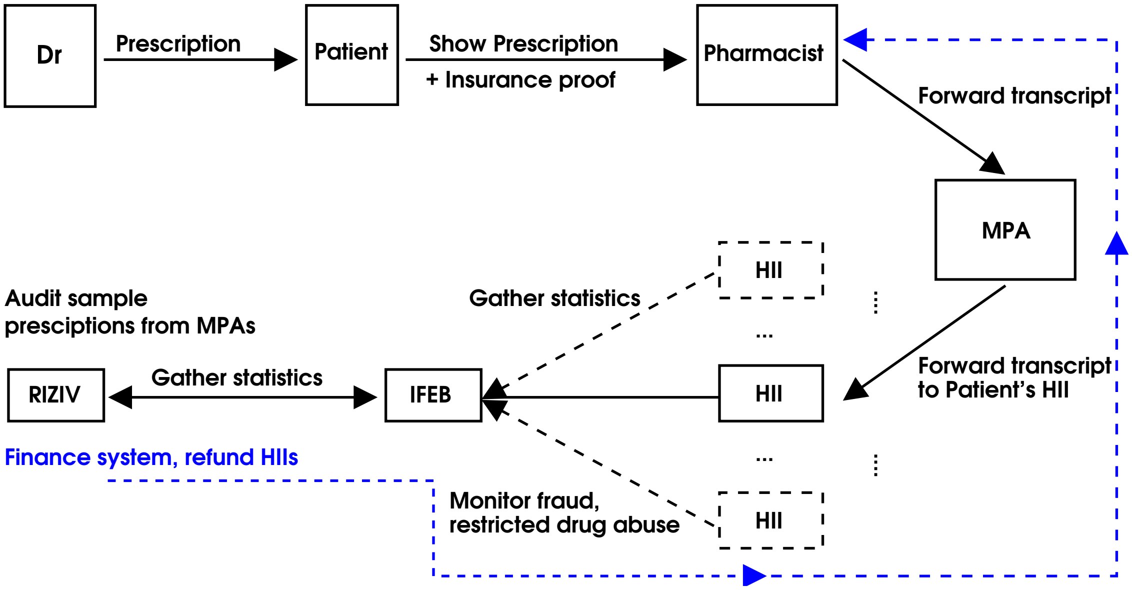

Figure 5.1: Overview of the Belgian Healthcare System

records (e.g., total medical expenses so far this year) and sends a reimbursement back to the MPA, who in turn relays it to the pharmacist.

Concurrently with executions such as the one above, the IFEB gathers statistical data from MPAs and interprets it. The IFEB also watches for fraud instances involving restricted drugs such as methadone. The RIZIV also plays a major role in the Belgian healthcare system. It finances the healthcare system by compensating the HIIs. In addition, the RIZIV oversees the overall healthcare system by retrieving and auditing sample prescriptions from the MPAs. The RIZIV is assumed to have direct access to the IFEB database.

System Model. Each player in the system above possesses a number of identity attributes. We describe the most important ones in the following.

Doctor: has a credential certificate *DrCred* asserting that he is allowed to practise as a doctor. The Doctor has a unique identifier *DrID*, and a pseudonym *DrNym*. The correspondence between *DrID* and *DrNym* is known only to a trusted oversight authority such as the ”College of Physicians”. The Doctor’s credential *DrCred* contains *DrID* and *DrNym* in addition to other identity attributes.

Patient: has an identifier *PtID*, and a social security status *PtSSS*. In addition, the patient has a “health expense account” *PtAcc* maintained by his HII. The latter is denoted *PtHII*. The value of *PtAcc* indicates the amount the patient has spent sofar in the current year on health expenses. Admissible health expenses charged to the patient beyond a predetermined maximum amount will be covered by the HII. Finally, the patient has a pseudonym *PtNym*. The correspondence between *PtID* and *PtNym* is known only to the patient’s HII. In summary, the patient’s credential contains the attributes {*PtID, PtNym, PtHII, PtSSS, PtAcc, ...*}

Pharmacist: has an identifier *PharmID*, and a corresponding MPA, denoted *PharmID_MPA*. The pharmacist’s credential contains a number of attributes, including *PharmID* and *PharmID_MPA*.

MPA: has a publicly known identifier *MPA_ID*, and a credential certifying its identity. The MPA serves a set of pharmacists, and generates statistics on prescription data on request from authorized organizations such as IFEB.

HII: has a publicly known identifier *HII_ID*, and a credential certifying its identity. The HII maintains the health expense accounts (*PtAcc*) of affiliated patients, and covers their admissible medical expenses.

IFEB: has a publicly known identifier *IFEB_ID*, and a credential certifying its identity. It gathers statistics, and conducts studies on public safety.

RIZIV: has a publicly known identifier *RIZIV_ID*, and a credential certifying its identity. It performs various oversight activities, and controls organizations such as IFEB.

5.4 Requirements

In this section, we discuss the main security and privacy properties we want to achieve in the proposed e-health protocol. The functional requirements can be easily derived from the workflow described in the previous section.

5.4.1 Security requirements

General security requirements.

- **Entity authentication (S1).** All parties should be able to properly authenticate each other. No party should be able to succeed in claiming a false identity, or false information about his identity.
- **Item integrity (S2).** Transcripts generated during the prescription lifecycle cannot be tampered with, without the tampering being detected with an overwhelming probability.
- **Revocability (S3).** It should be possible to revoke the credentials as well as the anonymity/pseudonimity of parties abusing the system.

Security requirements specific to the Belgian healthcare system.

- **Multiple prescription issuance detection capability (D1).** Oversight authorities such as the RIZIV should be able to detect malicious patients who visit multiple doctors for the same illness in order to get multiple prescriptions of a particular drug.
- **Single prescription spending (D2).** A patient must not be able to fill the same prescription multiple times.

- **Prescription non-transferability (D3).** It should not be possible for a party to fill a prescription, if he is not the patient to whom the prescription was originally issued.
- **Inappropriate prescription patterns detection capability (D4).** It should be possible to detect doctors who systematically prescribe expensive medicines (instead of generic, and hence, cheaper ones), or doctors who prescribe significantly more medicines of a certain type (e.g. antibiotics) despite known counter-indications, etc. In such cases, the doctors involved might be served a warning, or an investigation might be initiated.
- **Correct pharmacist reimbursement (D5).** A pharmacist who is not correctly refunded by the MPA, should be able to prove it in order to be compensated.
- **Payment fraud detection capability (D6).** The pharmacist should be refunded only if he has indeed delivered the medicine to the patient. It should be possible to detect pharmacists who claim expenses for non delivered medicine.
- **Correct statistics (D7).** The IFEB must be ensured that the received statistics are correct.

5.4.2 Privacy requirements

- **Minimum disclosure (P1).** During a medical consultation, the patient and doctor should be able to selectively (and provably) reveal to each other any property or predicate about their respective identities. In addition, parties involved in the prescription processing workflow should not be able to learn any information about the patient and doctor except what the latter willfully disclose to them. Data exchanged during the e-health protocol execution should satisfy the access control requirements defined in table 5.1.

- **Patient unlinkability (P2).** Prescriptions issued to the same patient should not be linkable to each other, except by the patient's HII, or by the doctor (if the patient accepts to reveal such information to the doctor.) On the other hand, two patient prescriptions that cross the same MPA should be linkable to each other, but not to the patient's identity.
- **Patient untraceability (P3).** No party involved in the prescription workflow, except the HII and RIZIV, should be able to determine the identity of the patient. The RIZIV identifies patients only in case of abuse.
- **Absence of provable doctors' prescription behaviour (P4).** To prevent elicit kickbacks and bribery between doctors and pharmaceutical companies, pharmacists should not be able to provide evidence to pharmaceutical companies about doctors' prescription behaviour.

Table 5.1 summarizes the privacy requirements, and indicates which data each party is allowed to access.

5.5 Building blocks: brief overview

5.5.1 Commitments

A commitment scheme [Ped91b, DF02] allows a committer to encode a set of attributes inside a token, also called commitment. The encoding process is known as the *commitment phase*. A commitment scheme is said to be *hiding* if a party who sees the commitment cannot learn information about the underlying attributes. A commitment scheme can be unconditionally, statistically, or computationally hiding [Gol01]. Later the committer can open the commitment by revealing the underly-

Table 5.1: Access control matrix

Party\Data	**Patient**	**Presc.**	**Doctor**	**Pharm.**	**MPA**	**HII**
Patient	ID (trivial)	all content	ID	ID	ID	ID
Doctor	nym	PrescID, data (trivial)	ID (trivial)	—	—	—
Pharm.	ss status	data	ID (if anomaly)	ID (trivial)	ID	—
MPA	nym, ss status	PrescID, data	nym	ID	ID (trivial)	ID
HII	ID	PrescID, cost	—	—	ID	ID (trivial)
IFEB	nym, ss status etc.	anon. stat. data	nym	geog. location	—	—

ing attributes. This is called the *opening phase*. A commitment scheme should be such that the committer cannot open the commitment to a set of attributes that is different from the one initially committed to in the commitment phase. This is known as the *binding* property. A commitment scheme can be unconditionally, or computationally binding [Gol01].

Notation. For a commitment *comm* with attributes $(x_1, \cdots, x_p)$, the expression *comm*.x_j denotes the j^{th} attribute embedded in *comm*. To further conceal the values of the attributes underlying a commitment, one of the embedded attributes can be chosen at random and used as a blinding factor. One way to open a commitment is by revealing the attributes embedded in it. The latter is called opening information, and denoted *openInfo*.

5.5.2 Digital credentials

A digital credential issued to user U is typically a set of assertions made by a certification authority about the identity attributes of U. To be viable, a credential system should satisfy a number of security properties such as unforgeability, and integrity. These properties are further discussed below. The X.509 public key certificate standard [IT05] is a well known example of digital credentials.

Privacy-preserving digital credentials [Bra00, CL02b, CL04], which we have discussed in Chapter 2, represent a more elaborate type of credentials, also referred to as *anonymous* credentials. In addition to the usual security properties necessary for traditional digital credentials, privacy-preserving credential systems possess a number of properties intended specifically to protect the identity of honest credential holders. Among these, we note *selective disclosure*, *token untraceability*, *tokens unlinkability*, *multi-show unlinkability*, *limited-show untraceability*, and *signed audit trails* [Bra00, CL02b, CL04]. Privacy-preserving credentials are used as a major building block in the protocol we propose in this Chapter.

Notation. For a credential *Cred* with attributes $(a_1, \cdots, a_n)$, the expression *Cred*.a_ℓ denotes the ℓ^{th} attribute of *Cred*. For example if we assume that the Doctor has an anonymous credential denoted *DrCred*, then *DrCred.ID* and *DrCred.exp* denote the identifier and expiry date of *DrCred* respectively.

Let A be a party holding an anonymous credential *Cred* and commitment *comm* encoding attributes $(a_1, \cdots, a_n)$ and $(x_1, \cdots, x_p)$ respectively. Party A can selectively disclose any information about the attributes underlying *Cred* and *comm*. Given: (1) a predicate $\mathcal{P}$ on attributes $(a_1, \cdots, a_n)$ and $(x_1, \cdots, x_p)$, and (2) a message m; the expression SPK$\{\mathcal{P}(a_1, \cdots, a_n, x_1, \cdots, x_p)\}(m)$ denotes a signed proof of knowledge on message m, of attributes $a_1, \cdots, a_n$, $x_1, \cdots, x_p$ underlying *Cred* and *comm* respectively, and satisfying predicate $\mathcal{P}$. The expression SPK$\{$*comm.DrID* $==$ *DrCred.ID* $\wedge$ *DrCred*$.exp \geq$ today$\}(m)$ for example, denotes a signed proof of knowledge on message m, where the prover convinces a verifier that the following holds: (1) he knows all the attributes underlying *comm* and *DrCred*, (2) that the *ID* embedded in *DrCred* is the same as the one embedded in *comm*, and (3) that credential *DrCred* has not yet expired.

5.5.3 Verifiable encryption

A verifiable encryption scheme (e.g., [CS98]) for a relation R, is a protocol that allows a prover to convince a verifier that a ciphertext is an encryption of a value w under a given public key such that w satisfies R, and no other information about w is disclosed. In a verifiable encryption scheme, the ciphertext is checked with respect to a public key associated with a known "decryptor".

Notation. The expressions $\mathit{VEnc}_A(\cdot)$ and $\mathit{Enc}_B(\cdot)$ denote the verifiable encryption under party A's public key, and the conventional public-key encryption under party B's public key, respectively.

Let $\mathcal{M}$ be the message space of $\mathit{VEnc}(\cdot)$ the verifiable encryption scheme, and let $\mathcal{P}$ be a Boolean predicate on elements of $\mathcal{M}$. The expression

$$vc = \mathit{VEnc}_{\text{RecID}}(m)\{\mathcal{P}(m)\}$$

denotes the verifiable encryption of m under the public key of RecID, the intended recipient. Given the public key of RecID, any verifier can be convinced that vc is an encryption under the public key of RecID, of a non-disclosed message that satisfies predicate $\mathcal{P}$.

5.6 The proposed protocol

5.6.1 Setting

Based on the system model and requirements described in Sections 5.3 and 5.4, we made a number of choices regarding the type of credentials needed by each participant involved in the e-health protocol. The patients and doctors are widely considered as private entities with high privacy expectations. We therefore provide them with anonymous credentials. The other parties are all public entities, and we think it is sufficient to supply them with conventional X.509 public key certificates. These credential choices are summarized in Table 5.2.

Table 5.2: Credential material per participant

Party / Cred. type	Patient	Dr.	Pharm.	MPA	HII	IFEB	RIZIV
Anon. Cred.	✓	✓					
X.509 Cert.			✓	✓	✓	✓	✓

The credentials of the MPAs, HIIs, RIZIV, IFEB, and pharmacists are issued by trusted government-approved certification organizations. The doctors' credentials are issued by a medical certification authority such as the college of physicians. The patients' credentials are issued by a central government-approved certification authority *CA*. The patient's pseudonym *PtNym* embedded in the patient's credential is not known to the *CA*. The correspondence between *PtNym* and *PtID* is known only to the patient's HII. Issuing anonymous credentials on secret but committed attributes is easily done by standard techniques such as those in [Bra00, CL02b].

5.6.2 Protocol description

I. Doctor (Dr.) ↔ Patient (Pt.)

a) Dr. anonymously authenticates to Patient using his *DrCred*.

b) Patient computes commitment $com_{Pt} := comm(PtID)$,

c) Patient anonymously authenticates to Doctor using his credential *PtCred*. Moreover, Patient sends com_{Pt} to Doctor, and proves that $com_{Pt}.PtID == PtCred.PtID$

d) Dr. computes commitment $com_{Dr} := comm(DrNym)$

e) Dr. sets $Presc_text := \{$plain prescription text$\}$

f) Dr. computes the prescription's serial number *PrescID*, e.g., as a hash of $Presc\_text$, com_{Pt}, and com_{Dr}.

g) Dr. computes

$$Presc := \text{SPK}\{DrCred.DrNym == com_{Dr}.DrNym\}(Presc\_text, PrescID, com_{Dr}, com_{Pt}),$$

and sends it to the patient, along with the opening information of com_{Dr}.

II. Patient ↔ Pharmacist

a) Pharmacist authenticates to Patient using his X.509 pharmacist certificate *PharmCred*.

b) Pt. recovers *PharmCred.MPA_ID*, the identity of the MPA serving the pharmacist.

c) Pt. anonymously authenticates to Pharmacist using *PtCred*, and provably discloses his social security status.

d) Pt. computes :

i. $vc_1 = VEnc_{\text{MPA}}(PtHII)\{PtHII = PtCred.PtHII\}$

ii. $vc_2 = VEnc_{\text{MPA}}(DrNym)\{DrNym = Presc.com_{Dr}.DrNym\}$

iii. $vc_3 = \mathit{VEnc}_{\text{RIZIV}}(\mathit{PtNym})\{\mathit{PtNym} = \mathit{PtCred}.\mathit{PtNym}\}$

iv. $vc'_3 = \mathit{VEnc}_{\text{RIZIV}}(\mathit{PtHII})\{\mathit{PtHII} = \mathit{PtCred}.\mathit{PtHII}\}$

v. $vc_4 = \mathit{VEnc}_{\text{MPA}}(\mathit{PtNym})\{\mathit{PtNym} = \mathit{PtCred}.\mathit{PtNym}\}$

vi. $vc_5 = \mathit{VEnc}_{\text{PtHII}}(\mathit{PtNym})\{\mathit{PtNym} = \mathit{PtCred}.\mathit{PtNym}\}$

vii. $c_5 = \mathit{Enc}_{\text{MPA}}(vc_5)$

e) Pt. sends to pharmacist :

i. *Presc* and SPK$\{\mathit{PtCred}.\mathit{PtID} == \mathit{Presc}.\mathit{com}_{Pt}.\mathit{PtID}\}$(nonce)[5]

ii. $vc_1, vc_2, vc_3, vc'_3, vc_4, c_5$ [6]

f) Pharmacist checks if *Presc*, SPK, and $vc_1, vc_2, vc_3, vc'_3, vc_4$ are correct. If all is correct then continue, else abort. If *Presc* contains an anomaly (e.g. unusual or possibly lethal dosage), the pharmacists asks Pt. to name the doctor. The pharmacist will contact the doctor to correct the problem.

g) Pharmacist charges patient, gets paid, and delivers drug.

h) Pharmacist issues an invoice to Patient with the prescription's serial number *PrescID* embedded in it.

i) Patient computes:

$$\mathit{reception\_ack} := \text{SPK}\{\mathit{PtCred}\}(\mathit{PrescID}, \mathit{PharmID}, vc_1, vc_2, vc_3, vc'_3, vc_4, c_5)$$

and sends it to Pharmacist. This proves that the patient has indeed received the medicine from the pharmacist.

j) Pharmacist checks if *reception_ack* is correct. If correct continue, else abort.

[5] The nonce can be chosen jointly by the patient and pharmacist, and may include information such as date, *PharmID*, etc.

[6] The patient Pt. sends c_5 to the pharmacist instead of vc_5, because Pt. wants to hide the identity of his HII from the pharmacist. In Belgium, health insurance institutes (HIIs) are managed by socio-political groups, and revealing the identity of a patient's HII, may disclose personal information about the patient's political inclination for example. That is why in the protocol above, the patient hides the identity of his HII from the pharmacist. Only the MPA (downstream) needs to know the identity of the patient's HII. The correctness of $vc_5 = \mathit{Dec}_{\text{MPA}}(c_5)$ will be checked by the MPA, prior to forwarding it to the right HII. Additional data that may be useful for statistics, such as *PtAge*, can be handed to the MPA inside vc_4.

III. Pharmacist ↔ MPA (*PharmCred.MPA_ID*)

a) Pharmacist and MPA mutually authenticate

b) Pharmacist forwards to MPA *Presc*, $vc_1, vc_2, vc_3, vc'_3, vc_4, c_5$, and *reception_ack*.

c) If all is correct, the MPA continues. Else if $\mathit{Dec}_{\mathrm{MPA}}(c_5)$ is incorrect, then forward vc_3, vc'_3, and rest of transcript to RIZIV and request patient deanonymization.[7]

d) MPA computes:

i. $\mathit{PtNym} = \mathit{Dec}_{\mathrm{MPA}}(vc_4)$,

ii. $\mathit{PtHII} = \mathit{Dec}_{\mathrm{MPA}}(vc_1)$,

iii. $\mathit{DrNym} = \mathit{Dec}_{\mathrm{MPA}}(vc_2)$,

iv. $vc_5 = \mathit{Dec}_{\mathrm{MPA}}(c_5)$

e) MPA adds a DB entry indexed by *PrescID*, *PtNym*, *DrNym*, and stores any information relevant to the prescription.

IV. MPA ↔ HII (*PtHII*)

a) MPA and HII mutually authenticate

b) MPA forwards *reception_ack* and vc_5 to the patient's HII

c) HII checks the integrity of *reception_ack* and vc_5

d) If correct, HII recovers $\mathit{PtNym} = \mathit{Dec}_{\mathrm{HII}}(vc_5)$, else abort and forward transcript to RIZIV for patient deanonymization.

e) HII recovers *PtID* corresponding to *PtNym*

f) HII updates patient *PtID*'s account *PtAcc* with proper amount

g) HII sends reimbursement amount due to the MPA, along with the corresponding invoice containing *PrescID*.

[7] The RIZIV first recovers *PtNym* and *PtHII* from vc_3 and vc'_3, then files a complaint with the judicial authorities who can subpoena the HII to provide the real identity of *PtNym*.

h) HII creates a database entry for the processed invoice with information such as *PtID*, *PrescID*, prescription cost, date etc.

i) After receiving the refund from the HII, the MPA compensates the pharmacist.

V. IFEB ↔ MPA

a) MPA and IFEB mutually authenticate

b) IFEB requests statistics

c) MPA provides statistics on prescription data anonymized according to the privacy laws in place.

The data available to the MPA is identified only by the pseudonyms of the Doctor and the Patient. This data is sufficient to generate meaningful statistics, including measurements requiring the aggregation of prescription data per patient or per doctor. The data available to the MPA, and the subsequently released statistics do not compromise the real identities of patients or doctors.

Alternatively, the IFEB can obtain statistics from the HIIs. This can be done without weakening the privacy of the patient or inducing additional disclosures, since the HIIs already know the prescription data of their affiliated patients. The IFEB first queries the different HIIs for a specific statistical measurement, and then aggregates the separate *anonymized* results to derive the global measurement for the whole population. Data from the HIIs can also be used to double-check the accuracy of statistics collected from the MPAs.

Remarks

- In step I-[g] the Doctor computes the prescription as a signed proof of knowledge on the tuple $(Presc\_text, PrescID, com_{Dr}, com_{Pt})$. The predicate being asserted in the proof is that com_{Dr} contains the same attribute *DrNym* embedded in *DrCred*. This has the following implications:
  - Because the prescription is a signed proof, any one can check its validity non-interactively.
  - The prescription is tied via (com_{Dr}, com_{Pt}) to the identity of both the Doctor and the Patient. Recall that the Doctor issues the prescription only if the value of *PtID* underlying com_{Pt} is consistent with *PtCred* (the consistency proof was performed by the Patient in step I-[c].)
  - The Doctor discloses the opening information of com_{Dr} to the patient, to allow him to verifiably encrypt *DrNym* under the public key of the pharmacist's MPA (in step II-[d-ii].) Note that the Doctor cannot encrypt *DrNym* in advance since the identity of the pharmacist where the patient will buy his medicines is usually not known at the time of the prescription issuing.

5.7 Protocol evaluation

In the following, we provide a brief analysis of the security of our protocol. We assume that all the underlying building blocks are secure.

5.7.1 General Security Requirements

- **Entity authentication (S1).** This property follows immediately from the soundness and unforgeability of the underlying anonymous and public key certificates.

- **Item integrity (S2).** All binding data (e.g., prescription, acknowledgement, verifiable encryptions) exchanged during the protocol presented in section 5.6, are signed either by a conventional public key signature or signed proof of knowledge, and are therefore resistant to any tampering.

- **Revocability (S3).** In case of abuse (which can be detected either by the MPA, the patient's HII, or the RIZIV), the user's identity is unveiled by opening one of the verifiable encryptions vc_3, vc_4, or vc_5. It is then possible to revoke the patient's credentials and prescriptions using a blacklist for example.

5.7.2 Security requirements specific to the Belgian healthcare system.

- **Multiple prescription issuance detection capability (D1).** When filling a prescription, the patient reveals information that will allow the MPA to recover his pseudonym (as shown in step II-[d-iv]). Because multiple prescriptions issued to the same patient are linked to each other through the patient's pseudonym, oversight organizations such as the RIZIV or IFEB are able to detect abusive behaviour and stop malicious patients.

- **Single prescription spending (D2).** Follows directly from the fact that prescriptions are uniquely identified by a *PrescID*, and resistant to tampering.

- **Prescription non-transferability (D3).** This Follows from the soundness of the signed proofs of knowledge in step II-[e] of the protocol. The patient proves to the pharmacist that the pseudonym in the prescription corresponds to the pseudonym embedded in the patient's credential *PtCred*.

- **Prescription fraud detection capability (D4).** This can only be detected by the RIZIV by searching for abnormal behaviour in the IFEB database. The IFEB database contains only doctor pseudonyms, which can be linked to doctors' real identities with the help of an authority such as the "college of physicians".

- **Correct pharmacist reimbursement (D5).** For each prescription, the patient generates a *reception_ack*, which can be considered as a confirmation from the patient that he received services from the pharmacist. This proof is verified and stored by the pharmacist, MPA and HII. In case reimbursement problems are detected, this proof can be used as evidence.

- **Payment fraud detection capability (D6).** The acknowledgement *reception_ack* issued by the patient guarantees to the HII that the patient has indeed received the medicine.

- **Correct statistics (D7).** The IFEB needs to rely on the trustworthiness of the MPAs to make sure it receives correct statistics. The latter property cannot be enforced by cryptographic means alone, since a malicious MPA could just ignore half of the transactions it has recorded. For better assurances, oversight organizations such as the RIZIV, can in practice request random sample data from health insurance institutes (HIIs) and cross-check them against data returned by the MPAs. A malicious MPA who fails to return consistent data, or returns incomplete data, can be further investigated and may have its licence revoked if proved guilty.

5.7.3 Privacy

- **Minimum disclosure (P1).** Owing to the selective disclosure feature offered by the signed proofs of knowledge, the security of the commitment and verifiable encryption schemes, the protocol presented in section 5.6 satisfies the access control requirements of table 5.1. This can be easily verified by simple examination of the protocol.

- **Patient unlinkability (P2).** Prescriptions are tied to the patient's pseudonym *PtNym* which can be recovered only by the MPA processing the prescription and the patient's health insurer (PtHII). All other parties have no access to the patient's identity or pseudonym, and thus cannot link any two prescriptions of the same patient. In the case of a treating doctor, the patient may freely decide to disclose his pseudonym to allow the linkability.

- **Patient untraceability (P3).** An examination of the protocol we presented in section 5.6 shows that the identity of the patient is accessible only to the patient's HII who knows the correspondence between *PtNym* and *PtID*. In case of apparent abuse, the RIZIV may also have access to the patient's identity by filing a complaint with the judicial authorities who can subpoena the HII to deanonymize the fraudulent patient.

- **Absence of provable doctor prescription behaviour (P4).** The protocol is designed in such a way that the real identity of a doctor is never associated with the prescriptions' content. The only exception occurs when the pharmacist sees a prescription anomaly (e.g. lethal dosage), in which case he asks the patient to name the doctor. This information however is not a reproducible proof, and thus cannot be used to convince a bribe-giver.

5.8 Enhancing the patient's control over their data

Further improvements can be made to the protocol we presented in section 5.6.2. For example one could enhance the patient's privacy, by using techniques that allow the patient to control access to his remotely stored health records according to self-chosen privacy policies. More precisely, in the protocol we outlined in section 5.6.2, the doctor queries a central medical database about the patient's records. In the current protocol, the doctor is trusted to query only the medical records of his patients, and to query them only on a need-to-know basis. To avoid relying on these trust assumptions, it may be desirable to modify the protocol in such a way that the health records are accessible by the doctor only if the latter has a valid permission from the patient owning the records. While adding this layer of access control, it is important that we avoid revealing access patterns to the database manager. In other words, for every query, the database manager should be convinced that the doctor is allowed to access the record targeted in the query, without learning the index of the record being queried. This prevents the leaking of sensitive information of the form: *A Doctor specialized, say in narcotics, queried the records of patient X*. Such information may imply that patient X may have a substance abuse problem. It is clear that leaking this kind of information can cause great harm to the patient.

To prevent the disclosure of sensitive health data and improve the privacy of patients, we can use the Accredited SPIR techniques described in Chapters 3 and 4 in the following way. For a self-chosen privacy policy, the patient gives an "ASPIR" authorization to his doctor to have access to his health records possibly stored on a medical database under the control of a health insurance institute, or the IFEB. The doctor is then able to retrieve the patient's records according to patient's privacy policy, without the database manager learning which record the doctor is looking at, and thus which patient he is treating.

5.9 Concluding remarks

We have presented a privacy-preserving protocol for the Belgian healthcare system. The proposed protocol protects patients' privacy throughout the prescription handling process, while complying with the current Belgian practise. Despite the large number of parties involved, and the complexity of the application, the protocol we presented minimizes information disclosure and satisfies the access control requirements of table 5.1. Furthermore, our protocol is equipped with a set of abuse detection and evidence gathering mechanisms that allow oversight authorities to address fraudulent activities and ensure accountability. In addition to protecting patients' privacy, our protocol provides a mechanism to prevent the intrusive monitoring of doctors' prescription patterns. The ability of third party players to determine the prescription patterns of a given doctor is often considered an undesirable aspect in healthcare, since it can be used sometimes by malicious pharmaceutical companies either, as a coercive tool against doctors who do not prescribe their products, or as an instrument to facilitate bribery and kick-backs to doctors who promote their products (see [Bio08, Bio03] for example). In our protocol, doctors are only pseudonymously identified to allow the legitimate gathering of statistical data about medicine consumption and its effect on the population. The real identity of the doctors is unveiled, via a judicial procedure, only in case of apparent abuse.

The design we proposed is highly modular and can be adapted to other healthcare systems comparable to the Belgian one. For example, if we consider a jurisdiction where the real identity of the doctor (as opposed to his pseudonym) has to be indicated in plain-text in all transcripts generated during the prescription lifecycle, then one can easily adapt our protocol to the new setting by replacing the *DrNym* attribute in the authentication step of phase I, with the *DrID* attribute already embedded in the doctor's credential. The rest of the protocol can be easily modified accordingly.

Further improvements can be made to the presented protocol. For example, one could strengthen the patients' privacy by allowing him to control access to his remotely stored health records according to self-chosen privacy policies. This can be achieved using the Accredited SPIR techniques described in Chapters 3 and 4. Another worthy avenue for future work would be to simplify the prescription workflow and reduce interactions (to the extent acceptable by the healthcare procedures and practices in place).

Chapter 6

Summary and Conclusion

The goal of this thesis has been to devise new ways to enhance users' privacy. To that end, we first surveyed the state of the art in privacy-preserving credential systems. The survey gives a comparison of the most representative credential systems available in the literature. It also describes the properties and features of interest to a privacy-preserving credential system. Moreover, the survey draws a historical timeline showing the sequence of contributions that helped advance the field of privacy.

In the second part of the thesis, we build upon privacy-preserving credential systems—a technology that allow users to selectively disclose information lying under their control—and propose a technique for controlling access to remotely stored user-data, in a privacy-preserving way. The proposed technique is called Accredited Symmetrically Private Information Retrieval, and uses credentials by Brands [Bra00] and a SPIR scheme by Lipmaa [Lip05].

We further extended the ASPIR scheme above, to a setting where database records belong to multiple owners simultaneously. We presented a protocol that allows record owners (data-subjects) to grant access to their records, to self-approved parties, without the database manager being able to learn if and when their records are accessed.

We provide constructions that allow a Receiver party to retrieve a database record only if he has authorizations from all owners of the target record (or, from a subset of the owners of size greater than a given threshold.) We also provided a construction where owners of the same record do not have equal ownership rights, and the record in question is retrieved using a set of authorizations consistent with a general access structure. The proposed constructions are efficient and use a pairing-based signature scheme. It is worth noting also that unlike the initial ASPIR protocol which worked only with the Lipmaa's SPIR system, the new construction we presented uses SPIR schemes in a black-box fashion, meaning that the new constructions work with *any* SPIR scheme.

Finally, in the last part of the thesis we applied privacy-preserving technologies to a real world setting. In particular, we considered the Belgian healthcare system, but our results can be extended to other healthcare systems and application domains. The motive behind choosing healthcare is to take on the challenge of designing privacy-friendly protocols for large and complex real world systems. In particular, we considered the Belgian healthcare system as a case study. We presented a protocol that protects the patients' privacy throughout the prescription handling process, while complying with most aspects of the current Belgian healthcare practise. The presented protocol relies on standard privacy-preserving credential systems, and verifiable public key cryptography.

It is worth mentioning here that much of the focus in this work has been on designing *practical* protocols that can be used to solve real-world problems, and that many of the security proofs were either incomplete or informal. We would like to stress however that, unless proven otherwise, there are no known ways so far to break any of the proposed protocols.

There are certainly many ways to extend the work presented in this thesis. One possible extension would be to build protocols similar to the ASPIR and Multi-Authorizer ASPIR protocols presented in chapters 3 and 4, but for a database containing encrypted data instead of plaintext data. More details on this research direction can be found in section 4.9.1.

Appendix A

Credential Systems Comparison: A Compendium

In Chapter 2 we have presented a number of credential systems, and a list of criteria to compare them. In this appendix, we compare the various features of the above systems, and evaluate their performances. A summary of this comparison is given in Table A.1. The complexity figures in Table A.1 account for small-size exponentiations[1]. Operations such as multiplications by a scalar, and additions are significantly cheaper, and are therefore considered negligeable. Pairings can be reduced to a single exponentiation of size less than the group order [Boy06], and are therefore counted as small-size exponentiations. It is worth noting that the performance figures for the CL-DL scheme, do not include the cost for the verifiable encryption.

Below we give typical system parameter values for each of the studied systems. It must be stressed however that these parameters are given only by way of illustration, and do *not* necessarily lead to system instances with equivalent levels of security. In fact not all the studied credential systems have known tight reductions to their re-

[1] Typically exponentiations with 256-bit long exponents.

spective underlying computational problems. If such reductions were available, one could use state of the art key length recommendations (e.g., [LV00, Len05, Gir08]) to obtain system instances with equivalent levels of security, and conduct a performance comparison on them.

Typical system parameters

Chaum Blind signatures. $\ell_n = 1600$, $\ell_e = 256$. The domain of the one-way function f is chosen to be 256-bit long ($f : \{0,1\}^* \rightarrow \{0,1\}^{256}$).

Chaum-Pedersen Signatures. $|p| = 1600$, $|q| = 256$.

Brands I. $|p| = 1600$, $|q| = 256$.

Brands II. $|p| = 1600$, $|q| = 256$.

CL-RSA. $\ell_n = 1600$, $\ell_m = 256$, $\ell_c = 160$, $\ell_s = 80$, and $\ell_e = 259$.

CL-DL. One can consider an elliptic curve E over $\mathbb{F}_p$, for $|p| = 257$. Let G to be a subgroup of E of order q, with $|q| = 256$. Elements of G are therefore of length $\ell_G = 257$ bit. Let G denote the target domain of the bilinear map, and let $|\mathsf{G}| = 2^{1600}$. Elements of G are 1600-bit long.

Table A.1: Comparison Matrix

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Token properties						
Token type	Signed messages	Public-key certificates	Public-key certificates	Secret-key certificates	Secret-key certificates	Secret-key certificates
Underlying assumption	RSA	DL	DL	DL	Strong RSA	DL-based LRSW
Can issuer encode attributes?	No[2]	No[2]	Yes	Yes	Yes	Yes
Can user encode hidden attributes?	Yes	Yes	Yes	Yes	Yes	Yes
Selective disclosure capabilities?	No[3]	No[4]	Yes	Yes	Yes	Yes
• Subset disclosure proofs?	n/a	n/a[4]	Yes	Yes	Yes	Yes
• NOT proofs?	n/a	n/a[4]	Yes	Yes	No[5]	No[5]

continues on next page

[2] Must use different signing keys

[3] Can be enabled for special instances of the one-way function f

[4] Could be enabled using Brands techniques

[5] Of course, it is always possible to use generic techniques, but these are not efficient

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
• General boolean statements?	n/a	n/a[4]	Yes[6]	Yes[6]	Yes[7]	Yes[7]
• Interval proofs?	n/a	n/a[4]	Yes	Yes	Yes	Yes
• Multiplicative relations?	n/a	n/a[4]	Yes[8]	Yes[8]	Yes[8]	Yes[8]
Signed transcripts?	No	Yes	Yes	Yes	Yes	Yes
Can user make signatures with token?	No	Yes	Yes	Yes	Yes	Yes
Privacy						
Is a token untraceable?	Yes[9]	Yes[9]	Yes[9]	Yes[9]	Yes[10]	Yes[10]
Are multiple tokens unlinkable?	Yes[9]	Yes[9]	Yes[9]	Yes[9]	Yes[11]	Yes[11]

continues on next page

[6] Linear relations
[7] With modest efficiency for non-linear relations
[8] Using generic non-efficient proof techniques
[9] Unconditional protection
[10] The untraceability is computational and relies on the trustworthiness of a key escrow party
[11] Statistical unlinkability

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Unlinkable multi-show for one token?	No	No	No	No	Yes[11]	Yes[11]
Recertification[12] without revealing attributes?	n/a	n/a	Yes[13]	No	Yes[14]	Yes[14]
Recertification with selective attribute updating?	n/a	n/a	Yes[13]	No	Yes[13]	Yes[13]
Selective depositing [15] capabilities?	n/a	n/a	Yes	Yes	n/a[16]	n/a[16]
Security						
Lending disincentive?	No	Yes[17]	Yes	Yes	Yes	Yes

continues on next page

[12] With no attribute updating
[13] Requires the revocation of the old credential
[14] Certificate re-blinding without the issuer's intervention (similar to the blinding performed at the showing)
[15] Can a verifier censor (with the prover's consent) disclosed attribute information from signed showing transcript?
[16] Could be enabled using Brands techniques
[17] Could be enhanced using techniques similar to those in the Brands and Camenisch-Lysyanskaya systems

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Limited-show tokens?	No	Yes[17]	Yes	Yes	Yes	Yes
k-show tokens for any k?	n/a[18]	Yes[18]	Yes[18]	Yes[18]	Yes[19]	Yes[19]
<u>Unforgeability</u>						
Verifier cannot forge or modify showing transcripts?	n/a	Yes	Yes	Yes	Yes	Yes
Issuer cannot frame users?	n/a	Yes	Yes	Yes	Yes	Yes
<u>Revocation</u>						
Can user self-revoke a token?	Yes	Yes	Yes	Yes	Yes	Yes

continues on next page

[18] The showings will be linkable

[19] Constructions are given for $k = 1$ and k indefinite. A more recent scheme [CHK+06] offers a special k-time show capability, where credentials can be shown unlinkably k times within a pre-defined time interval, for constant $k > 1$.

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Can verifier revoke a token that was shown to him?	Yes	Yes	Yes	Yes	Yes[20]	Yes[20]
Can issuer revoke or blacklist an issued token?	No	No	Yes	Yes	Yes	Yes
Can all of a user's unlinkable tokens be revoked if a showing fraud takes place with any one?	No	No	Yes	No	Yes[21]	Yes[21]
Wallet-with-observer setting						
Does a wallet-with-observer solution exist?	No	Yes	Yes	Yes	n/a	n/a

continues on next page

[20] Only if the data disclosed during the showing contains a uniquely identifying information (e.g., a serial number)

[21] Through blacklisting

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Does untraceability hold in wallet-with-observer setting?	n/a	Yes[22]	Yes	Yes	n/a	n/a
Does unlinkability hold in wallet-with-observer setting?	n/a	Yes[22]	Yes	Yes	n/a	n/a
Can observer dynamically interpret third-party policies to prevent unauthorized behavior?	n/a	Yes[23]	Yes	Yes	n/a	n/a
Can multiple issuers share one observer?	n/a	Yes[24]	Yes	No	n/a	n/a
Can inflow be controlled by user?	n/a	Yes[22]	Yes	Yes	n/a	n/a
Can outflow be controlled by user?	n/a	Yes[22]	Yes	Yes	n/a	n/a

continues on next page

[22] Relies mainly on the card's tamper-resistance

[23] It should be possible in principle

[24] If tamper-resistance is assured

Table A.1 (*continued*)

	Chaum blind signatures		Chaum-Pedersen signatures		Brands credentials issuing prot. I		Brands credentials issuing prot. II		Camenisch-Lysyanskaya SRSA-based credentials		Camenisch-Lysyanskaya DL-based credentials	
Performance[25,26]												
Issuing protocol complexity	Offline	Online	Offline	Online	Offline	Online	Offline	Online	Offline	Online	Offline	Online
• modexs for issuer	—	1	1	9	1	ℓ	1	—	1	$\ell + 3\ell'+3$	$2\ell + 4\ell'+6$	$\ell - \ell' + 1$
• modexs for user	—	1	8	17	$\ell + \ell' + 9$	1	$\ell + 4$	1	$6\ell'+4$	—	$\ell + 5\ell'+3$	—
• modexs for token verification	1		4		4		2		$\ell - \ell' + 2$		$4\ell + 6$	
• modexs for observer in wallet setting	n/a		1	14	1[27]	—	1[27]	—	n/a		n/a	
• Communication complexity	$2\|n\|$		Obs. ↔ U $10(\|p\| + \|q\|)$	U ↔ I $9\|p\|+6\|q\|$	$5\|p\| + (\ell' + 4)\|q\|$		$(\ell + 2)\|q\| + \|p\|$		$\ell'(2\ell_n+\ell_c+\ell_s)+ \ell\ell_m + 2(\ell_n + \ell_e+\ell_c)+\ell_s+21$		$(\ell + \ell' + 2)\|q\| + (2\ell + \ell' + 4)\ell_G$	

continues on next page

[25] For the CL-SRSA, CL-DL, and Brands I systems, the performance is given for tokens containing ℓ attributes, of which $\ell' < \ell$ are known only to the user.

[26] In the Brands II case, which is a secret-key certificate system, the performance is given for a token with ℓ attributes all of which are known to both the issuer and user. For the Chaum and Chaum-Pedersen systems, performance is given for mono-attribute credentials. Finally, the sign "—" indicates that no computations are needed.

[27] The observer, who holds a secret corresponding to the card's public key, assists the user in proving knowledge to the issuer, of some attributes necessary to initialize the issuing protocol. In order to prevent Oracle attacks, the observer may require the user to provide a fresh issuer-signed challenge.

Table A.1 (*continued*)

	Chaum blind signatures		Chaum-Pedersen signatures		Brands credentials issuing prot. I		Brands credentials issuing prot. II		Camenisch-Lysyanskaya SRSA-based credentials		Camenisch-Lysyanskaya DL-based credentials	
• Communication complexity (numerical)	400 bytes		Obs. ↔ U 2320 bytes	U ↔ I 1992 bytes	$1128 + 32\ell'$ bytes		$264 + 32\ell$ bytes		$518{+}430\ell'{+}32\ell$ bytes		$193 + 97\ell + 65\ell'$ bytes	
Showing protocol complexity	Offline	Online	Offline	Online	Offline	Online	Offline	Online	Offline	Online	Offline	Online
• modexs for verifier	—	1	—	4	—	$\ell + 6$	—	$\ell + 4$	—	$\ell + 3\ell'{+}2$	—	$6\ell + 3\ell'{+}9$
• modexs for user	—	—	2	2	$\ell + 1$	—	$\ell + 1$	—	$\ell + 4\ell'{+}2$	—	$2\ell + 4\ell' + 11$	—
• modexs for observer in wallet setting	n/a		1	2	1	—	1	—	n/a		n/a	
• Communication complexity	$2\|n\|$		Obs. ↔ U $2\|p\| + 3\|q\|$	U ↔ I $4\|p\| + \|q\|$	$3\|p\| + (\ell + 3)\|q\|$		$2\|p\| + (\ell + 3)\|q\|$		$\ell'(2\ell_n{+}\ell_c{+}\ell_s){+} \ell\ell_m + 2\ell_n + \ell_c$		$(2\ell + \ell' + 4)\ell_G + (\ell + \ell' + 3)\|q\|$	
• Communication complexity (numerical)	400 bytes		Obs. ↔ U 496 bytes	U ↔ I 832 bytes	$696 + 32\ell$ bytes		$496 + 32\ell$ bytes		$420{+}430\ell'{+}32\ell$ bytes		$225 + 97\ell + 65\ell'$ bytes	
Size of issuer signature on a token	$\|n\|$		$\|p\| + 2\|q\|$		$\|p\| + 2\|q\|$		$2\|q\|$		$\ell_n + 2(\ell_e + \ell_c + \ell_s + 1)$		$(2\ell + 3)\ell_G$	
Size of issuer signature on a token (numerical)	1600 bytes		264 bytes		264 bytes		64 bytes		325 bytes		$97 + 65\ell$ bytes	

continues on next page

Table A.1 (*continued*)

	Chaum blind signatures	Chaum-Pedersen signatures	Brands credentials issuing prot. I	Brands credentials issuing prot. II	Camenisch-Lysyanskaya SRSA-based credentials	Camenisch-Lysyanskaya DL-based credentials
Blacklist proof performance	n/a	n/a[28]	sublinear[29]	sublinear[29]	n/a[30]	n/a

[28] Because of the similarities in settings, blacklist proof techniques from the Brands credential systems can be used with credentials based on the Chaum-Pedersen signatures.

[29] A user can prove that his secret serial number is not on a blacklist in time sublinear (close to square root) in the length of the blacklist (see [BDDD06].)

[30] The authors provide a whitelisting construction[CL02a] instead, where the user proves anonymously that his serial number is contained in a public accumulator representing "the whitelist". Their construction requires 21 online exponentiation from the verifier, and 22 offline exponentiations from the prover, in addition to an interval proof which may take up to 20 additional exponentiations, using Boudot's technique [Bou00]. Recently, Li, Li, and Xue [LLX07] proposed a new accumulator construction, based on that of [CL02a], which allows users to make anonymous non-membership proofs. The latter construction can be used for anonymous blacklisting proofs. The construction of [LLX07] requires 25 offline exponentiations from the prover, and 39 online exponentiations from the verifier. In addition, the prover needs to make two interval proofs, which may require up to 40 extra exponentiations.

Appendix B

State-of-the-Art Credential Systems: A Detailed Overview

B.1 Chaum's blind signatures

B.1.1 Summary of security and privacy properties

In the following we give a high level overview of the security and privacy properties of Chaum's blind signatures.

Credential issuing phase

- Blinding capabilities: If the receiver follows Chaum's blind signature issuing protocol, then a signer with unlimited power, learns no information on the generated signature or the message that has been signed.
- Unforgeability: Assuming the RSA problem is hard, and f a one-way collision-free hash function, it is infeasible for a probabilistic polynomial time attacker to forge signatures on behalf of the signer.

- Other properties: Security against Oracle attacks; assuming f a one-way collision-free hash function, it is infeasible for a probabilistic polynomial time attacker to trick the signer into signing a message for which the attacker knows a pre-image through f.

Credential showing phase

- Selective disclosure: Does not apply to Chaum's basic blind signatures. This feature could be added however for special instances of the one-way function f.
- Multi-show unlinkability: The different showings of a signed token are linkable to each other. In that respect, blind signatures can be thought of as "one-show credentials".
- Security against false proofs: Assuming the one-way function f collision-resistant, a user cannot prove a false predicate about his credential's secret with non-negligeable probability.
- Impersonation resistance: Chaum's blind signatures are not protected against impersonation, since anyone who learns the value of the token can use it on behalf of the user to whom the token was initially issued.

Transcript depositing phase

- Untraceability: Showing a signed token, obtained through Chaum's blind signature issuing protocol, to a verifier does not help the latter link the shown signature to the instance of the signing protocol that produced it, even if it colludes with the signer.
- Multi-show unlinkability: multiple showings of a token obtained through Chaum's blind issuing signature protocol, are linkable to each other.

- Limited-show capabilities: Chaum's blind signatures do not offer a mechanism to enforce a limit on the maximum number of times a token can be shown.
- Framing resistance: In the standard construction, tokens obtained through Chaum's blind signature do not contain any attribute that binds them to any particular user. As a result, no one can claim that a token has or has not been shown by a particular user.

B.2 Protocols for the Chaum-Pedersen credential system

B.2.1 Credential issuing

A user first generates a random pseudonym h of its choice, and submits it to the issuer, who signs it. The issuer's signature on h, called *validator*, is considered as a credential attributed to the user's wallet on pseudonym h. The pseudonym generation and signing procedure works as follows. First, the wallet observer T and user-controlled computing module C jointly choose a random secret key $x \in \mathbb{Z}_q$, and compute the corresponding public key $h := (g^x \mod p)$. The secret x is only known to T. Next, T and C, engage in a blind signature protocol with the issuer, to obtain a signature $\sigma_{CA}(h)$ on their public key h. For a party Z with secret key x_Z and public key $h_Z := g^{x_Z}$, a signature on a message m consists of $z := m^{x_Z} \mod p$ and a proof that $\log_g h_Z = \log_m z \mod q$. This is done as shown in Figure B.1.

Using the Fiat-Shamir heuristic, the signature protocol of Figure B.1 can be made non-interactive by computing $c := H(m, z, a, b)$. The new verification relation is $c \overset{?}{=} H(m, z, h_Z^{-c} g^r, z^{-c} m^r)$.

User	Public Info: $(h_Z = g^{x_Z})$	Issuer Z (x_Z)
	$\xrightarrow{m}$	$z := m^{x_Z}$
		$s \in_R \mathbb{Z}_q$
	$\xleftarrow{(z,a,b)}$	$(a,b) := (g^s, m^s)$
$c \in_R \mathbb{Z}_q$	$\xrightarrow{c}$	
$g^r \overset{?}{=} ah_Z^c$	$\xleftarrow{r}$	$r := s + cx_Z$
$m^r \overset{?}{=} bz^c$		
$\sigma_Z(m) := (z, r, c)$		

Figure B.1: Basic Chaum-Pedersen signature scheme

In the basic protocol of Figure B.1, the signer may create a subliminal channel with the wallet by encoding covert information in the pair (a, b). This can be prevented by interposing the user-controlled computing module C, between the signer and the wallet observer T. This can be done by letting C contribute to the selection of the randomness s, used to compute (a, b), in such a way that neither C, nor the signer can influence the final value of s and hence, the values of a and b. The method is illustrated in Figure B.2.

Wallet (m)	Public Info: $(m, h_Z := g^{x_Z})$	Issuer Z (x_Z)
$s_0, t_0 \in_R \mathbb{Z}_q$		
$com := g^{s_0} h_Z^{t_0}$	$\xrightarrow{com}$	$s_1 \in_R \mathbb{Z}_q$
$(a_0, b_0) := (a_1 g^{s_0}, b_1 m^{s_0})$	$\xleftarrow{a_1,b_1}$	$(a_1, b_1) := (g^{s_1}, m^{s_1})$
	$\xrightarrow{s_0,t_0}$	$com \overset{?}{=} g^{s_0} h_Z^{t_0}$
		$(a_0, b_0) := (a_1 g^{s_0}, b_1 m^{s_0})$

Figure B.2: Joint randomness selection

The credential issuing protocol in Chaum-Pedersen's wallet, uses a "blinded" version of the signature scheme of Figure B.1. At the end of the signature issuing protocol, the issuer only learns that it has issued a credential to the wallet (C, T), and nothing more. In particular the issuer does not learn any information about the issued credential (i.e., pseudonym h and signature $\sigma_{CA}(h)$). The details of the issuing protocol are given in Figure B.3, on page 170.

B.2.2 Credential showing

A user can show a valid credential (1) by revealing a credential's public key (the pseudonym) along with an issuer's signature on it, and (2) by proving knowledge of the corresponding secret key.

In the Chaum-Pedersen credential system, personal information about the user (wallet owner) is stored in a database DB inside his wallet. In order to authenticate to a verifier (e.g., service provider), the user might be required to reveal some of the information stored in his wallet's DB. Let Q be the verifier's query, and let DB$[Q]$ denote the answer to that query. To convince the verifier that the answer to the query is correct, the user-controlled computing module C provides a signature from the trusted tamper-resistant wallet observer T on DB$[Q]$. More precisely, the pair (C, T) chooses a fresh wallet credential $\{h, \sigma_{CA}(h)\}$. The wallet observer T, then produces a signed proof of knowledge of the secret underlying the newly-chosen wallet pseudonym h, on the answer DB$[Q]$. We denote the signed proof by $\sigma_h(\text{DB}[Q]) = PK\{(\epsilon) : h = h_0^{\epsilon}\}(\text{DB}[Q])$. The tuple $\{h, \sigma_{CA}(h), \text{DB}[Q], \sigma_h(\text{DB}[Q])\}$ is then returned to the verifier. Notice that the wallet observer T uses a fresh pseudonym h to produce the signature, instead of the wallet's long-term public key h_T. This makes the answers to different queries unlinkable to the wallet's identity. Figure B.4 summarizes the showing protocol.

[1] This will represent the public key of the new credential.

Wallet observer T (x_T, h_T)		User's computer C (h_T)		Issuer Z (x_Z, h_Z)
		Establishing a new wallet pseudonym h[1]		
		$\beta_0, y_0 \in_R \mathbb{Z}_q^*$	$\xleftrightarrow{\text{exchange } h_T, h_Z}$	
$y_1 \in_R \mathbb{Z}_q^*$	$\xleftarrow{com_0}$	$com_0 := g^{y_0} h_T^{\beta_0}$		
$h_0 := g^{y_1}$	$\xrightarrow{h_0}$			
$com_0 \overset{?}{=} g^{y_0} h_T^{\beta_0}$	$\xleftarrow{y_0, \beta_0}$	$h := h_0^{y_0}$		
$x := y_0 y_1,\ h := h_0^{y_0}$				
		Issuing a blind signature on wallet pseudonym h		
$z := h_Z^x$	$\xrightarrow{z}$	$t \in_R \mathbb{Z}_q^*$		
		$h_1 := h^t$	$\xrightarrow{h_1}$	
		$s_0, t_0 \in_R \mathbb{Z}_q$		
		$\alpha := g^{s_0} h_Z^{t_0}$	$\xrightarrow{\alpha}$	$s_1 \in_R \mathbb{Z}_q$
		$(a_0, b_0) := (a_1 g^{s_0}, b_1 h_1^{s_0})$	$\xleftarrow{a_1, b_1}$	$(a_1, b_1) := (g^{s_1}, h_1^{s_1})$
			$\xrightarrow{s_0, t_0}$	$\alpha \overset{?}{=} g^{s_0} h_Z^{t_0}$
				$(a_0, b_0) := (a_1 g^{s_0}, b_1 h_1^{s_0})$
		$z \overset{?}{=} z_0^{1/t}$	$\xleftarrow{z_0}$	$z_0 := h_1^{xz}$
	$\xleftarrow{(a_0, b_0, u, v, t)}$	$(u, v) \in_R \mathbb{Z}_q^* \times \mathbb{Z}_q$		
$a := (a_0 g^v)^u$		$a := (a_0 g^v)^u$		
$b := (b_0^{1/t} h^v)^u$		$b := (b_0^{1/t} h^v)^u$		
$c := H(h, z, a, b)$		$c := H(h, z, a, b)$		
$c_0 := c/u,\ c_{0T} := c_0^{x_T}$		$c_0 := c/u$		
		$s_{0T}, t_{0T} \in_R \mathbb{Z}_q$		
$s_{1T} \in_R \mathbb{Z}_q$	$\xleftarrow{\alpha_T}$	$\alpha_T := g^{s_{0T}} h_T^{t_{0T}}$		
$(a_{1T}, b_{1T}) := (g^{s_{1T}}, c_0^{s_{1T}})$	$\xrightarrow{a_{1T}, b_{1T}}$	$(a_T, b_T) := (a_{1T} g^{s_{0T}}, b_{1T} c_0^{s_{0T}})$		
$\alpha_T \overset{?}{=} g^{s_{0T}} h_T^{t_{0T}}$	$\xleftarrow{s_{0T}, t_{0T}}$			
$(a_T, b_T) := (a_{1T} g^{s_{0T}}, b_{1T} c_0^{s_{0T}})$				
$r_T := x_T \times H(c_0, c_{0T}, a_T, b_T) + s_{0T} + s_{1T}$				
$\sigma_T(c_0) := (c_{0T}, a_T, b_T, r_T)$	$\xrightarrow{\sigma_T(c_0)}$	$g^{r_T} \overset{?}{=} a_T h_T^{H(c_0, c_{0T}, a_T, b_T)}$		
		$c_0^{r_T} \overset{?}{=} b_T c_{0T}^{H(c_0, c_{0T}, a_T, b_T)}$	$\xrightarrow{c_0, \sigma_T(c_0)}$	
				$g^{r_T} \overset{?}{=} a_T h_T^{H(c_0, c_{0T}, a_T, b_T)}$
				$c_0^{r_T} \overset{?}{=} b_T c_{0T}^{H(c_0, c_{0T}, a_T, b_T)}$
		$g^{r_0} \overset{?}{=} a_0 h_Z^{c_0}$	$\xleftarrow{r_0}$	$r_0 := s_0 + s_1 + c_0 x_Z$
	$\xleftarrow{r_0}$	$h_1^{r_0} \overset{?}{=} b_0 z_0^{c_0}$		
$r := (r_0 + v)u$		$r := (r_0 + v)u$		
$\sigma_Z(h) := (z, r, c)$		$\sigma_Z(h) := (z, r, c)$		

Figure B.3: Chaum-Pedersen's Wallet-with-Observer Credential Issuing protocol

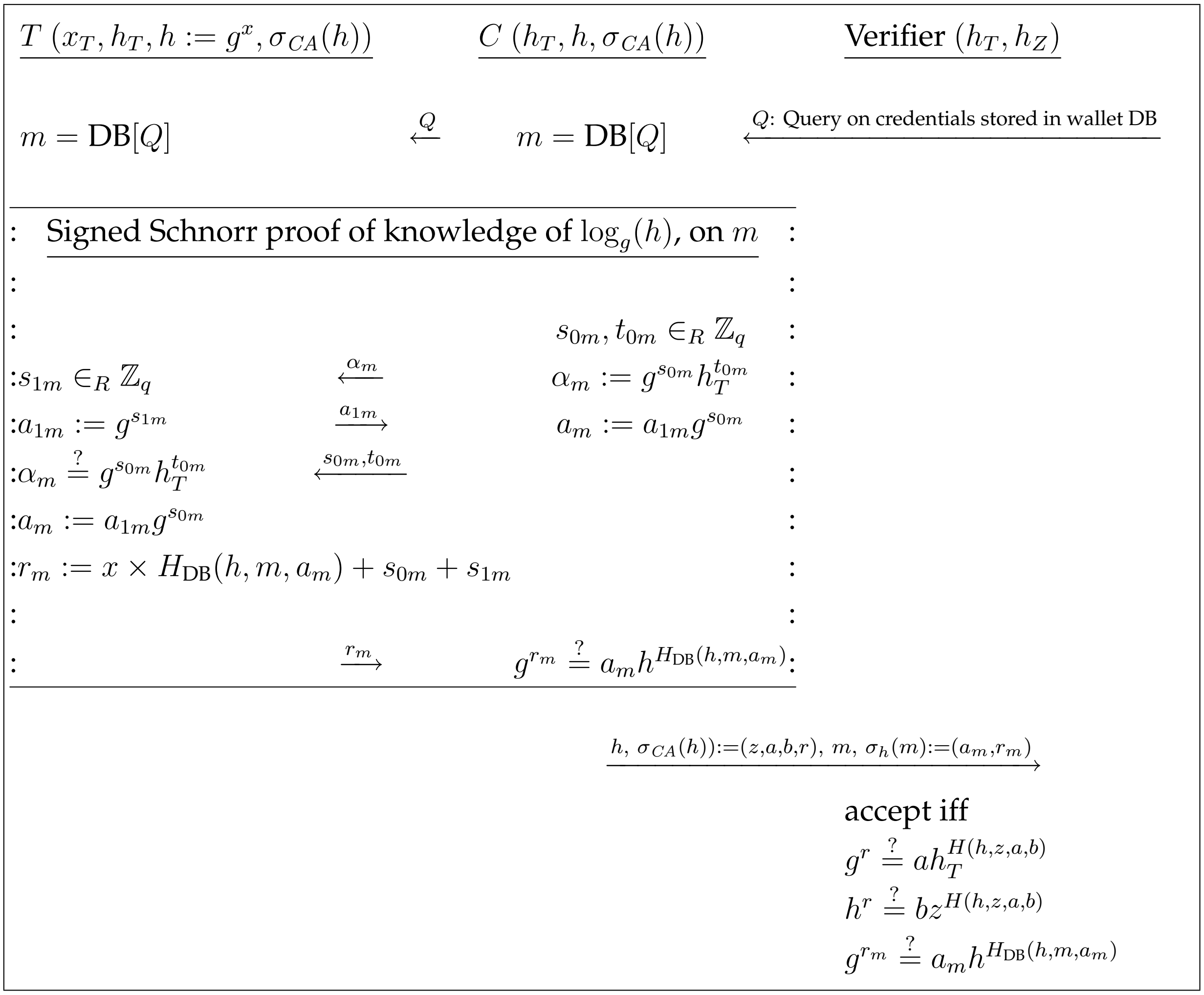
$T\ (x_T, h_T, h := g^x, \sigma_{CA}(h))$
$C\ (h_T, h, \sigma_{CA}(h))$
Verifier (h_T, h_Z)
$m = \mathrm{DB}[Q]$
$\xleftarrow{Q}$
$m = \mathrm{DB}[Q]$
Q: Query on credentials stored in wallet DB
Signed Schnorr proof of knowledge of $\log_g(h)$, on m
$s_{0m}, t_{0m} \in_R \mathbb{Z}_q$
$s_{1m} \in_R \mathbb{Z}_q$
$\xleftarrow{\alpha_m}$
$\alpha_m := g^{s_{0m}} h_T^{t_{0m}}$
$a_{1m} := g^{s_{1m}}$
$\xrightarrow{a_{1m}}$
$a_m := a_{1m} g^{s_{0m}}$
$\alpha_m \stackrel{?}{=} g^{s_{0m}} h_T^{t_{0m}}$
$\xleftarrow{s_{0m}, t_{0m}}$
$a_m := a_{1m} g^{s_{0m}}$
$r_m := x \times H_{\mathrm{DB}}(h, m, a_m) + s_{0m} + s_{1m}$
$\xrightarrow{r_m}$
$g^{r_m} \stackrel{?}{=} a_m h^{H_{\mathrm{DB}}(h,m,a_m)}$
$\xrightarrow{h,\ \sigma_{CA}(h)):=(z,a,b,r),\ m,\ \sigma_h(m):=(a_m,r_m)}$
accept iff
$g^r \stackrel{?}{=} a h_T^{H(h,z,a,b)}$
$h^r \stackrel{?}{=} b z^{H(h,z,a,b)}$
$g^{r_m} \stackrel{?}{=} a_m h^{H_{\mathrm{DB}}(h,m,a_m)}$

Figure B.4: Chaum-Pedersen's Wallet-with-Observer Credential Showing protocol

B.2.3 Summary of security and privacy properties

In this section we state the main security and privacy properties achieved by the issuing and showing protocols of the Chaum-Pedersen credential system.

Credential issuing phase

- Blinding capabilities: If the user-controlled computing module C follows the issuing protocol of Figure B.3, then the issuer does not learn any information about the wallet's pseudonym h or its signature $\sigma_{CA}(h)$. This holds even if the wallet's observer T and the signing organization have unlimited computing power. It is assumed however that the content of the wallet's observer is never disclosed, even after the wallet is expired and returned to the issuer.
- Unforgeability: Assuming the discrete logarithm assumption is hard, and the hash function H collision-resistant. A polynomially bounded user-controlled computing module $\tilde{C}$ cannot cheat the wallet's observer into validating a public key for which $\tilde{C}$ knows the corresponding secret key. This requirement ensures that all subsequent signatures made by the wallet, using the validated public key, are originating from (and hence approved by) the wallet's observer T.

Credential showing phase

- Selective disclosure: Does not apply to the Chaum-Pedersen's signatures since they do not have any embedded attributes. This feature can be added however using Brands techniques as we shall see in the next section.
- Multi-show unlinkability: If the same validated public key h is used more than once by a wallet, the different signatures made by the wallet are linked to each other, but not to the identity of the wallet (represented by the long-term pub-

lic key h_T). The credentials of [CP92] can therefore be considered as one-show credentials.

- Security against false proofs: Assuming the one-way function H collision-resistant, a user cannot prove a false predicate about his credential's secret with non-negligeable probability.
- Impersonation resistance: Assuming the discrete logarithm problem is hard, a polynomially bounded attacker, including the user-controlled module $\tilde{C}$, cannot with non-negligible probability forge signatures on behalf of the wallet's observer T.

Transcript depositing phase

- Untraceability: If the wallet (C, T) follows the showing protocol of Figure B.4, then no information about the wallet's identity is revealed to an all powerful verifier. The same assumption that the content of the wallet observer is never disclosed, applies here as well.
- Multi-show unlinkability: multiple showings of a token obtained through a Chaum-Pedersen signature protocol, are linkable to each other.
- Limited-show capabilities: The user in the Chaum-Pedersen credential system [CP92], can withdraw an infinite number of new credentials, as long as it has legally obtained the first one. As such, the Chaum-Pedersen credential system is not considered to have limited-show capabilities.
- Framing resistance: Similar to Chaum's basic blind signatures, the Chaum-Pedersen credential system [CP92], does not offer a mechanism to ensure non-deniability. In fact, the tokens (i.e., credentials) do not contain any identity attribute that binds them to any particular user or wallet. As a result, assuming the wallet

is tamper-proof, no one can claim that a token has or has not been shown by a particular user.

B.3 Protocols for the DL-based Brands credentials (format I and II)

B.3.1 Credential issuing protocol I

To obtain a credential, a user first convinces the issuer that he fulfills a set of application-specific requirements necessary to receive that credential. The issuer then encodes a pre-defined number ℓ of user attributes in the credential. It is also possible for the user to encode a subset of the attributes which will remain hidden from the certification authority. The user may be required, instead, to prove a certain property of this subset of attributes. Let $x_1, \cdots, x_\ell$ denote the attributes to be encoded. The credential's public key is then computed as $h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^\alpha$, where α is a secret blinding factor randomly chosen by the user in $\mathbb{Z}_q^*$. The issuer's digital signature on the credential is a triplet $(c'_0, r'_0, z') \in \mathbb{Z}_q^2 \times G_q$, satisfying the relation $c'_0 = H(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0} z'^{-c'_0})$. At the end of the issuing protocol, the issuer knows neither h nor the signature (c'_0, r'_0, z'). Figure B.5 shows how a user may obtain a credential with hidden attributes from the issuer. For $\ell' < \ell$, let $x_1, \cdots, x_{\ell'}$ denote the attributes chosen by the user and hidden from the issuer.

In step 1 of the issuing protocol, the user sends a signed proof to the issuer, where he shows that he knows a representation of the commitment com, and that this representation satisfies an agreed-upon predicate $\mathcal{P}$. The class of predicates $\mathcal{P}$ considered by Brands are of the form $\mathcal{P} = \underset{i}{\text{OR}} ((\underset{j}{\text{AND}}\ F_{ij})\ \text{AND}\ \neg E_i)$, where the E_i's and F_{ij}'s are linear formulas of the form $\mu_1 x_1 + \cdots + \mu_\ell x_\ell = \nu$, for constants$\nu$ and μ_i. In case

User	Public Info	Issuer
	$(g_i)_{0\leq i\leq \ell}, h_0, (g_i^{x_0})_{0\leq i\leq \ell}, h_0^{x_0}, q, G_q, H$	
	$\xleftarrow{\quad m \quad}$	$m = nonce\Vert\ldots$
$\beta \in_R \mathbb{Z}_q$		
$com := g_1^{x_1} \cdots g_{\ell'}^{x_{\ell'}} g_\ell^{\beta}$	$\xrightarrow[\substack{SPK\{(\varepsilon_1,\cdots,\varepsilon_{\ell'},\gamma):com=g_1^{\varepsilon_1}\cdots g_{\ell'}^{\varepsilon_{\ell'}}\, g_\ell^{\gamma}\wedge \\ \mathcal{P}(\varepsilon_1,\cdots,\varepsilon_{\ell'},\gamma)=\text{TRUE}\}(m)}]{\quad com \quad}$	
		$w_0 \in_R \mathbb{Z}_q$
$x_\ell := \beta$,		$a_0 := g_0^{w_0}$
$\alpha_1 \in_R \mathbb{Z}_q^*$,		$b_0 := (com\ .\ g_{\ell'+1}^{x_{\ell'+1}} \cdots g_{\ell-1}^{x_{\ell-1}}\ .\ h_0)^{w_0}$
	$\xleftarrow{\quad a_0, b_0 \quad}$	
$\alpha_2, \alpha_3 \in_R \mathbb{Z}_q$,		
$h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{\alpha_1}$		
$z := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{x_0}$		
$z' := z^{\alpha_1}$		
$a_0' := h_0^{\alpha_2} g_0^{\alpha_3} a_0$		
$b_0' := (z')^{\alpha_2} h^{\alpha_3} b_0^{\alpha_1}$		
$c_0' := H(h, z', a_0', b_0')$		
$c_0 := c_0' + \alpha_2 \mod q$	$\xrightarrow{\quad c_0 \quad}$	
	$\xleftarrow{\quad r_0 \quad}$	$r_0 := c_0 x_0 + w_0 \mod q$
$r_0' := r_0 + \alpha_3 \mod q$		
Accept iff $a_0' b_0' = (g_0 h)^{r_0'} (h_0 z')^{-c_0'}$		
Store h and $\sigma_{CA}(h) := (z', r_0', c_0')$		

Figure B.5: Brands Credential Issuing protocol I (with hidden attributes)

the i^{th} term in $\mathcal{P}$ does not contain a negation, the formula E_i can be just set off to FALSE, in which case it cancels out. Predicates with terms containing more than one negation are not supported. We illustrate the general idea with an example in Figure B.6, where we give a signed proof for the predicate $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$ $(*)$. The second term in $\mathcal{P}$ implies that there exists an $\varepsilon \neq 0$, such that $\varepsilon = 8 - x_1 - 2x_2 - 3x_3 - 5x_4$. It can then be easily derived from $(*)$ that $x_1 = (\varepsilon - 2) - x_3 - 6x_4$. If we replace this in a commitment of the form $h := g_1^{x_1} \cdots g_\ell^{x_\ell} h_0^\alpha$, we get :

$$\begin{aligned} h &= g_1^{x_1} \cdots g_\ell^{x_\ell} h_0^\alpha \\ &= g_1^{(\varepsilon-2-x_3-6x_4)} g_2^{x_2} \cdots g_\ell^{x_\ell} h_0^\alpha \\ \Rightarrow \quad hg_1^2 &= g_1^\varepsilon \, g_1^{(-x_3-6x_4)} g_2^{x_2} \cdots g_\ell^{x_\ell} h_0^\alpha \\ \Rightarrow \quad g_1^\varepsilon &= (hg_1^2)\, g_2^{-x_2} (g_1^{-1} g_3)^{-x_3} (g_1^{-6} g_4)^{-x_4} g_5^{-x_5} \cdots g_\ell^{-x_\ell} h_0^{-\alpha} \\ \Rightarrow \quad g_1 &= (hg_1^2)^{1/\varepsilon}\, g_2^{-x_2/\varepsilon} (g_1^{-1} g_3)^{-x_3/\varepsilon} (g_1^{-6} g_4)^{-x_4/\varepsilon} g_5^{-x_5/\varepsilon} \cdots g_\ell^{-x_\ell/\varepsilon} h_0^{-\alpha/\varepsilon} \end{aligned}$$

To prove predicate $\mathcal{P}$, it is therefore sufficient for the prover to produce a proof of knowledge of a representation of g_1 with respect to the basis $((hg^2), g_2, (g_1^{-1} g_3), (g_1^{-6} g_4), g_5, \cdots, g_\ell, h_0)$. The details are given in Figure B.6. Signed proofs for the more general type of predicates with OR operators use a standard technique. For more details on this technique, interested readers are invited to see [Bra00, Section 3.5]. Notice that the credential public key h in Figure B.5 has the following slightly different form:

$(h = g_1^{\alpha x_1} \cdots g_\ell^{\alpha x_\ell} h_0^{\alpha})$, for $\alpha \neq 0$. Let z_0 denote (α) and z_i denote (αx_i) for $1 \leq i \leq \ell$, then to prove a linear predicate of the form $\sum_i \mu_i x_i = \beta$, it is sufficient to prove that $\sum_i \mu_i z_i = \beta z_0$ AND $z_0 \neq 0$. The first term can be proved as indicated on Figure B.6, while the latter can be easily verified by checking that $h \neq 1$.

User	Public Info	Verifier
	$(g_i)_{0\leq i\leq \ell}, h_0, (g_i^{x_0})_{0\leq i\leq \ell}, h_0^{x_0}, q, G_q, H$	
$w_1, \cdots, w_{\ell+1} \in_R \mathbb{Z}_q$	$\xleftarrow{\quad m \quad}$	$m := nonce\|\|..$
$a := (hg_1^2)^{w_1} g_2^{w_2}$		
$\cdot\ (g_1^{-1}g_3)^{w_3}(g_1^{-6}g_4)^{w_4}$		
$\cdot\ g_5^{w_5} \cdots g_\ell^{w_\ell} h_0^{w_{\ell+1}}$		
$c := H(h, a, \mathcal{P}, m)$		
$\delta := -1/\varepsilon$		
$r_1 := -c\delta + w_1$		
$r_2 := c\delta x_2 + w_2$		
$\vdots$		
$r_\ell := c\delta x_\ell + w_\ell$		
$r_{\ell+1} := c\delta\alpha + w_{\ell+1}$	$\xrightarrow{h,\mathcal{P},(a,r_1,\cdots,r_{\ell+1})}$	$c := H(h, a, \mathcal{P}, m)$
		accept iff
		$g_1^c a = (hg_1^2)^{r_1} g_2^{r_2}$
		$\cdot\ (g_1^{-1}g_3)^{r_3}(g_1^{-6}g_4)^{r_4}$
		$\cdot\ g_5^{r_5} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$

Figure B.6: Signed proof of knowledge: $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$

B.3.2 Credential issuing protocol II

The issuing protocol presented in this section produces a secret-key certificate, as opposed to the public-key certificate issued in the protocol of the previous section.

Secret-key certificates [Bra95c] consist of a public key (corresponding to a secret key), and an issuer-supplied signature on it. The main difference between secret-key and public-key certificates is that secret-key certificates are simulatable by anyone according to a distribution indistinguishable from that generated by the real issuer, while public-key certificates are not. The validity of the secret-key certificate can be verified only after the user proves that he knows the secret key corresponding to the certificates public key. Figure B.7 summarizes the new issuing protocol.

User	Public Info	Issuer $(x_0, y_1, \cdots, y_\ell)$
	$(g_i)_{0 \le i \le \ell} := g_0^{y_i}, h_0 := g_0^{x_0}, q, G_q, H$	
$h' := (hh_0)^{\alpha_1}$	$\xleftrightarrow{x_1, \cdots, x_\ell}$	
		$w_0 \in_R \mathbb{Z}_q$
	$\xleftarrow{a_0}$	$a_0 := g_0^{w_0}$
$\alpha_1 \in_R \mathbb{Z}_q^*$,		
$\alpha_2, \alpha_3 \in_R \mathbb{Z}_q$,		
$h := g_1^{x_1} \cdots g_\ell^{x_\ell}$		
$c_0' := H(h', g_0^{\alpha_2}(hh_0)^{\alpha_3} a_0)$		
$c_0 := c_0' - \alpha_2 \mod q$	$\xrightarrow{c_0}$	
	$\xleftarrow{r_0}$	$r_0 := (w_0 - c_0)/(x_0 + \sum_{i=1}^{\ell} x_i y_i) \mod q$
$g_0^{c_0}(hh_0)^{r_0} \stackrel{?}{=} a_0$		
$r_0' := (r_0 + \alpha_3)/\alpha_1 \mod q$		
Store h' and $\sigma_{CA}(h') := (r_0', c_0')$		

Figure B.7: Brands Credential Issuing protocol II (with all attributes known to the issuer)

B.3.3 Credential showing

The following describes the showing protocol for credentials obtained through issuing protocol I. Credentials obtained through issuing protocol II can be shown in a very similar manner, as indicated in [Bra00, Section 5.2.2].

User U can show his credential to obtain goods and services, without the verifier being able to (1) learn information about the encoded attributes beyond what the user willingly discloses, or (2) link the credential to the user's identity even if it colludes with the issuer. In addition, if the special *identity revocation mechanism* is enabled, then the user's attributes can be uncovered if the credential is shown more than once. To show his credential, user U first reveals the credential's public key h along with an issuer's signature $\sigma_{CA} := (z, c'_0, r'_0)$. The verifier checks the validity of the signature by verifying if the relation $c'_0 \overset{?}{=} H(h, z', g_0^{r'_0} h_0^{-c'_0}, h^{r'_0} z'^{-c'_0})$ holds. If so, the user produces a signed proof of knowledge of the credential's secret key on a verifier-chosen challenge m. The signed proof (c.f., Figure B.6) is computed with respect to a predicate $\mathcal{P}$ agreed-upon by the user and the verifier, at the time of the showing. Figure B.8 sketches Brands' basic showing protocol.

As in the example of Figure B.6, the initial witness a depends on the predicate $\mathcal{P}$ to be proved, while the responses $r_1, \cdots, r_{\ell+1}$, depend in addition on $c := H(h, a, \mathcal{P}, m)$, where m is a verifier-chosen challenge. If the user can be forced to use the same initial witness a in two different instances of the showing protocol, then we get two tuples $(c, r_1, \cdots, r_{\ell+1})$ and $(c^*, r_1^*, \cdots, r_{\ell+1}^*)$, such that :

$$g_1^c a = (h g_1^2)^{r_1} g_2^{r_2} (g_1^{-1} g_3)^{r_3} (g_1^{-6} g_4)^{r_4} g_5^{r_5} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$$

User	Public Info	Verifier
	$(g_i)_{0 \leq i \leq \ell}, h_0, (g_i^{x_0})_{0 \leq i \leq \ell}, h_0^{x_0}, q, G_q, H$	
$a := ...$	$\xleftarrow{m, \mathcal{P}}$	$m := nonce\|\|...$
$c := H(h, a, \mathcal{P}, m)$		
$r_1 := ...$		
$\vdots$		
$r_{\ell+1} := ...$	$\xrightarrow[SPK\{(\varepsilon_1, \cdots, \varepsilon_{\ell+1}): h = g_1^{\varepsilon_1} \cdots g_\ell^{\varepsilon_\ell} h_0^{\varepsilon_{\ell+1}} \wedge \mathcal{P}(\varepsilon_1, \cdots, \varepsilon_\ell) = \text{TRUE}\}(m)]{h,\ \sigma_{CA}(h) := (z, c_0', r_0')}$	
		accept iff $\sigma_{CA}(h)$ and the signed proof are valid

Figure B.8: Brands Basic Credential Showing protocol

and

$$g_1^{c^*} a = (h g_1^2)^{r_1^*} g_2^{r_2^*} (g_1^{-1} g_3)^{r_3^*} (g_1^{-6} g_4)^{r_4^*} g_5^{r_5^*} \cdots g_\ell^{r_\ell^*} h_0^{r_{\ell+1}^*}$$

which yields

$$h^{r_1^* - r_1} = g_1^{(c^* - c) - 2(r_1^* - r_1) + (r_3^* - r_3) + 6(r_4^* - r_4)} g_2^{r_2 - r_2^*} g_3^{r_3 - r_3^*} \cdots g_\ell^{r_\ell - r_\ell^*} h_0^{r_{\ell+1} - r_{\ell+1}^*} \tag{B.1}$$

If $r_1^* = r_1$, then $((c^* - c) - 2(r_1^* - r_1) + (r_3^* - r_3) + 6(r_4^* - r_4), r_2 - r_2^*, \cdots, r_{\ell+1} - r_{\ell+1}^*) = 0^{\ell+1}$, otherwise we would obtain a non-trivial representation of 1, which is not possible assuming the discrete logarithm representation problem is hard. Therefore it must be that $r_i^* = r_i$ for all $1 \leq i \leq \ell + 1$. But this cannot be the case, because the challenges c and c^* are different, and the equation above reduces to $g_1^{(c^* - c)} = 1$, which is absurd.

Therefore equation (B.1) yields

$$\begin{aligned} h &= g_1^{\frac{(c^*-c)-2(r_1^*-r_1)+(r_3^*-r_3)+6(r_4^*-r_4)}{(r_1^*-r_1)}} g_2^{(r_2-r_2^*)/(r_1^*-r_1)} g_3^{(r_3-r_3^*)/(r_1^*-r_1)} \cdots g_\ell^{(r_\ell-r_\ell^*)/(r_1^*-r_1)} h_0^{(r_{\ell+1}-r_{\ell+1}^*)/(r_1^*-r_1)} \\ &= g_1^{x_1} g_2^{x_2} g_3^{x_3} \cdots g_\ell^{x_\ell} h_0^{\alpha} \end{aligned}$$

Assuming the the discrete logarithm representation problem is hard, the representation $(x_1, \cdots, x_\ell, \alpha)$ must be the same as

$$\left(\frac{(c^* - c) - 2(r_1^* - r_1) + (r_3^* - r_3) + 6(r_4^* - r_4)}{(r_1^* - r_1)}, \frac{(r_2 - r_2^*)}{(r_1^* - r_1)}, \cdots, \frac{(r_{\ell+1} - r_{\ell+1}^*)}{(r_1^* - r_1)} \right),$$

otherwise user U will be able to produce different sets of attributes for the same credential. In summary, if the user shows his credential more than once using the same initial witness a, then all of the user's attributes encoded in the credential will be revealed. In order to force U to use the same witness a in his signed proofs when showing a given credential, the issuing protocol of Figure B.5 is modified by enabling the *identity revocation mechanism*. The modification is summarized as follows.

Enabling identity revocation. The user's witness a is bound to the credential's signature (c'_0, r'_0, z'), where c'_0 is from now on computed as $c'_0 := H(h, a, z', a'_0, b'_0)$. This means that once he obtains a credential's signature (c'_0, r'_0, z'), the user is forced to use the same witness a in all showings of the credential, unless he breaks the collision-resistance of hash function H. The problem with the latter modification is that, since the witness a depends on the predicate $\mathcal{P}$ to be proved, the user has to know $\mathcal{P}$ at the issuing time. To go around this problem, the user binds his credential to a generic witness a^* at the issuing protocol, and produces correction factors that fit the predicate to be proved at the showing time. Figure B.9 shows the new issuing protocol.

User	Public Info	Issuer
	$(g_i)_{0\leq i\leq\ell}, h_0, (g_i^{x_0})_{0\leq i\leq\ell}, h_0^{x_0}, q, G_q, H$	
	$\xleftarrow{\quad m \quad}$	$m = nonce{\vert}{\vert}...$
$\beta \in_R \mathbb{Z}_q$		
$com := g_1^{x_1} \cdots g_{\ell'}^{x_{\ell'}} g_\ell^{\beta}$	$\xrightarrow[\substack{SPK\{(\varepsilon_1,\cdots,\varepsilon_{\ell'},\gamma):com=g_1^{\varepsilon_1}\cdots g_{\ell'}^{\varepsilon_{\ell'}} g_\ell^{\gamma}\wedge \\ \mathcal{P}(\varepsilon_1,\cdots,\varepsilon_{\ell'},\gamma)=\text{TRUE}\}(m)}]{com}$	
		$w_0 \in_R \mathbb{Z}_q$
$x_\ell := \beta$,		$a_0 := g_0^{w_0}$
$\alpha_1 \in_R \mathbb{Z}_q^*$,		$b_0 := (com\ .\ g_{\ell'+1}^{x_{\ell'+1}} \cdots g_{\ell-1}^{x_{\ell-1}}\ .\ h_0)^{w_0}$
$\alpha_2, \alpha_3 \in_R \mathbb{Z}_q$,	$\xleftarrow{\quad a_0,b_0 \quad}$	
$w_1, \cdots, w_{\ell+1} \in_R \mathbb{Z}_q$		
$h := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{\alpha_1}$		
$a^* := (hg_1)^{w_1} g_2^{w_2} \cdots g_{\ell+1}^{w_{\ell+1}}$		
$z := (g_1^{x_1} \cdots g_\ell^{x_\ell} h_0)^{x_0}$		
$z' := z^{\alpha_1}$		
$a_0' := h_0^{\alpha_2} g_0^{\alpha_3} a_0$		
$b_0' := (z')^{\alpha_2} h^{\alpha_3} b_0^{\alpha_1}$		
$c_0' := H(h, a^*, z', a_0', b_0')$		
$c_0 := c_0' + \alpha_2 \mod q$	$\xrightarrow{\quad c_0 \quad}$	
	$\xleftarrow{\quad r_0 \quad}$	$r_0 := c_0 x_0 + w_0 \mod q$
$r_0' := r_0 + \alpha_3 \mod q$		
accept iff		
$a_0' b_0' = (g_0 h)^{r_0'} (h_0 z')^{-c_0'}$		

Figure B.9: Brands One-Show Credential Issuing protocol I – (with hidden attributes)

To accommodate the above changes brought to the issuing protocol, the user produces a set of correction factors that will enable the verifier in the showing protocol to compute the witness a necessary to prove predicate $\mathcal{P}$. For example, in order to prove the predicate $\mathcal{P}$ of Figure B.6, the user needs a single correction factor $e_1 := w_1 - w_3 - 6w_4$, to retrieve from generic a^* the right witness $a := a^*.g_1^{e_1}$. The signed proof in the showing protocol is therefore modified as indicated in Figure B.10.

Assume user U shows the same credential twice. Let $\mathcal{P}$ and $\mathcal{P}^*$ be the predicates proved by U in those instances ($\mathcal{P}$ and $\mathcal{P}^*$ might be identical). Let $(a^*, c, e_1, \cdots, e_{\ell+1}, r_1, \cdots, r_{\ell+1})$ and $(a^*, c^*, e_1^*, \cdots, e_{\ell+1}^*, r_1^*, \cdots, r_{\ell+1}^*)$ be the two proof transcripts the verifier ends up with. We have then

$$g_1^c a^* g_1^{e_1} \cdots g_\ell^{e_\ell} h_0^{e_{\ell+1}} = (hg_1^2)^{r_1} g_2^{r_2} (g_1^{-1} g_3)^{r_3} (g_1^{-6} g_4)^{r_4} g_5^{r_5} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$$

and

$$g_1^{c^*} a^* g_1^{e_1^*} \cdots g_\ell^{e_\ell^*} h_0^{e_{\ell+1}^*} = (hg_1^2)^{r_1^*} g_2^{r_2^*} (g_1^{-1} g_3)^{r_3^*} (g_1^{-6} g_4)^{r_4^*} g_5^{r_5^*} \cdots g_\ell^{r_\ell^*} h_0^{r_{\ell+1}^*}$$

which yields

$$\begin{aligned} h^{r_1^*-r_1} &= g_1^{(c^*-c)+(e_1^*-e_1)-2(r_1^*-r_1)+(r_3^*-r_3)+6(r_4^*-r_4)} \\ &\qquad \cdot g_2^{(e_2^*-e_2)-(r_2^*-r_2)} \cdots g_\ell^{(e_\ell^*-e_\ell)-(r_\ell^*-r_\ell)} h_0^{(e_{\ell+1}^*-e_{\ell+1})-(r_{\ell+1}^*-r_{\ell+1})} \end{aligned} \tag{B.2}$$

If $r_1^* = r_1$ then all the exponents on the right-hand side of equation (B.2) will be zeros, otherwise the user would be able to compute non-trivial representations of 1. In particular, this means that given a predicate $\mathcal{P}^*$, and its associated correction factors $(e_1^*, \cdots, e_{\ell+1}^*)$, the user can choose a valid c^* such that $(c^* - c) + (e_1^* - e_1) +$

$(e_3^* - e_3) + 6(e_4^* - e_4) = 0$, regardless of challenge m^*, which is impossible if H is a collision-resistant one-way hash function (recall that to be valid, c^* should satisfy the relation $c^* = H(h, a^*, e_1^*, \cdots, e_{\ell+1}^*, \mathcal{P}^*, m^*)$). Therefore, it must be that $r_1 \neq r_1^*$, and h can be derived as

$$h = g_1^{\frac{(c^*-c)+(e_1^*-e_1)-2(r_1^*-r_1)+(r_3^*-r_3)+6(r_4^*-r_4)}{r_1^*-r_1}} g_2^{\frac{(e_2^*-e_2)-(r_2^*-r_2)}{r_1^*-r_1}} \cdots g_\ell^{\frac{(e_\ell^*-e_\ell)-(r_\ell^*-r_\ell)}{r_1^*-r_1}} h_0^{\frac{(e_{\ell+1}^*-e_{\ell+1})-(r_{\ell+1}^*-r_{\ell+1})}{r_1^*-r_1}}$$

As a result, the user's attributes can be recovered as required from two showing transcripts.

User	Public Info	Verifier
	$(g_i)_{0 \leq i \leq \ell}, h_0, (g_i^{x_0})_{0 \leq i \leq \ell}, h_0^{x_0}, q, G_q, H$	
$e_1 := w_1 - w_3 - 6w_4$	$\xleftarrow{\quad m \quad}$	$m := nonce\|\|..$
$a := a^*.g_1^{e_1}$		
$c := H(h, a^*, e_1, \mathcal{P}, m)$		
$\delta := -1/\varepsilon$		
$r_1 := -c\delta + w_1$		
$r_2 := c\delta x_2 + w_2$		
$\vdots$		
$r_\ell := c\delta x_\ell + w_\ell$		
$r_{\ell+1} := c\delta\alpha + w_{\ell+1}$	$\xrightarrow{h, \mathcal{P}, e_1, (a^*, r_1, \cdots, r_{\ell+1})}$	$a := a^*.g_1^{e_1}$
		$c := H(h, a^*, e_1, \mathcal{P}, m)$
		accept iff
		$g_1^c a = (hg_1^2)^{r_1} g_2^{r_2}$
		$\cdot\ (g_1^{-1}g_3)^{r_3}(g_1^{-6}g_4)^{r_4}$
		$.\ g_5^{r_5} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$

Figure B.10: Signed proof with generic witness: $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$

Multiple showing and credential replication. The previous paragraph described how Brands credentials operate in the one-show mode. Credentials suitable for multiple showings can be built as follows. Users need to obtain multiple versions of the same credential, each with a different witness a^*. Each of these versions is used only once, and the different versions are unlinkable to each others. When these versions are issued in batch, significant savings in computation and communication can be made.

Credential updating. It happens often that users need to update attributes encoded in credentials they have not spent (shown) yet. Let U be a user with a fresh (unspent) credential $\{h, (c_0, r_0)\}$, where h encodes a set of attributes $x_1, \cdots, x_\ell$. It should be possible for U to obtain an updated credential version $\{h_1, (c_{01}, r_{01})\}$, where h_1 encodes attributes $x_1 + u_1, \cdots, x_\ell + u_\ell$, for some correction factors $u_1, \cdots, u_\ell$. The updating and recertification procedure should be possible without the issuer learning the current attributes $x_1, \cdots, x_\ell$, or any information about the representation of the current credential's public key h. Full details of the method used for the updating and recertification of unspent credentials is provided in [Bra00, Section 5.2.1].

Selective depositing or enhanced verifier privacy vis-a-vis transcript collectors. Verifiers are often required to deposit credential showing transcripts to issuers or more generally to transcript collectors. These transcripts may serve for a variety of purposes ranging from fraud detection to general book-keeping. Sometimes though, for business or legal reasons, verifiers may not want to disclose to the transcript collectors all the information the user revealed to them (the verifiers) during the showing protocol. Let $\mathcal{P}$ denote the predicate a user proves to the verifier in a showing protocol. In [Bra00, Section 5.3], a method is given to allow a verifier to selectively disclose to the issuer a proof transcript of a sub-predicate $\mathcal{P}^*$ extracted from the predicate $\mathcal{P}$, without the issuer being able to learn anything about the hidden parts of the original predicate $\mathcal{P}$. The sub-predicate $\mathcal{P}^*$ should be derived from $\mathcal{P}$ by pruning some of the AND connectives. Selective depositing for predicates that contain NOT operators is not supported. The sub-predicate $\mathcal{P}^*$ has to be chosen by the verifier at the time when the user shows his credential to the verifier. The verifier cannot later deposit a proof transcript for a predicate different from $\mathcal{P}^*$ (not even the original $\mathcal{P}$). The verifier, also, cannot in any case produce a valid proof transcript for a predicate $\mathcal{E}$ that is not derived from $\mathcal{P}$ by pruning AND connectives. Details of the mechanics used to enhance verifiers' privacy (by selective depositing) can be found in [Bra00, Section 5.3].

Integrating tamper-resistant devices. The techniques described above provide mechanisms to detect abuses and identify perpetrators. While these mechanisms represent a strong deterrent against cheating, it may be desirable to deploy additional safeguards to prevent cheating from occurring at the first place. One way to achieve this is by using tamper-resistant hardware. As argued by Brands in [Bra00, Section 6.1], hardware-only solutions are not sufficient to protect either the privacy of the user or the interests of the issuers or verifiers. For instance, an issuer can embed a unique serial number or a predictable pseudo-random generator in a user's smartcard, and use it to trace all of the user's transactions; the card may also work as a Trojan Horse and communicate to the outside world (e.g., the issuer) information about the user's transaction without his knowledge. On the other hand, in the absence of additional (e.g., software-only) safeguards, a user that manages to break the tamper-resistance of the card may be able to use it, lend it, or sell it infinitely many times without ever being traced. Both types of solutions (software and hardware) need to be combined in order to achieve higher levels of security and privacy. In particular it is a desirable feature to have the tamper-resistant smartcard capable of controlling the use of credentials in such a way that the user-controlled computer is not able to show credentials without the assistance of the smartcard. In addition, it is also a desirable feature to have the user-controlled computer able to filter out all traffic flowing between the smartcard and the outside world, and break any channel that may be used by the smartcard or a verifier to communicate covert information.

Basic smartcard integration. In the following, Figure B.11 shows how a showing protocol such as that of Figure B.6 can be extended to work on a combination of smartcard and user-controlled computer. Let $\mathcal{S}$ and $\mathcal{U}$ denote the smartcard and the user-controlled computer, respectively. Without loss of generality, assume for example that the attribute x_5 is generated at random in $\mathbb{Z}_q$, and is known to $\mathcal{S}$ but not to $\mathcal{U}$. Let $h_{\mathcal{S}} := g_5^{x_5}$, and $h_{\mathcal{U}}$ be such that $h = g_1^{x_1} \cdots g_\ell^{x_\ell} h_0^{x_{\ell+1}} = h_{\mathcal{S}} h_{\mathcal{U}}$. To demonstrate the formula of Figure B.6, the pair $< \mathcal{S}, \mathcal{U} >$ now proceed as follows. To compute the initial witness a, $\mathcal{S}$ chooses a random $w_5 \in \mathbb{Z}_q$, and computes $a_{\mathcal{S}} := g_5^{w_5}$ and sends it to $\mathcal{U}$. The user-controlled computer $\mathcal{U}$ generates the rest of the w_i's, and computes $a := a_{\mathcal{S}}^{\delta}.a_{\mathcal{U}}$, where $a_{\mathcal{U}} := (hg_1^2)^{w_1} g_2^{w_2} (g_1^{-1} g_3)^{w_3} (g_1^{-6} g_4)^{w_4} g_6^{w_6} \cdots g_\ell^{w_\ell} h_0^{w_{\ell+1}}$. Notice that $\mathcal{U}$ is able to compute δ on its own because the smartcard's secret x_5 does not appear in the predicate $\mathcal{P}$ to be proved. The user-controlled computer $\mathcal{U}$ then computes c as in Figure B.6, and sends it to $\mathcal{S}$. $\mathcal{S}$ computes the response $r_{\mathcal{S}} := cx_5 + w_5$, and sends it to $\mathcal{U}$. $\mathcal{U}$ then computes $r_5 := r_{\mathcal{S}}\delta$, and the rest of the r_i's as indicated in the original proof of Figure B.6. The different steps are summarized in Figure B.11.

The same proof can be easily adapted to the setting of Figure B.10, where a generic witness a^* is used. This methodology can also be used with the more general form of predicates with OR connectives. A general treatment of these variants can be found in

Observer $\mathcal{S}(x_5)$		User's computer $\mathcal{U}(x_{i\neq 5})$		Verifier
$w_5 \in_R \mathbb{Z}_q$		$w_{(i\in\{1,\cdots,\ell+1\}\setminus\{5\})} \in_R \mathbb{Z}_q$	$\xleftarrow{m}$	$m := nonce\|\|..$
$a_{\mathcal{S}} := g_5^{w_5}$	$\xrightarrow{a_{\mathcal{S}}}$	$a_{\mathcal{U}} := (hg_1^2)^{w_1}(g_1^{-1}g_3)^{w_3}(g_1^{-6}g_4)^{w_4}$		
		$\cdot\ g_2^{w_2}g_6^{w_6}\cdots g_\ell^{w_\ell}h_0^{w_{\ell+1}}$		
		$\delta := -1/\varepsilon$		
		$a := a_{\mathcal{S}}^{\delta}.a_{\mathcal{U}}$		
$r_{\mathcal{S}} := cx_5 + w_5$	$\xleftarrow{c}$	$c := H(h, a, \mathcal{P}, m)$		
	$\xrightarrow{r_{\mathcal{S}}}$	$r_5 := r_{\mathcal{S}}\delta$		
		$r_1 := -c\delta + w_1$		
		$r_i := c\delta x_i + w_i;$		
		$i \in \{2,\cdots,\ell\} \setminus \{5\}$		
		$r_{\ell+1} := c\delta\alpha + w_{\ell+1}$	$\xrightarrow{h,\mathcal{P},(a,r_1,\cdots,r_{\ell+1})}$	
				$c := H(h, a, \mathcal{P}, m)$
				accept iff
				$g_1^c a = (hg_1^2)^{r_1}g_2^{r_2}$
				$\cdot\ (g_1^{-1}g_3)^{r_3}(g_1^{-6}g_4)^{r_4}$
				$\cdot\ g_5^{r_5}\cdots g_\ell^{r_\ell}h_0^{r_{\ell+1}}$

Figure B.11: Smartcard-assisted signed proof of : $\mathcal{P} = (2x_1+2x_2+4x_3+11x_4 = 6)$ AND $(x_1+2x_2+3x_3+5x_4 \neq 8)$

[Bra00, Section 6.3]. As shown in Figure B.11, the only way a user can circumvent the smartcard is by physically damaging it and extracting the secret x_5, but as indicated earlier, the software-only mechanisms built in the showing protocols would kick in and allow the issuer to identify the user in case he shows his credentials more than what is allowed.

Smartcard integration with inflow prevention. In the basic smartcard implementation above, the smartcard receives a challenge c that depends on a verifier-chosen message m. The verifier may try to establish a covert inflow channel with the smartcard through m and c. This inflow channel may be used by the verifier to instruct the smartcard to behave in a certain way not specified in the protocol. To prevent such inflow of data, the basic signed proof of Figure B.11 can be modified as follows. When computing a from $a_{\mathcal{S}}$, the user-controlled computer $\mathcal{U}$ "throws" in a randomization factor $h_{\mathcal{S}}^{\beta\delta}$, where $\beta \in_R \mathbb{Z}_q$. In addition, instead of sending c back to the smartcard, $\mathcal{U}$ sends $c_{\mathcal{S}} := c + \beta$ to the smartcard. The changes are summarized in Figure B.12. Similar mechanisms that may be added to prove other forms of predicates (e.g., with OR connectives) can be found in [Bra00, Section 6.4].

Smartcard integration with both inflow and outflow prevention. In the signed proof of Figure B.12, the verifier sees the response r_5 which is a randomized version of the smartcard response $r_{\mathcal{S}}$. Although, it is hard for the verifier to guess the randomization factor δ, we will assume that he manages to guess with non-negligible probability not only δ, but also β, as well as the smartcard's secret x_5. As a result, the smartcard may open an outflow channel with the verifier and send covert data in w_5. To prevent this outflow, the signed proof of Figure B.12 can be modified as follows. When computing a from $a_{\mathcal{S}}$ and $h_{\mathcal{S}}$, the user-controlled computer $\mathcal{U}$ multiplies in a

Observer $\mathcal{S}(x_5)$		User's computer $\mathcal{U}(x_{i\neq 5})$		Verifier
$w_5 \in_R \mathbb{Z}_q$		$w_{(i\in\{1,\cdots,\ell+1\}\setminus\{5\})}, \beta \in_R \mathbb{Z}_q$	$\xleftarrow{\quad m \quad}$	$m := nonce\|\|..$
$a_{\mathcal{S}} := g_5^{w_5}$	$\xrightarrow{a_{\mathcal{S}}}$	$a_{\mathcal{U}} := (hg_1^2)^{w_1}(g_1^{-1}g_3)^{w_3}(g_1^{-6}g_4)^{w_4}$		
		$\cdot\ g_2^{w_2}g_6^{w_6}\cdots g_\ell^{w_\ell}h_0^{w_{\ell+1}}$		
		$\delta := -1/\varepsilon$		
		$a := a_{\mathcal{S}}^{\delta}h_{\mathcal{S}}^{\beta}.a_{\mathcal{U}}$		
		$c := H(h, a, \mathcal{P}, m)$		
$r_{\mathcal{S}} := c_{\mathcal{S}}x_5 + w_5$	$\xleftarrow{c_{\mathcal{S}}}$	$c_{\mathcal{S}} := c + \beta$		
	$\xrightarrow{r_{\mathcal{S}}}$	$r_5 := r_{\mathcal{S}}\delta$		
		$r_1 := -c\delta + w_1$		
		$r_i := c\delta x_i + w_i;$		
		$i \in \{2, \cdots, \ell\} \setminus \{5\}$		
		$r_{\ell+1} := c\delta\alpha + w_{\ell+1}$	$\xrightarrow{h,\mathcal{P},(a,r_1,\cdots,r_{\ell+1})}$	
				$c := H(h, a, \mathcal{P}, m)$
				accept iff
				$g_1^c a = (hg_1^2)^{r_1}g_2^{r_2}$
				$\cdot\ (g_1^{-1}g_3)^{r_3}(g_1^{-6}g_4)^{r_4}$
				$\cdot\ g_5^{r_5}\cdots g_\ell^{r_\ell}h_0^{r_{\ell+1}}$

Figure B.12: Smartcard-assisted signed proof with inflow prevention : $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$

randomization factor g_5^γ, where $\gamma \in_R \mathbb{Z}_q$. In addition, $\mathcal{U}$ later computes $r_5 := r_\mathcal{S}\delta + \gamma$. The changes are summarized in Figure B.13. Similar mechanisms that may be added to prove other forms of predicates (e.g., with OR connectives) can be found in [Bra00, Section 6.4].

Observer $\mathcal{S}(x_5)$		User's computer $\mathcal{U}(x_{i\neq 5})$		Verifier
$w_5 \in_R \mathbb{Z}_q$		$w_{(i\in\{1,\cdots,\ell+1\}\setminus\{5\})}, \beta, \gamma \in_R \mathbb{Z}_q$	$\xleftarrow{m}$	$m := nonce\|\|..$
$a_\mathcal{S} := g_5^{w_5}$	$\xrightarrow{a_\mathcal{S}}$	$a_\mathcal{U} := (hg_1^2)^{w_1}(g_1^{-1}g_3)^{w_3}(g_1^{-6}g_4)^{w_4}$		
		$\quad \cdot\, g_2^{w_2} g_6^{w_6} \cdots g_\ell^{w_\ell} h_0^{w_{\ell+1}}$		
		$\delta := -1/\varepsilon$		
		$a := a_\mathcal{S}^\delta h_\mathcal{S}^\beta g_5^\gamma . a_\mathcal{U}$		
		$c := H(h, a, \mathcal{P}, m)$		
$r_\mathcal{S} := c_\mathcal{S} x_5 + w_5$	$\xleftarrow{c_\mathcal{S}}$	$c_\mathcal{S} := c + \beta$		
	$\xrightarrow{r_\mathcal{S}}$	$r_5 := r_\mathcal{S}\delta + \gamma$		
		$r_1 := -c\delta + w_1$		
		$r_i := c\delta x_i + w_i;$		
		$\quad i \in \{2, \cdots, \ell\} \setminus \{5\}$		
		$r_{\ell+1} := c\delta\alpha + w_{\ell+1}$	$\xrightarrow{h, \mathcal{P}, (a, r_1, \cdots, r_{\ell+1})}$	
				$c := H(h, a, \mathcal{P}, m)$
				accept iff
				$g_1^c a = (hg_1^2)^{r_1} g_2^{r_2}$
				$\quad \cdot\, (g_1^{-1}g_3)^{r_3}(g_1^{-6}g_4)^{r_4}$
				$\quad \cdot\, g_5^{r_5} \cdots g_\ell^{r_\ell} h_0^{r_{\ell+1}}$

Figure B.13: Smartcard-assisted signed proof with inflow and outflow prevention : $\mathcal{P} = (2x_1 + 2x_2 + 4x_3 + 11x_4 = 6)$ AND $(x_1 + 2x_2 + 3x_3 + 5x_4 \neq 8)$

For an even stronger protection against outflow attacks, it is recommended that smartcards are not returned to the issuer even after their use period has ended. This would stop the issuer from possibly probing the smartcard's log and learning about the user's past transactions. Additional mechanisms to prevent other covert channels are extensively discussed in [Bra00, Sections 6.4 & 6.5].

B.3.4 Summary of security and privacy properties

In this section, we consider all three phases of the credential management cycle: the issuing, showing, and depositing. For each of these phases we state the main security and privacy properties achieved, and indicate their underlying assumptions. Furthermore, we revisit the smartcard-based implementation, and highlight its main security and privacy features. A more exhaustive analysis can be found in [Bra00].

Credential issuing phase

- Blinding capabilities: If the user follows the issuing protocol of Figure B.5, then the issuer does not learn any information about the user's hidden attributes beyond what the user willfully discloses. The issuer's view of the issuing protocol is perfectly indistinguishable from the uniform random distribution. In particular, the issuer does not learn any information about the issued certificate.

- Unforgeability: Assuming the discrete logarithm problem is hard. An attacker cannot issue a forged credential on behalf of the issuer with a non-negligible probability of success. This property holds in the random oracle model for any distribution of the attributes, and requires restrictions on the length of the used hash function.

Credential showing phase

- Selective disclosure: If user U follows the showing protocol of Figure B.9, then a verifier V does not learn any information about U's attributes beyond what U willfully discloses. This holds unconditionally.
- Multi-show unlinkability: The credentials in [Bra00] are one-time show, i.e., showing a credential a second time can be linked to the first showing.
- Security against false proofs: Assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. If user U successfully shows his credential with respect to a predefined predicate $\mathcal{P}$, then with overwhelming probability, (1) U knows the set of attributes underlying the credential, and (2) this set of attributes does satisfy the required predicate $\mathcal{P}$.
- Impersonation resistance: Assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. If user U successfully shows a credential, then with overwhelming probability, U knows the set of attributes that has been initially encoded in the credential, and thus owns the credential.

Transcript depositing phase

- Untraceability: If user U follows the showing protocol of Figure B.9, then the issuer cannot link a showing transcript of that credential to the issuing protocol instance that generated it.
- Multi-show unlinkability: As indicated earlier, Brands' credentials are one-time show. The issuer can link any two showing transcripts of the same credential.
- Limited-show capabilities; assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. If the verifiers follow

the showing protocol of Figure B.9, then a user who shows his credential more than once will be detected with overwhelming probability. If in addition the identity revocation mechanism is enabled, then the identity of the perpetrator will be revealed with overwhelming probability.

Brands does provide however a method to withdraw a k-show credential (for k constant) that is more efficient than trivially withdrawing k separate one-time show credentials. The different showings of the obtained k-show credential are nonetheless linkable.

- Framing resistance: If the discrete logarithm representation problem is hard, and the hash function H is collision-intractable, then an attacker cannot forge with non-negligible probability a valid showing transcript on behalf of user U.
- Selective depositing: Assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. If verifier V follows the showing protocol of [Bra00, Section 5.3], then V can successfully deposit to a third party a showing transcript with a signed proof of predicate $\mathcal{P}'$, only if $\mathcal{P}'$ is obtained from the original predicate $\mathcal{P}$ by pruning AND connectives. In particular, V cannot submit a transcript with a signed proof of predicate $\mathcal{E}$ that is not implied by $\mathcal{P}$. The third party learns nothing about $\mathcal{P}$ beyond what is revealed in $\mathcal{P}'$.

Smartcard-based integration

- Smartcard assistance: Assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. A user U cannot, with non-negligible probability, show a credential without the assistance of the smartcard, unless it physically damages the smartcard and extracts its secrets.

- Abuser identification: Assuming the discrete logarithm representation problem is hard, and the hash function H collision-intractable. A user who shows his one-show credential more than once, can be identified with overwhelming probability, even if the smartcard is tampered with.

- Inflow and outflow channels prevention: Assuming H is a one-way hash function and the smartcard is never returned to the issuer. If a pair of wallet observer and user-controlled computer follow a showing protocol such as that of Figure B.13, then no inflow or outflow of data passes between the smartcard and a verifier. This property holds unconditionally. Other subliminal channels such as the timing, halting, and van Eck channels can be dealt with by other means. An exhaustive treatment of secure smartcard integration can be found in [Bra00, Section 6.3 and 6.5].

Other features such as lending prevention are also very useful in the smartcard (wallet-with-observer) setting. Among the mechanisms that can be used to discourage lending, we note:

1. Encoding sensitive user information in a credential, which need to be disclosed to the borrower in case the user decides to lend his credentials.

2. Including biometric authentication mechanisms in the smartcard to prevent a borrower from using someone else's card even if he knew the underlying secrets

3. Charging a fee for credential issuing would prevent large-scale lenders from being compensated without being detected.

4. Contact or close-range contact-less smartcard technology could also be used to make sure the credential showing is performed by the card holder present in front of the verifier (e.g., a customs officer), and not by a nearby rogue card.

B.4 Protocols for the Strong-RSA-based Camenisch-Lysyanskaya credentials

B.4.1 CL-SRSA credential issuing

In the following we treat the general case, where the issuer supplies the user with a credential containing attributes $x_1, \cdots, x_{\ell'}, x_{\ell'+1}, \cdots, x_\ell$, such that $x_1, \cdots, x_{\ell'}$ are chosen by the user and hidden from the issuer, while the remaining attributes are known to both of them. The user first computes commitments to the hidden attributes and sends them to the issuer, and then runs the protocol of Figure B.14 with the issuer.

B.4.2 CL-SRSA credential showing

Assume user U holds a credential public key encoding attributes $(x_1, \cdots, x_\ell)$, and let (A, e, v) denote the issuer's signature on it. The protocol of Figure B.15, allows U to prove to a verifier V that he has a valid credential from the issuer, while selectively disclosing any portion $(x_1, \cdots, x_{\ell'})$ of the attributes, and hiding the others. The protocol of Figure B.15 can be straightforwardly extended to allow proofs of linear relations on the attributes.

B.4.3 Summary of security and privacy properties

Credential issuing phase

- Blinding capabilities: Under the Strong RSA assumption, and the decisional Diffie-Hellman assumption modulo a safe prime product, the credential issuing protocol of Figure B.14 allows a user U to obtain a valid credential,without the issuer learning any information about the user's secret attributes. The is-

User | Public Info | Issuer

$$n, (R_1, \cdots, R_\ell, S, Z \in QR_n), \ell_m$$

Set $x_1, \cdots, x_{\ell'} \in \pm\{0,1\}^{\ell_m}$

choose $r_1, \cdots, r_{\ell'} \in_R \mathbb{Z}_q$

$c_1 = g^{x_1} h^{r_1}, \cdots, c_{\ell'} = g^{x_{\ell'}} h^{r_{\ell'}}$ $\quad \xleftrightarrow{c_1,\cdots,c_{\ell'},x_{\ell'+1},\cdots,x_\ell}$

choose $v' \in_R \{0,1\}^{\ell_n+\ell_s}$

$C := R_1^{x_1} \cdots R_{\ell'}^{x_{\ell'}} S^{v'} \mod n$ $\quad \xrightarrow[PK\{(\mu_1,\cdots,\mu_{\ell'},\rho_1,\cdots,\rho_{\ell'},\nu):\ c_1=g^{\mu_1}h^{\rho_1}\wedge\cdots\wedge c_{\ell'}=g^{\mu_{\ell'}}h^{\rho_{\ell'}}\wedge C:=R_1^{\mu_1}\cdots R_{\ell'}^{\mu_{\ell'}}S^{\nu} \mod n \wedge \mu_1,\cdots,\mu_{\ell'}\in\{0,1\}^{\ell_m+\ell_c+\ell_s}\}]{C}$

choose $e' \in_R \{0,1\}^{\ell_e}$ s.t.,

$e := 2^{\ell_e+\ell_c+\ell_s+1} + e'$ is prime

choose $v'' \in \mathbb{Z}_e$

$A := (\frac{Z}{CR_{\ell'+1}^{x_{\ell'+1}} \cdots R_\ell^{x_\ell} S^{v''}})^{1/e} \mod n$

$$\xleftarrow[PK\{(\delta):\ A:=\pm(\frac{Z}{CR_{\ell'+1}^{x_{\ell'+1}}\cdots R_\ell^{x_\ell}S^{v''}})^{\delta} \mod n\}]{(A,e,v'')}$$

Accept if

$e > 2^{\ell_e+\ell_c+\ell_s+1}$ is prime

and $A^e C R_{\ell'+1}^{x_{\ell'+1}} \cdots R_\ell^{x_\ell} S^{v''} = Z \mod n$

store $(x_1, \cdots, x_\ell)$ and

$\sigma_{CA}(x_1, \cdots, x_\ell) := (A, e, v := v' + v'')$

Figure B.14: CL-SRSA Credential issuing protocol (with hidden attributes)

User | Public Info | Verifier

$n, (R_1, \cdots, R_\ell, S, Z \in QR_n), \ell_m$

choose $r_1, \cdots, r_{\ell'} \in_R \mathbb{Z}_q$

$c_1 = g^{x_1} h^{r_1}, \cdots, c_{\ell'} = g^{x_{\ell'}} h^{r_{\ell'}}$

choose $r_A \in_R \{0,1\}^{\ell_n + \ell_s}$

$\tilde{A} := AS^{r_A} \mod n$

$$\xrightarrow{\tilde{A}, c_1, \cdots, c_{\ell'}, x_{\ell'+1}, \cdots, x_\ell}$$

$$PK\{(\epsilon, \mu_1, \cdots, \mu_{\ell'}, \rho_1, \cdots, \rho_{\ell'}, \nu) : \ c_1 = g^{\mu_1} h^{\rho_1} \wedge \cdots \wedge c_{\ell'} = g^{\mu_{\ell'}} h^{\rho_{\ell'}} \wedge$$
$$\frac{Z}{\tilde{A}^{2^{\ell_e + \ell_c + \ell_s + 1}} R_{\ell'+1}^{x_{\ell'+1}} \cdots R_\ell^{x_\ell}} = \tilde{A}^\epsilon R_1^{\mu_1} \cdots R_{\ell'}^{\mu_{\ell'}} S^\nu \mod n \wedge$$
$$\epsilon \in \{0,1\}^{\ell_e + \ell_c + \ell_s} \ \wedge \ \mu_1, \cdots, \mu_{\ell'} \in \{0,1\}^{\ell_m + \ell_c + \ell_s}\}$$

Figure B.15: CL-SRSA Credential showing protocol

suer's view of the secrets underlying U's credential is statistically indistinguishable from the uniform random distribution.

- Unforgeability: Under the Strong RSA assumption, an attacker that sees no more than polynomially many valid credential tuples, cannot produce a new valid credential tuple with a non-negligible probability of success.

Credential showing phase

- Selective disclosure: Under the Strong RSA assumption, and the decisional Diffie-Hellman assumption modulo a safe prime product, the credential showing protocol of Figure B.15 does not allow a verifier V to learn any information about user U's credential, beyond its validity status, and what U willfully discloses. The verifier's view of U's credential is statistically indistinguishable from the uniform random distribution.

- Multi-show unlinkability: If user U follows the showing protocol of Figure B.15 properly, then no link can be established between any two executions of the showing protocol that U may engage in, beyond what can be inferred from the information U has willfully revealed. This property holds unconditionally.
- Security against false proofs: Assuming the Strong RSA problem hard. If user U successfully shows his credential with respect to a predefined predicate $\mathcal{P}$, then with overwhelming probability, (1) U knows the set of attributes underlying the credential, and (2) this set of attributes satisfies the required predicate $\mathcal{P}$.
- Impersonation resistance: Assuming the Strong RSA problem is hard. If a verifier V follows the credential showing protocol of Figure B.15, then a user U that deviates from the protocol will be detected with overwhelming probability. In particular a user cannot succeed, with non-negligible probability, in showing a invalid credential or a credential he does not own.

Transcript depositing phase

- Untraceability: Assuming the Strong RSA problem is hard. Given a credential showing transcript from user U, an issuer does not learn any information about the identity of U. For the issuer, the transcript is statistically indistinguishable from the uniform random distribution. In particular the issuer cannot link the shown transcript to the issuing protocol instance that generated the corresponding credential.
- Multi-show Unlinkability: Assuming the Strong RSA problem is hard. Given no more than polynomially many showing transcripts of a multi-show credential, the issuer cannot establish, with non-negligible probability, a link between these transcripts.

- Limited-show capabilities: For one-show credentials, given any two showing transcripts of the same credential, the issuer is able to identify the credential owner with overwhelming probability.
- Framing resistance: Assuming the Strong RSA problem is hard, an attacker cannot forge a showing transcript on behalf of user U with non-negligible probability.

B.5 Protocols for the DL-based Camenisch-Lysyanskaya credentials

B.5.1 CL-DL credential issuing

We consider the case where a user obtains a credential with ℓ attributes $\{m^{(1)}, \cdots, m^{(\ell)}\}$. For $\ell' < \ell$, we assume the subset $\{m^{(1)}, \cdots, m^{(\ell')}\}$ to be known only to the user and hidden from the issuer. In the issuing protocol, the user first commits to each of the hidden attributes $\{m^{(i)}\}_{(1 \leq i \leq \ell')}$, and sends those commitments along with $P = g^v \prod_{i=1}^{\ell} Z_i^{m^{(i)}}$ to the issuer. The user then proves knowledge of a representation of P in basis $(g, Z_1, \cdots, Z_\ell)$, and that this representation is consistent with the content of the commitments. The issuer then issues to the user a credential in the form of a signature on the secret representation of P in basis $(g, Z_1, \cdots, Z_\ell)$. A summary of the issuing protocol is given in Figure B.16.

B.5.2 CL-DL credential showing

The user first generates $\tilde{\sigma}$, a blinded version of his original credential σ. He then proves to the verifier that he knows the secret attributes underlying σ. We consider a

User | Public Info | Issuer

$q, G, \mathsf{G}, g, \mathsf{g}, e, X := g^x, Y := g^y, \{Z_i := g^{z_i}\}_{(1 \le i \le \ell)}$

$v \in_R \mathbb{Z}_q$

$P = g^v \prod_{i=1}^{\ell} Z_i^{m^{(i)}}$

For $1 \le i \le \ell'$

$r_i \in_R \mathbb{Z}_q,\ c_i := g^{m^{(i)}} h^{r^{(i)}}$

$$\xrightarrow[PK\{(\mu^{(0)},\cdots,\mu^{(\ell')},\rho^{(1)},\cdots,\rho^{(\ell')}):\left(\prod_{i=\ell'+1}^{\ell} Z_i^{-m^{(i)}}\right)P=g^{\mu^{(0)}}\prod_{i=1}^{\ell'} Z_i^{\mu^{(i)}}\wedge c_1:=g^{\mu^{(1)}}h^{\rho^{(1)}}\wedge\cdots\wedge c_{\ell'}:=g^{\mu^{(\ell')}}h^{\rho^{(\ell')}}\}]{c_1,\cdots,c_{\ell'},P}$$

if accepted

$r \in_R \mathbb{Z}_q$

$a := g^r,\ b := a^y$,

for $1 \le i \le \ell$

$A_i := a^{z_i},\ B_i := A_i^y$,

$c := a^x P^{rxy}$

$$\xleftarrow{(a,\{A_i\},b,\{B_i\},c)}$$

Accept iff

for $1 \le i \le \ell$

$e(a, Z_i) = e(g, A_i)$ and

$e(A_i, Y) = e(g, B_i)$ and

$e(a, Y) = e(g, b)$ and

$e(g, c) = e(X, a)e(X, b)^v \prod_{i=1}^{\ell} e(X, B_i)^{m^{(i)}}$

Store $\sigma := (a, \{A_i\}, b, \{B_i\}, c, v)$

Figure B.16: CL-DL Credential issuing protocol

scenario where the user may choose to keep the first $\ell' \leq \ell$ attributes hidden from the verifier, and reveal the remaining attributes. To allow for anonymity revocation in case of abuse, user U, in addition, verifiably encrypts the representation underlying P, under the revocation manager's public key. The latter ciphertext, denoted enc, also specifies the condition L under which it may be decrypted, and the user's identity revealed. The encryption scheme used in this case is that of Camenisch and Shoup [CS03]. One should note however that the scheme in [CS03] is based on the strong RSA assumption, a less standard assumption than the DDH assumption required in the basic CL-DL. The details of the extended showing protocol are given in Figure B.17.

B.5.3 Summary of security and privacy properties

Credential issuing phase

- Blinding capabilities: Under the discrete logarithm assumption, the CL-DL credential issuing protocol of Figure B.16 allows a user to obtain a valid credential from the issuer while perfectly hiding his secret key.
- Unforgeability: Under the LRSW assumption, it is infeasible for an probabilistic polynomial-time attacker to forge a new credential with non negligible probability.

Credential showing phase

- Selective disclosure: Assuming the discrete logarithm representation, and the Decisional Diffie-Hellman problems hard. If the verifiable encryption system of [CS98] is secure, then the credential showing protocol of Figure B.17 allows user U to show a valid credential without the verifier V learning any information

User | Public Info | Issuer

$$q, G, \mathsf{G}, g, \mathsf{g}, e, X := g^x, Y := g^y, \{Z_i := g^{z_i}\}_{(1 \le i \le \ell)}$$

Init. Knowledge:

$\{m^{(i)}\}, \sigma := (a, \{A_i\}, b, \{B_i\}, c, v)$

For $1 \le i \le \ell'$

$r_i \in_R \mathbb{Z}_q,\ c_i := g^{m^{(i)}} h^{r^{(i)}}$

$r, r' \in_R \mathbb{Z}_q$

$\tilde{\sigma} := (a^{r'}, \{A_i^{r'}\}, b^{r'}, \{B_i^{r'}\}, c^{rr'})$

$:= (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \tilde{c}^r)$

$:= (\tilde{a}, \{\tilde{A}_i\}, \tilde{b}, \{\tilde{B}_i\}, \hat{c})$

$enc = E_{pk_R}(v, \{m^{(1)}, \cdots, m^{(\ell')}, L\})$ $\xrightarrow{\substack{\tilde{\sigma}:=(\tilde{a},\{\tilde{A}_i\},\tilde{b},\{\tilde{B}_i\},\hat{c}),\ enc,\\ c_1,\cdots,c_{\ell'},m^{(\ell'+1)},\cdots,m^{(\ell)}}}$

$\xleftarrow{m}$ $m := nonce||\cdots$

$\mathsf{V}_x := e(X, \tilde{a})$ | $\mathsf{V}_x := e(X, \tilde{a})$

$\mathsf{V}_{xy} := e(X, \tilde{b})$ | $\mathsf{V}_{xy} := e(X, \tilde{b})$

$\mathsf{V}_{(xy,i)} := e(X, \tilde{B}_i)$ | $\mathsf{V}_{(xy,i)} := e(X, \tilde{B}_i)$

$\mathsf{V}_s := e(g, \hat{c})$ | $\mathsf{V}_s := e(g, \hat{c})$

$$\xrightarrow{\hspace{12cm}}$$

$$SPK\{(\nu, \mu^{(0)}, \cdots, \mu^{(\ell')}, \rho^{(1)}, \cdots, \rho^{(\ell')}) : c_1 = g^{\mu^{(1)}} h^{\rho^{(1)}} \wedge \cdots \wedge c_{\ell'} = g^{\mu^{(\ell')}} h^{\rho^{(\ell')}} \wedge$$
$$\mathsf{V}_x^{-1} \textstyle\prod_{i=\ell'+1}^{\ell} (V_{(xy,i)})^{-m^{(i)}} = (\mathsf{V}_s)^{-\nu} (\mathsf{V}_{xy})^{\mu^{(0)}} \textstyle\prod_{i=1}^{\ell'} (V_{(xy,i)})^{\mu^{(i)}} \wedge$$
$$enc = E_{pk_R}(\mu^{(0)}, \cdots, \mu^{(\ell')}, L)\}(m)$$

Accept iff

SPK is valid and

for $1 \le i \le \ell$

$e(\tilde{a}, Z_i) = e(g, \tilde{A}_i)$ and

$e(\tilde{A}_i, Y) = e(g, \tilde{B}_i)$ and

$e(\tilde{a}, Y) = e(g, \tilde{b})$ and

Figure B.17: CL-DL Credential showing protocol

about the user's credential. V's view of U's credential is statistically[2] indistinguishable from the uniform random distribution.

- Multi-show unlinkability: Assuming the decision Diffie-Hellman and, the discrete logarithm representation problems hard, and assuming the hash function H (of the Cramer-Shoup cryptosystem) collision-resistant. Given no more than polynomially many showing transcripts of the same credential, the verifier cannot establish, with non negligible probability, a link between any pair of such transcripts.

- Security against false proofs: Assuming the discrete logarithm representation problem is hard. If user U successfully shows his credential with respect to a predefined predicate $\mathcal{P}$, then with overwhelming probability, (1) U knows the set of attributes underlying the credential, and (2) this set of attributes satisfies the required predicate $\mathcal{P}$.

- Impersonation resistance (and Unforgeability): Under the decision Diffie-Hellman assumption, the discrete logarithm representation assumption, and assuming the hash function H (of the Cramer-Shoup cryptosystem) collision-resistant. If verifier V follows the credential showing protocol of Figure B.17, then a user U that deviates from the protocol will be detected with overwhelming probability. In particular, a cheating user cannot succeed with a non negligible probability in showing a fake credential or a credential he does not own. Furthermore, a user that does not properly encrypt his secret key under the escrow party's public key, will be caught with overwhelming probability.

[2] This is due to the fact that the cryptosystem of [CS98] is statistically hiding.

Transcript depositing phase and anonymity revocation:

- Untraceability: Assuming the discrete logarithm representation, and decision Diffie-Hellman problems hard. Given a credential showing transcript from user U, an issuer does not learn any information about the identity of U. For the issuer, the transcript is statistically indistinguishable from the uniform random distribution. In particular the issuer cannot link the shown transcript to the issuing protocol instance that generated the credential.
- Multi-show unlinkability: Under the decision Diffie-Hellman assumption, the discrete logarithm representation assumption, and assuming the hash function H (of the Cramer-Shoup cryptosystem) collision-resistant. Given no more than polynomially many showing transcripts of the same credential, the issuer cannot establish, with non negligible probability of success, a link between any pair of such transcripts.
- Limited-show capabilities (and abusers deanonymization): Assuming the decisional Diffie-Hellman problem is hard in groups of large prime order, and the hash function H (of the Cramer-Shoup cryptosystem) collision-resistant. The revocation manager is able to retrieve, with overwhelming probability, the identity of an abuser from a valid showing transcript.
- Framing resistance: Assuming the discrete logarithm representation, and decisional Diffie-Hellman problems hard, and the hash function H (of the Cramer-Shoup cryptosystem) collision-resistant. An attacker cannot forge a showing transcript on behalf of user U with non-negligible probability.

Bibliography

[AdM02] G. Ateniese and B. de Medeiros. Anonymous e-prescriptions. In *WPES*, pages 19–31, 2002.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology – EUROCRYPT'01*, volume 2045 of *LNCS*, pages 119–135. Springer-Verlag, 2001.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *ACM Conference on Computer and Communications Security*, pages 132–145. ACM, 2004.

[BCL04] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In Bruce Christianson, Bruno Crispo, James A. Malcolm, and Michael Roe, editors, *Security Protocols Workshop*, volume 3957 of *Lecture Notes in Computer Science*, pages 20–42. Springer, 2004.

[BDDD06] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. Report CW 472, K.U.Leuven, Department of Computer Science, December 2006.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 1992.

[Bio03] Biovail faces heart drug kickback inquiry. Pharma Marketletter, 1 Sep 2003. http://goliath.ecnext.com/coms2/summary_0199-3378487_ITM.

[Bio08] Grand jury probes biovail over sales practices. The Toronto Star, 1 Feb 2008. http://www.thestar.com/Business/article/299682.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer, 2001.

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *ACM Conference on Computer and Communications Security*, pages 244–250, 1993.

[BM05] Walid Bagga and Refik Molva. Policy-based cryptography and applications. In *Financial Cryptography*, volume 3570 of *LNCS*, pages 72–87. Springer-Verlag, 2005.

[BM06] Walid Bagga and Refik Molva. Collusion-free policy-based encryption. In *Proceedings of the 9th International Information Security Conference*, volume 4176 of *LNCS*, pages 233–245. Springer-Verlag, 2006.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[Bou00] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology – EuroCrypt'00*, volume 1807 of *LNCS*, pages 431–444. Springer Verlag, 2000.

[Boy06] Xavier Boyen. A promenade through the new cryptography of bilinear pairings. In *IEEE Information Theory Workshop—ITW 2006*, pages 19–23. IEEE Press, 2006.

[Bra93] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 1993.

[Bra95a] Stefan Brands. Off-line electronic cash based on secret-key certificates. In Ricardo A. Baeza-Yates, Eric Goles Ch., and Patricio V. Poblete, editors, *LATIN*, volume 911 of *Lecture Notes in Computer Science*, pages 131–166. Springer, 1995.

[Bra95b] Stefan Brands. Restrictive blinding of secret-key certificates. In *EUROCRYPT*, volume 921 of *LNCS*, pages 231–247. Springer Verlag, 1995.

[Bra95c] Stefan Brands. Secret-key certificates. Technical Report CS-R9510, CWI, 1995.

[Bra97] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In *EUROCRYPT*, volume 1233 of *LNCS*, pages 318–333. Springer Verlag, 1997.

[Bra00] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. The MIT Press, 2000. Available online at http://www.credentica.com/the_mit_pressbook.html.

[CE86] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer, 1986.

[CFN88] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer, 1988.

[CFSY96] Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers, and Moti Yung. Multi-autority secret-ballot elections with linear work. In *EUROCRYPT*, pages 72–83, 1996.

[CG99] Ran Canetti and Shafi Goldwasser. An efficient *hreshold* public key cryptosystem secure against adaptive chosen ciphertext attack. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer, 1999.

[CGN98] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003, 1998.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[Cha87] David Chaum. Blinding for unanticipated signatures. In *EUROCRYPT*, pages 227–233, 1987.

[Che95] Lidong Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 232–243. Springer, 1995.

[CHK$^+$06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 201–210. ACM, 2006.

[CHL06] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2006.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[CL02a] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*

'02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, pages 61–76. Springer-Verlag, 2002.

[CL02b] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *Advances in Cryptology – EuroCrypt'01*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in cryptology – Crypto'04*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag, 2004.

[CMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In *Advances in Cryptology – EUROCRYPT'00*, volume 1807 of *LNCS*, pages 122–138. Springer-Verlag, 2000.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Advances in cryptology – Crypto'92*, volume 740 of *LNCS*, pages 89–105. Springer-Verlag, 1992.

[CP94] R. J. F. Cramer and T. P. Pedersen. Improved privacy in wallets with observers. In *EUROCRYPT '93: Workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 329–343, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[CPJ04] Claudine Conrado, Milan Petkovic, and Willem Jonker. Privacy-preserving digital rights management. In Willem Jonker and Milan

Petkovic, editors, *Secure Data Management*, volume 3178 of *Lecture Notes in Computer Science*, pages 83–99. Springer, 2004.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in cryptology – Crypto'98*, volume 1462 of *LNCS*, pages 13–25. Springer-Verlag, 1998.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Advances in Cryptology - Crypto 2003*, volume 2729 of *LNCS*, pages 126–144. Springer-Verlag, 2003.

[Dam88] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335. Springer, 1988.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT*, pages 125–142, 2002.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001.

[DJ03] Ivan Damgård and Mads Jurik. A length-flexible threshold cryptosystem with applications. In *ACISP*, volume 2727 of *Lecture Notes in Computer Science*, pages 350–364. Springer, 2003.

[FAL04] Keith Frikken, Mikhail Atallah, and Jiangtao Li. Hidden access control policies with hidden credentials. In *WPES '04: Proceedings of the 2004*

ACM workshop on Privacy in the electronic society, pages 27–27, New York, NY, USA, 2004. ACM Press.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[FTY98] Yair Frankel, Yiannis Tsiounis, and Moti Yung. Fair off-line e-cash made easy. In *ASIACRYPT*, volume 1514 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1998.

[Gas04] William I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the European Association for Theoretical Computer Science*, 82:72–107, 2004.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160. ACM Press, 1998.

[Gir08] Damien Giry. Cryptographic key length recommendation, 2008. http://www.keylength.com (Available as of Oct. 2008).

[GLNS93] Li Gong, T. Mark A. Lomas, Roger M. Needham, and Jerome H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[GMM06a] Philippe Golle, Frank McSherry, and Ilya Mironov. Data collection with self-enforcing privacy. In *ACM Conference on Computer and Communications Security—ACM CCS 2006*, pages 69–78. ACM, 2006.

[GMM06b] Philippe Golle, Frank McSherry, and Ilya Mironov. Data collection with self-enforcing privacy. In *ACM Conference on Computer and Communications Security*, pages 69–78, 2006.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, pages 291–304. ACM, 1985.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS*, pages 174–187. IEEE, 1986.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, pages 803–815, 2005.

[HL7a] Health level 7 (hl7). `http://www.hl7.org/`.

[HL7b] Hl7 reference information model. `http://www.hl7.org/library/data-model/RIM/modelpage_non.htm`.

[IHE] Integrating the healthcare enterprise. `http://www.ihe.net/`.

[IS90] Ingemar Ingemarsson and Gustavus J. Simmons. A protocol to set up shared secret schemes without the assistance of a mutualy trusted party. In *EUROCRYPT*, pages 266–282, 1990.

[ISN89] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[IT05] ITU-T. Information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks – X.509 Recommendation, 2005. http://www.itu.int/rec/T-REC-X.509/en.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC Pres, 2007.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.

[KSN$^+$07] Reto Krummenacher, Elena Paslaru Bontas Simperl, Lyndon J. B. Nixon, Dario Cerizza, and Emanuele Della Valle. Enabling the european patient summary through triplespaces. In *CBMS*, pages 319–324. IEEE Computer Society, 2007.

[Lay07] Mohamed Layouni. Accredited symmetrically private information retrieval. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *IWSEC*, volume 4752 of *Lecture Notes in Computer Science*, pages 262–277. Springer, 2007.

[Len05] Arjen Lenstra. *Handbook of Information Security*, volume II, chapter Key Lengths. Wiley, December 2005.

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *Proceedings of the 8th International Information Security Conference*, volume 3650 of *LNCS*, pages 314–328. Springer-Verlag, 2005.

[LLX07] Jiangtai Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In *Procceding of the International Conference on Applied Cryptography and Network Security*, 2007. To appear.

[LRSW00] Anna Lysyanskaya, Ronald Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *SAC '99: Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *LNCS*, pages 184–199. Springer-Verlag, 2000.

[LV00] Arjen Lenstra and Eric Verheul. Selecting cryptographic key sizes. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography – PKC '00*, volume 1751 of *LNCS*, pages 446–465. Springer-Verlag, 2000.

[NP99] Moni Naor and Benny Pinkas. Oblivious transfer with adaptive queries. In *Advances in Cryptology - CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 573–590. Springer, 1999.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.

[OO89] Tatsuaki Okamoto and Kazuo Ohta. Disposable zero-knowledge authentications and their applications to untraceable electronic cash. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 481–496. Springer, 1989.

[OS07] Rafail Ostrovsky and William E. SkeithIII. A survey of single-database private information retrieval: Techniques and applications. In *Public Key Cryptography*, pages 393–411, 2007.

[OU98] Tatsuaki Okamoto and Shigenori Uchiyama. A new public-key cryptosystem as secure as factoring. In *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *LNCS*, pages 308–318. Springer-Verlag, 1998.

[Ped91a] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[Ped91b] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.

[Ped96] Torben P. Pedersen. Electronic payments of small amounts. In *Security Protocols Workshop*, volume 1189 of *Lecture Notes in Computer Science*, pages 59–68. Springer, 1996.

[PV04] Giuseppe Persiano and Ivan Visconti. An efficient and usable multi-show non-transferable anonymous credential system. In *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2004.

[Rot01] Marc Rotenberg. Fair information practices and the architecture of privacy. *Stanford Technology Law Review*, 1, 2001.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4(3):161–174, 1991.

[Sho01] Victor Shoup. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001.

[Sol04] Daniel Solove. *The Digital Person: Technology And Privacy In The Information Age*. NYU Press, 2004.

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society, 2000.

[TPB$^+$06] Svetlena Tavena, Philippe Palanque, Sandra Basnyat, Marco Antonio Winckler, and Effie Law. Clinical application design: Task modeling with failure in mind. In *World Congress on Internet in Medicine (MedNet), Toronto, Canada, 15/10/06-18/10/06*, 2006.

[Ver01] Eric R. Verheul. Self-blindable credential certificates from the weil pairing. In *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 533–551. Springer, 2001.

[VGP05] E. Della Valle, L. Gadda, and V. Perdoni. COCOON: Building knowledge driven and dynamically networked communities within european healthcare systems, April 06 2005.

[YDB06] Yanjiang Yang, Robert H. Deng, and Feng Bao. Fortifying password authentication in integrated healthcare delivery systems. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 255–265, New York, NY, USA, 2006. ACM.

[YXFD04] Y. Yang, X.Han, F.Bao, and R. H. Deng. A smart-card-enabled privacy preserving e-prescription system. *IEEE Transactions on Information Technology in Biomedicine*, 8(1):47–58, 2004.

List of publications

- **Privacy-Preserving Telemonitoring for eHealth**. Mohamed Layouni, Kristof Verslype, Mehmet Tahir Sandıkkaya, Bart De Decker, and Hans Vangheluwe. In Proc. of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec'09), Lecture Notes in Computer Science, vol. 5645, Springer, 2009, pp. 95–110.

- **Efficient multi-authorizer accredited symmetrically private information retrieval**. Mohamed Layouni, Maki Yoshida, and Shingo Okamura. In Proc. of the 10th International Conference on Information and Communications Security (ICICS'08), Lecture Notes in Computer Science, vol. 5308, Springer, 2008, pp. 387–402.

- **A privacy-preserving ehealth protocol compliant with the belgian healthcare system**. Bart De Decker, Mohamed Layouni, Hans Vangheluwe, and Kristof Verslype. In Proc. of the fifth European PKI Workshop (EuroPKI'08), Lecture Notes in Computer Science, vol. 5057, Springer, 2008, pp. 118–133.

- **Accredited symmetrically private information retrieval**. Mohamed Layouni. In Proc. of the 2nd International Workshop on Security (IWSEC'07), Lecture Notes in Computer Science, vol. 4752, Springer, 2007, pp. 262–277.

- **Anonymous k-show credentials**. Mohamed Layouni and Hans Vangheluwe. In Proc. of the fourth European PKI Workshop (EuroPKI'07), Lecture Notes in Computer Science, vol. 4582, Springer, 2007, pp. 181–192.

- **A survey of privacy-preserving digital credentials**. Mohamed Layouni. To be submitted.