KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)

Cryptographic Protocols and System Aspects for Practical Data-minimizing Authentication

Patrik BICHSEL

Dissertation presented in partial
fulfillment of the requirements
for the degree of Doctor
in Engineering

March 2012

Cryptographic Protocols and System Aspects for Practical Data-minimizing Authentication

Patrik BICHSEL

Jury:
Prof. Dr. Adhemar Bultheel, chairman
Prof. Dr. Ir. Bart Preneel, promotor
Dr. Jan Camenisch
(IBM Research – Zurich, Switzerland)
Prof. Dr. Ir. Bart De Decker
Prof. Dr. Ir. Claudia Diaz
Prof. Dr. Dieter Gollmann
(TU Hamburg-Harburg, Germany)
Dr. Gregory Neven
(IBM Research – Zurich, Switzerland)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

March 2012

D/2012/7515/28
ISBN 978-94-6018-492-5

Privacy is Dead.

Steven Rambam

Privacy is Not Dead.

Danah Boyd

Preface

Over the past four years, I have met a number of very inspiring people. It would be extremely challenging to mention each and everyone who has contributed in one or the other way to this work or, even more important, to my personal development. Moreover, it is impossible for me to thank each of those interesting individuals in a way that would be appreciated by the respective person. Therefore, I want to name here only the people who had a key role with respect to to my PhD from a scientific as well as an administrative view.

First, I want to thank my promoter, Prof. Bart Preneel, for agreeing to supervise me in a setting that has complicated several matters, for dealing with those complications in a correct and precise, yet pragmatic way. Second, a special 'thank you' goes to Dr. Jan Camenisch for supervising my work on a weekly basis, coping with various frustrations, and motivating me to strive for continuous improvement. Third, I am very thankful to my examination committee consisting of Prof. Bart De Decker, Prof. Claudia Diaz, Prof. Dieter Gollmann, and Dr. Gregory Neven, for their tremendously helpful input, their guidance, and the interesting discussions. Further, I would like to thank Prof. Adhemar Bultheel for kindly agreeing to act as chairman of my jury. Finally, I am extremely thankful for the priceless support from Péla Noë for the various administrative challenges, and for the work by Emilia Käsper and Sebastiaan Indesteege who spared me from a lot of trouble in typesetting this manuscript.

As mentioned before, there are no words that really do justice to the way that so many people have contributed to this thesis. Therefore, I rather mention a memorable event that I shared with each one of the great people I happened come across, and that I thankfully look back on – I hope you find yours! I am very grateful for the collaborations I had during the past years, where especially the hours right before a submission deadline will definitely be one of my most favourite experiences. Similarly, I am looking back with some grieve on the time I spent in our cafeteria – not because of the delicious meals. I enjoyed the discussions I had with many of you irrespective of the particular topic. Be it server functionality, Russian specialities, motorbikes, optimizing personal finances, or drum kit developments, I enjoyed exchanging news and opinions with so many of you. Moreover, I will miss all the bike rides and jogging sessions where we discussed random thoughts, ideas, or (research or other) challenges. Clearly, I did not spend the whole last four years in Zurich and during the numerous trips I

enjoyed during my time at IBM Research, I also collected some memories. For example, I remember a funny stroll in the rain somewhere in the US, several delicious espressos in Italy, a run in the snow of Munich, and even some nights in a prison cell in Sweden[1].

There is more to a PhD than programming and publishing papers. I cannot think of words that would get even close to expressing my gratitude for my family and friends. You managed to find the right reaction to each situation that I was getting myself into. You comforted me after each (perceived) failure, celebrated with me upon each success, encouraged me when I did not see the end of the tunnel, helped me focus when I lost sight of my goals, and cheered for me at the finishing line.

You made it possible.
You helped me a lot.
Now it is finished.
Thank you!

Patrik Bichsel

[1] No worries – it was in the Långholmen hotel in Stockhom.

Abstract

The Internet is transforming itself and the daily lives of people at a fascinating pace. It revolutionizes the exchange, retrieval, and publishing of information for companies just as it does for individuals. This transformation manifests itself, for example, in the myriad of services that are being offered over the Internet today. *Authentication* mechanisms are one aspect that has not kept pace with this development. While being ubiquitous in today's digital society, most of the current service providers still rely on authentication based on a username and password combination.

This popular approach to authentication and its realization in practice have several shortcomings. First, users tend to choose weak passwords and to re-use the same username and password for several service providers. This user behaviour gives rise to a multitude of attack vectors threatening the effectiveness of the authentication approach. Second, for accountability reasons, service providers request the release of extensive amounts of personal information at registration time. As only a few attributes comprise authentication capabilities, a user may supply arbitrary attribute values, which results in bad data quality for service providers. Consequently, users and service providers have a mutual interest in looking for innovative approaches to authentication.

Techniques to increase the data quality of service providers by exchanging certified attributes, using technologies such as OpenID or SAML, steadily gain interest. However, such methods come at the price of making the personal information hosted at service providers a more attractive target for attackers, thereby raising the cost for protective measures. In addition, they lead to an immense dispersion of personal information of users, effectively leading to all transactions of a person becoming linkable. *Data-minimizing authentication* does not suffer from such deficiencies. Leveraging cryptographic techniques, this authentication paradigm realizes the seemingly conflicting goals of service providers, who desire good data quality or strong authentication guarantees, and users, who want to only reveal as little (personal) information as possible.

The main goal of this work lies in enhancing the practicability of data-minimizing authentication techniques. We structure our contributions into two parts. In the first part, we introduce mechanisms that improve the efficiency of group signature schemes and anonymous credentials, which are systems implementing the concepts of data minimization. Namely, we propose

a group signature scheme that provides the shortest signature size and comes with an efficient signature-generation algorithm. In addition, we present an efficient implementation of anonymous credentials on a smart card, an extremely resource-constrained device. Thereby we demonstrate the feasibility of implementing computationally-intensive authentication technology on currently available hardware. Further, we present a modular architecture for the Identity Mixer anonymous credential system and show how it can be integrated into a standards-compliant authentication environment.

In the second part, we analyze how people and organizations present themselves in the digital domain and what mechanisms exist to achieve recognition processes as in the offline world. Such an analysis proves useful for anticipating problems that may arise when deploying data-minimizing authentication. At the same time we use the insights gained in the recognition and authentication process to illustrate the main ideas of data minimization, thus contributing to a better understanding of this authentication concept. Further, using our conceptual approach to digital identity, we propose an intuitive mechanism for managing trust in attributes of people. Our approach aims at replacing today's cumbersome bootstrapping of personal trust relations.

Samenvatting

Het Internet verandert zowel zichzelf als onze dagelijkse levens aan een razende snelheid. Het zorgt voor een revolutie in de uitwisseling, het opvragen en het publiceren van informatie, zowel voor bedrijven als voor individuele personen. Deze transformatie manifesteert zich bijvoorbeeld in de ontelbare diensten die vandaag online worden aangeboden. In deze omgeving zijn authenticatiemechanismen een aspect dat geen gelijke tred heeft gehouden. Alhoewel ze alomtegenwoordig zijn in de huidige digitale maatschappij, vertrouwen de meeste online dienstenaanbieders nog steeds op authenticatie op de combinatie van gebruikersnamen en wachtwoorden.

Deze populaire aanpak voor authenticatie en de realisatie ervan in de praktijk hebben verschillende tekortkomingen. Ten eerste hebben gebruikers de neiging zwakke wachtwoorden te kiezen en dezelfde gebruikersnamen en paswoorden te hergebruiken bij verschillende dienstenaanbieders. Dit gebruikersgedrag geeft aanleiding tot meerder aanvalsvectoren die de effectiviteit van de authenticatie bedreigen. Ten tweede vragen dienstenaanbieders om verantwoordingsredenen buitensporige hoeveelheden aan persoonlijke informatie op tijdens de registratie. Gezien de juistheid van de opgegeven attribuutwaarden slechts zelden gecontroleerd kan worden, kunnen gebruikers valse waarden opgeven, met slechte gegevenskwaliteit voor de dienstenaanbieders als resultaat. Gebruikers en dienstenaanbieders delen daarom een interesse in het zoeken naar alternatieve oplossingen voor authenticatie.

Technieken om de gegevenskwaliteit van dienstenaanbieders te verbeteren winnen geleidelijk aan belangstelling, meestal door het uitwisselen van gecertificeerde attributen met behulp van technologieën zoals OpenID of SAML. Dergelijke methodes hebben echter het nadeel dat de persoonlijke informatie die opgeslagen wordt door de dienstenaanbieder een aantrekkelijkere prooi vormt voor aanvallers, zodat ook de kost om die informatie te beschermen omhoog gaat. Bovendien werken ze een enorme verspreiding van persoonlijke informatie in de hand, waardoor alle transacties van gebruikers aan mekaar gekoppeld kunnen worden. Dataminimaliserende authenticatie heeft niet te lijden onder dergelijke nadelen. Met behulp van cryptographische technieken realiseren zulke authenticatiemechanismen de schijnbaar tegengestelde doelstellingen van dienstenaanbieders, die hoge gegevenskwaliteit of sterke authenticatiegaranties verlangen, en gebruikers, die zo weinig mogelijk persoonlijke informatie willen vrijgeven.

We structureren onze bijdragen aan het dichter bij de praktijk brengen van dataminimaliserende authenticatietechnieken in twee delen. In het eerste deel stellen we mechanismen voor die de efficiëntie van groepshandtekeningsschema's en anonieme certificaten verbeteren; beide zijn systemen die het concept van dataminimalisatie verwezenlijken. We stellen namelijk een groepshandtekeningsschema voor met de kortste handtekeningen en met een efficiënt algoritme voor het genereren van handtekeningen. Bovendien presenteren we een efficiënte implementatie van anonieme certificaten op een chipkaart, qua rekenkracht een uiterst beperkt toestel. Daarmee tonen we de technische haalbaarheid aan van de implementatie van computationeel intensieve authenticatietechnologie op huidig beschikbare hardware. Verder presenteren we een modulaire architectuur voor het Identity Mixer anonyme certificaatssysteem en tonen we hoe het geïntegreerd kan worden in gestandaardiseerde authenticatieomgeving.

In het tweede deel analyseren we hoe mensen en organisaties zichzelf voorstellen in het digitale domein en welke mechanismen bestaan om gelijkaardige herkenningsprocessen te realiseren als in de offline wereld. Een dergelijke analyse is nuttig om problemen te voorzien die kunnen ontstaan bij het uitrollen van dataminimaliserende authenticatie. Tegelijkertijd gebruiken we de inzichten die verworven werden in het herkennings- en authenticatieproces om de belangrijkste ideeën van dataminimalisatie te illustreren, om op die manier bij te dragen aan een beter begrip van dit authenticatieconcept. Tenslotte stellen we, gebruik makende van onze conceptuele aanpak van digitale identiteit, een intuïtief mechanisme voor om vertrouwen in persoonlijke attributen te beheren, dat de huidige moeizame opstartprocedure van persoonlijke vertrouwensrelaties tracht te vervangen.

Contents

List of Figures

List of Tables

List of Abbreviations

Idemix	Identity Mixer
API	Application Programming Interface
BBS	Boneh-Boyen-Shacham signatures
BCNSW	Bichsel-Camenisch-Neven-Smart-Warinschi signatures
CARL	Credential-based Authentication Requirements Language
CCA	Chosen Ciphertext Attack
CL	Camenisch-Lysyanskaya signatures
CPA	Chosen Plaintext Attack
CPU	Central Processing Unit
CRL	Certificate Revocation Lists
DAA	Direct Anonymous Attestation
DNS	Domain Name System
DP	Delerablée-Pointcheval signatures
ECC	Elliptic Curve Cryptography
EEPROM	Electrically Erasable Programmable Read-Only Memory
eID	electronic IDentity
EPL	Eclipse Public License
ESN	Electronic Social Network
IdP	Identity Provider
OSI	Open Systems Interconnection
PIN	Personal Identification Number

PKI	Public-Key Infrastructure
RP	Relying Party
RSA	Rivest, Shamir and Adleman
SAML	Security Assertion Markup Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
Tor	The Onion Router
UI	User Interface
URL	Uniform Resource Locator
WoT	Web of Trust
XACML	eXtensible Access Control Markup Language

Data-Minimizing Authentication

Introduction

The number of people with regular access to the Internet and the time that those people spend online are increasing at a fascinating pace. Several reasons such as the decrease of bandwidth and device prices, or the availability of portable devices are contributing to this increase. The constant availability of the Internet effectively changes key aspects of traditional businesses, such as service delivery and customer relations. In addition, the widespread availability of Internet access creates opportunities for emerging companies using entirely new business models. Examples of such businesses are the online book store Amazon, the social network sites Facebook or MySpace, or the music streaming services Spotify or Last.fm. The combination of traditional businesses leveraging the Internet and new businesses emerging because of it, lead to a substantial increase of the number of online *services*.

There are various challenges that come along with offering services over the Internet. A major issue is the *security* of those services, which consists of many facets and is interpreted differently depending on the audience. Traditionally, information security distinguishes confidentiality, integrity, and availability of data, where those properties apply for data residing on computing systems as well as for communication data. Intuitively, confidentiality refers to data only being accessible to its intended recipient, integrity denotes the fact that data is unchanged with respect to a defined previous state (e.g., before being sent over a network), and availability denotes that data can be retrieved when being requested. Furthermore, a central aspect when thinking about integrity and confidentiality in communication networks, is the authenticity of data, which denotes the assurance that the originator of transmitted data corresponds to a the intended party. In this work we concentrate on communication security aspects, more specifically on the authenticity of exchanged information. One reason of this

choice lies in the fact that *authentication* of the communication partner is the first step when initiating communication over a network with no inherent security guarantees, such as the Internet. Therefore, authentication is key to bootstrap secure communication and basis for attaining further security properties [81]. A second reason is that technology for the establishment of message confidentiality and integrity, namely Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) [61], is already widely deployed. Our main focus will be on the authentication of users of a service since SSL/TLS provide server-side authentication, and the privacy requirements of users are more multifaceted than those of service providers. We strive for an efficient, user-friendly mechanism with well-defined security properties that has the potential for as broad acceptance as SSL/TLS. Note that the effectiveness of SSL/TLS server-side authentication is challenged [59, 64, 104] but there are efforts of browser manufacturers to improve this situation [77].

1.1 Authentication

Abstractly, an authentication operation is an interactive protocol between two communication partners, which we denote as the *authenticator* and the *authentication subject*. The goal of the authenticator is to get an asserted statement about the authentication subject. The statement bears an assurance on the basis of which the authenticator further operates, for example, it may be used to take an access control decision. In the context of transactions between a user and a service provider, this translates to the service provider requesting the assurance that the user it is communicating with, has certain *properties*. We consider anything that specifies and distinguishes a subject (or a set of subjects) from another set of subjects, as a property. Examples of such properties are an employment relation, access right, or personally identifying attributes such as the nationality, age, name, or date of birth of a person.

In a successful authentication operation the authenticator learns that the authentication subject possesses certain properties. These properties implicitly define a set comprising all entities that own the same properties. The authentication subject remains anonymous withing this set entities, thus the latter is typically denoted as its *anonymity set*. For example, after an authenticator requests the assurance that the subject currently lives in the city of Zurich and the subject provides a corresponding proof, the latter may still be any person living in Zurich. Thus, the anonymity set of the authentication subject is the set of people living in the city of Zurich. Authentication may lead to the *identification* of a subject, which is the case if the anonymity set consists of a single entity.

We can distinguish three categories of authentication mechanisms. More concretely, the authenticator can verify (1) something a person *knows*, (2) something she *has*, or (3) something she *is*. The first category includes a password, personal identification number (PIN), or the response to a previously agreed

question. The second one assumes possession of a device such as a smart card or a one-time-password token, whose possession can be verified by the authenticator. The third category subsumes biometric elements such as the signature, typing pattern, fingerprint, or face of a person. If the authenticator requests a strong assurance level, it may require elements of two categories to be combined in one authentication transaction, thereby implementing *two-factor authentication*. For instance, the combined use of a smart card and a password in an authentication interaction implements such two-factor authentication.

1.1.1 Authentication Methods Today

Today, the most favoured authentication method on the Internet is to request a user to register an *authentication tuple* consisting of a username and a password when she signs up for a service. Authentication of a returning customer is achieved through requesting her to provide a valid authentication tuple. In other words, in online scenarios the dominating authentication method uses the knowledge of a user. This stands in discrepancy to the offline world where the second and third method (authentication based on something a user *has* or *is*) are predominant and knowledge-based authentication is merely added as second method for two-factor authentication. The reason for this discrepancy results from the differences between offline and online situations. In the offline world an authenticator has plenty of options, such as physical credentials or biometric features, to authenticate a user. Contrarily, in an online scenario the authentication subject and authenticator are not physically close, that is, the relevant information needs to be exchanged over a network. This forces all authentication data to be digital, thus rendering the traditional verification of physical credentials and biometric features difficult. Using the Internet for the information transfer, no (a priori) guarantees about a communication partner can be taken for granted.

While the use of an authentication tuple is a simple method for authenticating users, this mechanism has severe issues. First, users tend to pick passwords that are easy to remember, consequently, they are easily guessed by an attacker. Second, the method can be attacked using techniques such as phishing or spear phishing. Third, the myriad of services that are offered today on the Internet causes users to re-use their authentication tuple [112]. There are several implications to such practice: (1) a (malicious) service provider gains an advantage in attacking further (honest) service providers, and (2) a successful attack on one service provider results in a compromise of additional, unrelated service providers [72]. In particular, if the tuple is used for differently valued information, for example, e-banking and music streaming, an attacker may gain knowledge for breaking into a high-value account with significantly less effort compared to attacking the latter directly.

A further weakness of the current authentication approach lies in the fact that service providers request a user to release a large number of self-provided attribute values upon registration. One reason for this requirement lies in the desire of a

service provider to hold users accountable. However, as most attribute values such as the name or date of birth can be chosen arbitrarily by the user, the quality of a service provider's data may be poor. Only very few attributes (e.g., email, telephone number, or address) bear an external verification mechanism, which can be used to assure the service provider of the correctness of an attribute value. In combination with the fact that service providers require users to release personal information irrelevant to the service at hand, the latter end up with huge amounts of data that is to be protected as if it were personal information. Incidents such as the attack on Sony [8, 9, 105], where company servers have been hacked and significant amounts of personal information have been stolen, show the downsides for companies of hosting an extensive set of user data. Specifically, inappropriate protection of such information can pose a considerable risk to business operation and company reputation [107]. The situation of users is just as uncomfortable. One can argue that the diffusion of personal information can be limited if a user simply provides wrong information, but such action is often forbidden according to the terms and conditions of a service. Moreover, during the registration process users are focused on getting access to the service and they often release too much personal information. In summary, the current usage of authentication tuples offers very limited assurance, results in bad data quality for service providers, and severely threatens users' privacy.

There are a few approaches to authentication that do not rely on a simple tuple for authenticating users. One example are CAPTCHAs that, combined with an authentication mechanism, provide the authenticator with the assurance that the subject is probably a human and not a machine. A further procedure consists of using different forms of two-factor authentication. This is especially attractive for high-value services such as e-banking or remote workplace access. However, such solution is not applicable for the bulk of services because of (1) its implementation costs, and (2) the identification (instead of authentication) it results in.

1.1.2 Attribute-based Authentication

The dispersal of personal information of an authentication subject is not unavoidable. Mainly for the reason of attaining better data quality and easing the registration process, protocols that allow for attribute transfer between service providers start gaining interest. Examples of such protocols are OpenID [52], SAML [88], or WS-Federation [11]. The general information flow of those protocols happens between a so called *Identity Provider* (IdP) and a *Relying Party* (RP), where the IdP sends (attribute) information about an authentication subject to the RP. Assuming that the RP trusts the IdP on its statements about the subject, this approach mitigates the data quality issue and relieves the user of the plethora of authentication operations. She would only need to authenticate to the IdP who forwards attribute information to RPs upon their request and the user's authorization of such transaction. A disadvantage of this solution is that it introduces an entity, namely the IdP, that learns detailed information on the

behaviour of its users. Some protocols (e.g., OpenID) offer to hide the RP with respect to the IdP, which is only a satisfactory solution assuming that the parties do not collude and compare their transaction information. Further, all mentioned protocols require the IdP to be online, which comes along with elevated costs for guaranteeing availability and risks of attacks on the IdP.

There are technologies that allow the transfer of attributes between an *offline* IdP and an RP. The most prominent example for such technology are X.509 certificates [53], in which an IdP signs the correspondence between a number of attributes. An important use case of X.509 certificates are e-business applications, where the certificate asserts the binding between the Uniform Resource Locator (URL) and the public key of a company. Combined with a user's trust in the IdP who created the certificate, she may verify the authenticity of the key and use it to establish an authentic and confidential connection to the intended service provider. This example shows that the transfer of attribute values (e.g., the public key) of the service provider to the RP (i.e., the user) using X.509 certificates avoids the need of direct communication between RP and IdP. One issue with X.509 certificates when it comes to user authentication lies in the fact that *all* attributes enclosed in a certificate need to be revealed to the verifier. Consequently, using a comprehensive X.509 credential for persons effectively solves the data quality issue of service providers at the cost of user's privacy. While simple certificates containing only one attribute seem to solve this issue, they raise other challenges because the certificates of several parties may be combined and used in a malicious way.

1.2 Data-minimizing Authentication

In many scenarios the knowledge of non-identifying properties about a subject are sufficient to enforce access control, supply information that has been requested, or otherwise deliver a service. However, if less information is to be communicated, the authenticator requires trustworthy data, for example, data that is certified. We speak of *data-minimizing authentication* when an authentication subject uses a minimal set of certified statements to convince an authenticator that she possesses certain properties. Determining such minimal set of statements, which may differ significantly based on the scenario, is a research question of its own and out of the scope of this work. An advantage for service providers in minimizing authentication data lies in the fact that they are relieved from storing and protecting more personal information than what is relevant to their specific business. In addition, users profit from less diffusion of personal information, that is, better privacy, reduced risk of identity theft, and an improved authentication experience by not having to enter authentication credentials overly often. Thereby, this technology has the potential to introduce a paradigm shift in authentication.

1.2.1 Basic Functionality

Cryptographic protocols are the basis of data-minimizing authentication. Specifically, *zero-knowledge proofs* [14, 24, 92] allow a user to prove a statement to a communication partner, where the latter learns the statement but no additional information. Such proof can be used during the authentication process, that is, a user can prove that she has some attribute on the basis of which she should be granted access without revealing further details about this or any other attribute. For instance, a user can prove that she is of Belgian nationality and therefore should be granted access to a Belgian petition system [76].

In Chapter 2 we discuss data-minimizing authentication systems in depth, where we focus on our contributions towards their efficiency [17,18] and integration into a general authentication landscape [16, 19]. Let us first illustrate one main feature of data-minimizing authentication, namely, *unlinkable* transactions, on the example of anonymous credential systems [29, 38]. We provide an intuition of this functionality through the comparison of anonymous credentials with X.509 certificates. Similar to X.509 certificates, anonymous credentials certify a set of attributes together with an element that corresponds to a secret key. In an X.509 certificate this element is the public key that uniquely corresponds to the secret key as it is generated, for example, using the RSA key generation algorithm [73, 96]. We denote this element corresponding to a secret key in an anonymous credential system as a *pseudonym*. Conversely a public key in a X.509 certificate, in anonymous credential systems an entity can compute an arbitrary number of unlinkable pseudonyms from a single secret key. More concretely, we can determine two characteristics of pseudonyms: (1) a party can only successfully relate to a given pseudonym with knowledge of the secret key using which it has been generated (and possibly further information), and (2) given two pseudonyms it cannot be decided whether or not they correspond to the same secret key. Consequently, a user can obtain an anonymous credential using one pseudonym and authenticate towards an authenticator using a different pseudonym. Due to the second property of pseudonyms, according to which they are unlinkable, a user can achieve the unlinkability of all of her transactions. Note, depending on the implementation, the unlinkability of two pseudonyms may hold computationally or information-theoretically.

1.2.2 Anonymous Communication

The unlinkability of transactions using data-minimizing authentication can only effectively be implemented if the underlying communication channels offer similar functionality. There are several proposals for implementing anonymous communication networks ranging from simple proxies to sophisticated techniques such as mixes introduced by Chaum [45], mix networks, or peer-to-peer anonymization networks such as Crowds [95]. An anonymization proxy is a relay between incoming and outgoing messages, which results in all users of the same proxy becoming part of one anonymity set. Such approach requires high confidence

that the operator of the proxy, who learns all transactions of its users with any service provider, will protect and not misuse this information. As the operator of a proxy learns more about a user than any of the individual service providers it becomes an attractive target for an attacker trying to de-anonymize transactions.

A mix is a more sophisticated approach compared to a single proxy to attain anonymous communication. It stores messages and forwards them (possibly delayed) thereby hiding the relation between its incoming and outgoing messages. There are several implementations based on the concept of a mix, where we can distinguish high-latency (e.g., Mixmaster [82] and Mixminion [57]) from low-latency (e.g., Java Anon Proxy [106]) versions. The former serve well for email anonymization and the latter may be used to anonymize Web traffic. Onion routing is an instantiation of a mix network in which the path of a message is chosen by the sender who applies multiple layers of encryption. Each party receiving a message removes one encryption layer to reveal the (possibly still encrypted) message and the party to which the message should be sent. An implementation of this technique is called The Onion Router (Tor) [63]. It delivers low-latency, high-bandwidth communication [68] and is the most popular anonymization network [84]. Tor presented weaknesses such as data leakage through DNS requests and to end nodes [66], vulnerability to traffic analysis [84], or to remote device fingerprinting [74]. The I2P project [71] uses so called garlic routing, an evolution of the onion routing methodology. Here, multiple messages are packaged together to increase robustness against traffic analysis. While there are less publications on the security of I2P compared to Tor, Herrmann and Grothoff demonstrate a practical attack on I2P [69].

The selection of a suitable anonymization network, however, not only depends on performance properties such as the latency or bandwidth. Rather, the degree of anonymity it provides is of predominant interest. Diaz [60] or Serjantov and George Danezis [99] provide anonymity metrics that allow for the comparison of the anonymity a user may achieve using different mechanisms. While anonymous communication on the data link and physical layer of the OSI model are a requirement for realizing data-minimizing authentication, we consider them as orthogonal issue.

1.3 Authentication Environment

Data-minimizing authentication makes extensive use of public-key cryptography [62, 96]. This technology comes with many advantages such as its capabilities in bootstrapping network security properties [81]. However, Diffie and Hellman [62] already point out that their key-agreement protocol requires authenticated communication partners, and Maurer and Schmid [80] formalise the requirements to attain secure (i.e., authentic and confidential) communication channels. In summary, authentic retrieval of public keys is a requirement for the use of public-key cryptography.

The requirements of public-key cryptography are not the only challenge that we face in digital communication. Namely, the extensive use of digital communication channels for various purposes raises further challenges. We start Chapter 3 with a discussion of authentication concepts that demonstrate those challenges. Thereby, we extend our own contributions that model digital interaction and recognition processes of people [21]. Further, using the introduced concepts, we show a mechanism to bootstrap trust in a user-friendly, yet secure, way [22]. Our mechanism makes extensive use of Electronic Social Networks (ESNs) to create a Web of Trust (WoT). The main advantage of our approach comes from the fact that people use ESNs already, so our method only leads to a small overhead for them. Such approach could prove useful when it comes to personal key and trust management. For example, it provides the possibility for a person to recover from the loss of her secret key in a simplified manner, not forcing her to start the establishment of people's trust in her public key (and further attribute values) from scratch.

1.4 Guide to this Thesis

This is a publication-based thesis constituting of two parts. The first part situates our contributions in the broader scientific context and compares it to related research results. It further outlines the connections among the publications and investigates their contributions to the respective field. The second part lists the selected scientific contributions of this thesis in their original form (slightly reformatted). Each chapter in Part II consists of a paper published at a peer-reviewed conference.

We structured Part I into an introduction (see Chapter 1) that sets the context relevant to data-minimizing authentication. More concretely, we discuss the authentication situation today, outline the drawbacks and propose means to overcome their limitations. We elaborate on the principles of data-minimizing authentication and its implementation requirements in Chapter 2. This chapter also summarizes our contributions on making data-minimizing authentication more practical. It also critically analyze their strengths and weaknesses by comparing them to other research results. Chapter 3 addresses the issue how people represent themselves online and how they can be recognized by others. Moreover, we outline a trust establishment mechanism for public keys of individuals, which can be leveraged by data-minimizing authentication. Finally, in Chapter 4 we sketch challenges that are obstacles for the transition of data-minimizing authentication from research prototypes to products.

Data-Minimizing Authentication Systems

The goal of data-minimizing authentication lies in the reduction of information disclosed during an authentication operation. However, this authentication paradigm is only likely to succeed if both authenticator and authentication subject benefit from adopting the technology. Consequently, it needs to offer an advantage to two parties who seemingly have conflicting interests, namely, the authenticator requires strong assurances while the authentication subjects prefer to reveal as little information as possible. Cryptographic protocols have the potential to mitigate this conflict of interests, in fact, they allow authenticators to obtain stronger, cryptographically certified, assertions while authentication subjects reveal only minimal information. There are several systems employing public-key cryptography to realize the goals of data minimization. We will look into two systems that differ greatly in terms of functionality, that is, one is highly efficient in a specific scenario but it does not offer many features. The other system offers a very general and extensive set of features but comes with additional architectural and computational complexity.

We start in Section 2.1 with an investigation of a system tailored towards a specific scenario, namely, *group signatures*. Abstractly, a group signature scheme allows a group of people to sign messages on behalf of the group. In a system with a well-defined functionality, a main optimization metric is efficiency, which consists of the length of a signature, the time for computing a signatures, or the computation time when verifying the correctness of a signature. In this discussion we focus on our contribution [18] in this field that consists of combining existing security assumptions in a novel way to attain a group signature scheme with

the shortest group signatures known, combined with the most efficient signature generation algorithm. Section 2.2 focuses on anonymous credential systems, which can be seen as a privacy-enhanced Public-Key Infrastructure (PKI). We first compare anonymous credential systems to a standard PKI, thus following up on the comparison with X.509 certificates started in Section 1.1. Next, we introduce the protocols needed for the implementation of anonymous credentials, shed light on challenges, and highlight implementation aspects. This section presents three of our contributions. Specifically, we show how anonymous credentials can be efficiently implemented on standard Java Cards [17] (see page 103), outline a modular architecture and specification languages [16] (see page 131), and demonstrate how to integrate anonymous credentials into a general authentication system [19] (see page 153). All of these results contribute towards making anonymous credential systems more practical.

2.1 Group Signature Schemes

Group signatures, as originally proposed by Chaum and van Heyst [50], consist of a group manager, an optional opener, and a set of group members $\mathcal{U}$. Each group member $U_i \in \mathcal{U}$ has a (secret) signing key that allows her to sign a message on behalf of the group. After a group member created such a signature, it may be verified by any entity, where the latter only learns that a group member created the signature. More concretely, during the verification process of a signature, the verifier does not learn which *specific* user has created the signature. Consequently, group signatures provide anonymity to the group member U_i signing a message, where the anonymity set of U_i is the entire group $\mathcal{U}$. Each group signature is unlinkable with respect to any other signature, which implies that it is hard to distinguish whether or not two given signatures originate from the same signer. Adding anonymity and unlinkability of signatures to a signature scheme has the potential of misuse by a malicious group member for conducting illegitimate actions. For this case, a practical group signature scheme must provide means to reveal the identity of the group member who has created a given signature, that is, revoke the anonymity of the signature. This task is assumed by the opener, who has a method at its disposal to reveal the creator of any given group signature.

We begin with a discussion of the terminology of group signature schemes in Section 2.1.1. In Section 2.1.2, we continue with an analysis of the security notions that have been proposed for group signature schemes, where we focus on the contributions relevant to our group signature scheme. Finally, Section 2.1.3 summarizes the key aspects of our group signature scheme. Using a new combination of security notions, we achieve the most efficient group signature scheme known [18] (see page 69).

2.1.1 Terminology

Formally, a group signature scheme is composed of a group manager M, an opener O, a set of group members $\mathcal{U} = \{U_i\}$, each with a unique index i, as well as a set of algorithms. A group is specified by the group public key *gpk* and each user has a private signing key $\mathbf{gsk}[i]$ corresponding to this public key. The group manager M and the opener O have secret keys *gmsk* and *osk*, respectively. Further, each user has a user public key pair $(\mathbf{upk}[i], \mathbf{usk}[i])$ of an external signature scheme. Such key pair can be used to ensure that a user U_i cannot be framed, that is, no set of entities (including group manager or opener) can collaboratively create a signature that looks as if it stems from U_i. The public key $\mathbf{upk}[i]$ must be published and authentically retrievable, which could be attained using a PKI. Note that some schemes combine the manager and opener into one entity, thereby elevating the required trust level of group members in the combined entity.

A setup algorithm GSetup generates the group public key *gpk* and the secret key *gmsk* of M as well as *osk* of O upon input of a security parameter. In a *static* group signature scheme the set of group members is fixed at setup time and the private signing keys $\mathbf{gsk}[i]$ of all users are generated during the setup procedure together with *gmsk* and *osk*. *Dynamic* group signature schemes allow users to be added dynamically. Thus, a user i who wants to join the group, first obtains a public key pair $(\mathbf{upk}[i], \mathbf{usk}[i])$ and, second, runs an interactive protocol with the group manager M to get the private signing key $\mathbf{gsk}[i]$. During the protocol, M may retain registration information $\mathbf{reg}[i]$.

Using the signing algorithm, a user can generate a signature σ on a message m with $\mathsf{GSign}(\mathbf{gsk}[i], m)$, which takes the user signing key and the message m as input. Any entity may verify the signature using the signature verification algorithm $\mathsf{GVerify}(\sigma, gpk, m)$. This algorithm merely outputs whether or not the verification has been successful, that is, failure to verify a signature implies that the signature σ (1) has not been generated by a group member of the group specified by *gpk*, or (2) that not the message m has been signed.

The opening entity may further make use of the anonymity revocation algorithm $\mathsf{GOpen}(\sigma, osk, m)$, which outputs the index of the user who created the signature σ or $\perp$ in case of failure. When opening a signature, the opener also creates a proof π attesting to the fact of having correctly revoked the anonymity of the signature. This proof can be verified with the judging algorithm $\mathsf{GJudge}(\pi, i, \mathbf{upk}[i], \sigma, gpk, m)$. The output of the judging algorithm indicates if the proof π, issued by the opener, is correct and that indeed group member with index i created signature σ on message m. Note that the opening and judging algorithm may use of the verification algorithm to assure that a signature σ has been generated correctly. Furthermore, to simplify the notation of the group public key *gpk*, the latter may be composed of the tuple $(gmpk, gopk)$, which are public keys corresponding to the private key of M and O, respectively.

2.1.2 Evolution of Security Notions

Various group signature schemes [5, 25, 26, 58] have been proposed since their introduction by Chaum and van Heyst [50]. The different proposals brought along various security properties such as anonymity, traceability, unforgeability, collusion resistance [7], framing resistance [51], and unlinkability. Bellare, Micciancio and Warinschi [12] proposed the notions of *full-traceability* and *full-anonymity* and showed that they imply all previously proposed security notions in a static setting. They present a group signature scheme in which the number of group members and their identities are fixed at system setup. Specifically, the setup procedure is executed by a party that is trusted by all system participants to correctly execute its tasks. It generates all public and private parameters, which results in two severe drawbacks. First, group members cannot be added to the group after the setup procedure, and second, it requires a high level of trust in the entity running the setup procedure. This limits the applicability of static group signatures in real life, for which reason most schemes do not follow this approach.

Bellare, Shi and Zhang [13] recognize these problems and provide security notions for dynamic group signature schemes. In their setting, the number of group members and their identities are not known at setup time and the trusted entity only chooses the group public key as well as the secret key of the group manager M and opener O. Bellare et al. distinguish the role of the group manager M, who is responsible to add people to the group, and the opener O, who can revoke the anonymity of signatures. Adding group members can be achieved by an interactive join protocol between a user and the group manager. Note that in the static setting, the tasks of the group manager are included into the setup algorithm as, by definition, no user can join the group after the initial setup procedure. Further, Bellare et al. suggest to add the *non-frameability* (or exculpability) notion that ensures that even a colluding opener and group manager cannot construct a signature that would be (falsely) attributed to an honest group member. To this end, an signature scheme external to the group signature scheme assures a group member of not being framed.

Separately to the efforts mentioned before, Boneh and Shacham [27] proposed an anonymity notion that relaxes the full-anonymity notion put forth by Bellare et al. [12]. Specifically, for full-anonymity to hold, the private information of a group member U_i must not provide an advantage in recognizing signatures created by U_i. Expressed differently, the anonymity of a signature must hold even towards the group member that created the signature in question, that is, a user must not be able to recognize her own signatures. In their paper, Boneh and Shacham conclude that this requirement may be too rigorous for some practical purposes and propose *selfless anonymity*. This notion does not require the anonymity of a signature to hold towards the group member who created it. Note that there are different attack models for anonymity, namely anonymity against CPA (chosen plaintext attack) or CCA2 (chosen ciphertext attack). As experienced in the example of encryption [23], we foresee the latter notion to be relevant in practice.

2.1.3 Our Group Signature Scheme

Several initial proposals of group signature schemes were based on the Strong-RSA assumption [5,6,39]. Elliptic curve cryptography (ECC) typically allows for smaller key lengths compared to RSA-based schemes to achieve the same security level. As a consequence, the most efficient schemes in signature length and computational efficiency today [26, 27, 40, 58, 100] all employ ECC. In our paper [18] we propose for an efficient ECC-based group signature scheme.

A main contribution of our paper is the group signature notion that builds a hybrid between the full-traceability, non-frameability [13], and the selfless anonymity [27] notions. Specifically, we propose a syntax of a group signature scheme in the dynamic setting that obtains selfless anonymity, traceability, and non-frameability. Most notably, the anonymity notion allows us to break with a widely used construction paradigm, where a group signature consists of (1) an anonymous signature, (2) an encryption of the signer's index on behalf of O, and (3) a non-interactive zero-knowledge proof convincing a verifier that the encrypted index indeed corresponds to the signer. Our group signature scheme achieves the desired security properties without explicit encryption of the identity of the signer, which results in the most efficient group signature scheme in signature length and signature computation. Note that our scheme combines the group manager and the opener into one entity: the group management entity. During the join protocol, the user and the group management entity jointly generate the user secret key and the management entity issues a Camenisch-Lysyanskaya (CL) signature [40] on this secret to the user. Signature generation involves the randomization of the CL signature and a proof of knowledge of the user secret key underlying the signature. The key point in our scheme is that the joint generation of the user secret key leaves the management entity with sufficient information to open signatures, yet not enough to frame an honest user.

Scheme	$\mathbb{G}_1$	$\mathbb{Z}_q$
BCNSW	3	2
CL	7	4
DP	4	5
BBS*	4	5

Table 2.1: Comparison of the length of one signature for the most efficient group signature schemes. We list the number of elements from $\mathbb{G}_1$ as well as from $\mathbb{Z}_q$.

A further contribution of our paper is an extensive comparison of our own scheme (BCNSW) with the three most efficient group signature schemes, namely Camenisch-Lysyanskaya [40], Delerablée-Pointcheval [58], and Boneh-Boyen-Shacham [26] signatures. In our comparison tables we denote those schemes as CL, DP, and BBS, respectively. Our comparison comprises the bit length

of one group signature (see Table 2.1), and the (theoretical) computation time for generating as well as for verifying a signature (see Table 2.2 and Table 2.3). Note that our comparison needs to consider the security notions with respect to which the schemes are proven secure. For example, we outline and compare against a variant of BBS signatures [26] incorporating comments published by Shacham [100], which we denote as BBS*. In contrast to the original BBS signature scheme, the BBS* version guarantees CCA2-anonymity, where an attacker has access to an opening oracle. Consequently, all compared signature schemes offer CCA2-anonymity.

Scheme	$\mathbb{G}_T^5$	$\mathbb{G}_T^3$	$\mathbb{G}_T^2$	$\mathbb{G}_T$	$\mathbb{G}_1^2$	$\mathbb{G}_1$
BCNSW				1		3
CL			1		1	11
DP		1			1	6
BBS*	1				3	5

Table 2.2: Comparison of *signature generation* costs for the most efficient group signature schemes in terms of the number of exponentiations.

Scheme	P^2	P	$\mathbb{G}_T^3$	$\mathbb{G}_2^2$	$\mathbb{G}_1^4$	$\mathbb{G}_1^3$	$\mathbb{G}_1^2$	$\mathbb{G}_1$
BCNSW	2						1	1
CL	2			1		2	2	1
DP		1	1	1		1	2	
BBS*	1				1	1	4	

Table 2.3: Comparison of *signature verification* costs for the most efficient group signature schemes in terms of the number of exponentiations.

In Table 2.1 we can see that our scheme offers the shortest signatures by approximately a factor of two. The computation times are more difficult to compare theoretically as they vary drastically depending on the exact implementation details such as the type of the pairing [67]. As a consequence, an exact comparison of the computation times of ECC-based group signature schemes can only be provided by an implementation of all schemes, which is a huge effort considering the various optimisations that can be made for each specific choice of the underlying mathematical structures. Therefore we compare the computational complexity using the number of exponentiations and pairing, where in our tables the operations are sorted from left to right; from computationally most to least expensive. We list the minimal number of required exponentiations in each group (e.g., $\mathbb{G}_1$ denotes an exponentiation in group $\mathbb{G}_1$), multi-exponentiations (e.g., $\mathbb{G}_1^3$ denotes a multi-exponentiation with three bases and corresponding exponents), pairings (i.e., P denotes a pairing), and multi-pairings (e.g., P^n denotes the multiplication of n pairings). From Table 2.2 we can conclude that our proposed scheme results in the most efficient signature computation as it involves computationally less intensive operations compared to its competition. Table 2.3 shows that our scheme is very competitive in its signature verification complexity, where we note that the verification of group signatures can be performed in different ways resulting, for example, in less pairings but more exponentiations in different groups.

Short signatures are desirable because they can be efficiently transmitted, and they require little storage space. The efficient transmission helps preserving bandwidth and reducing the overall transaction time. For scenarios involving the implementation of group signatures on resource-constrained devices, it is even crucial for limiting the energy consumption. The storage efficiency contributes to realising group signatures in the same scenario. Finally, the small number of computationally complex operations in the signature generation process is also most relevant for resource-constrained devices as computation is another energy intensive operation.

An efficiency drawback of our scheme is the fact that revocation of the anonymity of a signature becomes an operation with complexity linear in the number of group members. This directly results from removing the encryption towards the opener, which is the source for attaining short signatures and an efficient signing process. A further drawback of our proposal lies in the combination of the roles of the opener and the group manager. This implies that users need to trust this combined entity for correctly executing all group management tasks. In other words, if the combined entity is corrupt, it has a more significant impact on the group signature scheme. A re-design on the join protocol, in which a user would interact with M and O individually, could lead to separating the management entities as in Bellare et al. [13].

2.2 Anonymous Credential Systems

Anonymous credential systems as first outlined by Chaum [45,48] and implemented by Brands [29] or Camenisch and Lysyanskaya [37] can be seen as a privacy-enhanced PKI. Therefore, we start their discussion by briefly explaining the differences in terminology between a standard PKI and an anonymous credential system before outlining the roles involved in an anonymous credential system in Section 2.2.1. We elaborate in Section 2.2.2 on concepts that allow for an implementation of anonymous credentials. We compare the procedure of the two known approaches in realizing the features of anonymous credentials and highlight their distinctions. Next, we discuss the protocols used in an anonymous credential system in Section 2.2.3, where we also introduce extensions on an abstract level, not focusing on any particular implementation. In Section 2.2.5 we conclude with a description of important implementation aspects when making anonymous credential systems practical.

The final section comprises three of our contributions. First, we describe our implementation of anonymous credentials on a resource-constrained device [17] (see page 103). Our implementation demonstrates the feasibility of running an anonymous credential system on a smart card. It even meets the stringent constraints for electronic identity (eID) cards, which is an interesting scenario for the use of anonymous credentials. Second, we outline our design of a modular architecture of the Identity Mixer (*Idemix*) anonymous credential system [16] (see

page 131). The architecture includes the specification of general components that allow for interoperability between credential technologies and contribute towards the ease of using *Idemix*. Third, in order to establish anonymous credential systems in an authentication ecosystem, the authentication and access control policies become a key aspect. More concretely, we need policies that can express the data-minimizing authentication principles of the underlying technology. To this end, we provide an prototype integrating *Idemix* into a general authentication framework [19] (see page 153). In the same paper we describe how other technologies such as U-Prove may be incorporated into the framework and compare authentication solutions with respect to to their support for data minimization.

2.2.1 Terminology

A PKI serves the purpose of binding the public key of an entity to attribute values of the same entity. This allows a third party to be convinced of the correctness of this binding, and therefore use the public key for an operation related to those attributes. For instance, assuming that the URL and public key of an entity are bound in the ascribed way allows a third party to verify the information on the Web page located at the given URL. In this context, an attribute consists of the combination of a (universal) name and an entity-specific value. Examples of attribute names are first name, birth date, email address, or URL. In a standard PKI, a set of attributes together with a signature is called a certificate, in which each attribute name is associated with the value of the attribute specific to the entity receiving the certificate. In an anonymous credential system, the concept of a certificate is denoted as a *credential*. More generally, a *certification token* refers to either a certificate or a credential. Further, we refer to the signature of the issuing entity more generally as *cryptographic evidence*. Where a certificate is issued by a certificate issuing authority (CA), we denote the entities issuing credentials as *issuers* or *identity providers* (IdPs). An IdP executes an interactive protocol to issue a credential to the *recipient* of the credential. We denote the entity receiving a credential also as the *holder* or *owner* of a credential and, for now, abstract from the problem that a credential could be held by an entity not in possession of the certified attribute values. Note, the intuition behind the name of an IdP comes from the scenario where an issuer certifies personally identifiable information, thereby providing the credential recipient with certified attribute values of a part of her identity.

Issuing a certification token assumes the presence of verifying entities, *verifiers* or *relying parties* (RPs). The name of an RP relates to the fact that it relies on the IdP for properly verifying the correspondence of a credential's attribute values to the entity the credential is issued to. In an anonymous credential system, the verification is an interactive protocol in which a credential owner acts as *prover* and communicates with a verifier to release a *proof of possession*. This protocol bears the biggest difference between a standard PKI and the privacy-enhanced variant. A certificate of a standard PKI can only be verified if all information

pertaining to it, that is, all certified attribute values including unique identifiers, are disclosed to the verifying entity. Credentials, however, allow its holder to release a subset of the information contained in the credential to an RP. Note that there are extensions to anonymous credential systems that require a *trusted third party* (TTP), which we discuss in Section 2.2.3. Further, the mentioned roles can be assumed by any entity (e.g., user, company, government). As an example, a company can act as an RP and verify a credential of a person just as well as it can act as an IdP and issue a credential to a person. Usually companies, institutions, or governments assume the role of an IdP and natural persons act as credential recipient. Similar, it is more common for organisations to verify credentials in a transaction where an individual acts as prover.

2.2.2 Realizing Anonymous Credentials

Abstractly, the basis for building anonymous credential systems lies in a complete separation of any two transactions involving one credential, for example, issuing and verification of a credential. This separation, that results in *unlinkable* transactions, comes with two requirements for the issuance and proof protocol: (1) both protocols must not use unique identifiers, and (2) the information released through the protocols must not result in making several transactions linkable. The first requirement can be solved by requiring that candidate systems must avoid universal identifiers and carry out each transaction pseudonymously. Therefore, anonymous credential systems are also known as *pseudonym systems* [48, 78]. As pointed out before, a pseudonym is similar to a public key with the difference that arbitrarily many pseudonyms can be generated based on one secret key. Thus, an entity may use a pseudonym p_i when requesting a credential from an IdP and a different pseudonym p_j when running the proof protocol with some RP, where p_i and p_j are unlinkable. Assuming that the credential owner will use a different pseudonym for each run of the proof protocol results in all her transactions being unlinkable. The second requirement is harder to specify precisely, but intuitively we see that unlinkability heavily depends on the information released to an RP. We have to consider the expected anonymity set for a given set of released attributes or even attribute values, which becomes the metric for assessing the effectiveness of the privacy protection. For example, we can conclude that the release of the first name may imply transactions to become linkable. However, since the expected number of people releasing the value “John” is much larger than the number of people having “Euphemia” as attribute value, all people releasing “John” enjoy better protection of their anonymity.

We can distinguish two principles that allow for an implementation of pseudonymous protocols. First, *blind signatures* [46, 47] achieve the required separation by enabling an issuer to sign an message without learning its content. Using the envelope analog mentioned by Chaum [48], we can imagine this process to work like an entity packaging a message together with a carbon paper into an envelope. The sealed envelope is passed to the issuer, who will sign the envelope.

Due to the carbon paper, the signature will end up on the message that has been put into the envelope. The recipient can open the envelope and show the signed message to an RP. The transactions are unlinkable since the issuer has never seen the actual message, which is the only element that the RP knows. Cryptographic techniques even allow the issuer to check properties of the message that it will sign to make sure that no entity can attain a signature on an attribute value that does not correspond to itself. The anonymous credential system proposed by Brands [29] uses this approach, which is implemented in the U-Prove credential system [31]. A second idea is inspired by group signatures (see Section 2.1) that can be seen as credentials without attributes. Intuitively, group signature schemes in the dynamic setting allow to separate the joining process (i.e., the issuance) from the operation of signing a message (i.e., the proof). Based on a group signature scheme we may build an anonymous credential system by adding two concepts, namely, the ability to certify attributes and to have several issuing entities. The proposal by Camenisch and Lysyanskaya [37] follows the latter design principle and is implemented as Identity Mixer (*Idemix*) credential system [98].

The most significant difference of the two approaches results from the fact that a credential created by the group signature approach can be shown multiple times without those interactions becoming linkable. Conversely, the concept of blind signatures only guarantees the unlinkability between the issuance and one proof protocol execution. Using the envelope analogy that illustrates blind signatures, we can compare multiple interactions in a proof protocol as showing the same message to several entities. Those entities can easily see that the message originates from the same entity. Similarly, interacting with several RPs results in those interactions becoming linkable. This can be considered a feature for certain scenarios such as electronic cash (e-cash) [49], where multiple use of one token is not permitted. However, in situations where we want to use a credential several times, such as an eID or an access token to a building, multiple unlinkable interactions with a verifier are more desirable. We may mitigate the issue by creating a set of credentials in one issuance interaction, but this results in additional bandwidth and memory requirements. If we envision a credential to be hosted on a resource-constrained device (e.g., a smart card), those drawbacks may make it infeasible to deploy a system based on blind signatures. The separation of issuance and proof protocol is the basis for an anonymous credential system but only an extensive feature set truly allow the paradigm shift from current to data-minimizing authentication.

2.2.3 Protocols

An anonymous credential system builds on two interactive protocols: the issuance and the proof protocol. First, an entity with the intention of obtaining a credential engages in the *issuance protocol* with an IdP. If the protocol is successful, the IdP will provide input such that the requesting entity (i.e., the recipient) can create a credential. The goal of the IdP is to determine the attribute values corresponding to its communication partner, because it will assert their correspondence by issuing

a credential. To this end, the parties exchange information that may partially be certified and, upon successful termination of the protocol, the recipient obtains the desired credential and becomes holder of the credential.

Second, in the *proof protocol*, a holder of a credential interacts with an RP to convince the latter that it holds a credential issued by a given IdP. In a proof of possession, the minimal version of the proof protocol, the verifier does not learn more than the fact that the prover owns a credential issued by a specific IdP. In fact, the proof may even hide the specific IdP and prove the credential to be issued by an entity within a set of IdPs [54]. However, the proof protocol also allows for further information contained in the credential to be revealed to the RP. More specifically, the information released may consist of a subset of the attribute values or even of a mere proof of a predicate over an attribute value. In the following we elaborate on extensions to the two basic protocols and we show the possibilities to include information in a privacy-friendly way into credentials and release different subsets of the certified information towards RPs.

Issuance Protocol Extensions

The main goal of the issuance protocol is the exchange of attribute values that will be certified in a credential and which will be provided to the credential owner. A naive approach to achieve this goal is to simply transfer the values a recipient wants to have certified from the recipient to the IdP. However, this approach does not meet possible sub-goals of the involved parties. We discuss here the situations where (1) a recipient does not want to reveal the attribute values to be certified, (2) an IdP requires a certification of some attribute values, and (3) the credential values can be updated. We describe the general process of issuing an anonymous credential and highlight the extension points realizing the additional goals of both parties. Note that not all anonymous credential system implementations support all of the extensions.

As preparation for the issuance protocol, the IdP and the recipient of the credential must agree on a set of attributes that should be included in the credential. This is usually a unilateral decision by the IdP and the recipient only has the option of accepting the proposed attribute set or not requesting a credential at all. However, the IdP may allow the inclusion of attributes without learning their values, which corresponds to the mentioned goal of the recipient for privacy of its attribute values. Such methodology bears the risk for the IdP that the credential will contain wrong information, which may affect the IdP's reputation. We can distinguish two cases in which the IdP may accept to include values in a credential without getting to know them: (1) the correctness of the attribute value is relevant to the credential recipient in order for the credential to be used, and (2) the IdP can verify the value. In the first case, the IdP relies on the interest of its communication partner to supply correct information. The rationale is that there is little value for a user in getting an attribute value, for example, an email address or username, certified if she does not control the corresponding account.

For the second case, the IdP requests a certification of the attribute value to be enclosed in the credential. Interestingly, even though the two goals of verifying a value and not learning it seem to be conflicting, they can be realized at the same time by using anonymous credentials. To this end, the parties first engage in a proof protocol to exchange the certification information before including the certified values into the newly issued credential. We denote this combination of a proof and issuance protocol as the *extended issuance protocol*.

The issuance protocol tends to be costly, for example, in terms of availability of the involved parties because it is an interactive protocol, or because it may require special security precautions such as the physical presence of the recipient of a credential at the premises of the IdP. As a consequence, re-running this protocol is undesirable and the possibility of updating credential values non-interactively, as proposed by Camenisch, Kohlweiss and Soriente [36], becomes attractive. In their approach, the IdP defines which set of attributes can be updated, where only attributes whose values are known to the IdP may be part of this set. In addition, IdP and recipient agree on a location where the former will deposit update information as well as on a time schedule or event upon which the updates will be issued. The update process then requires both parties to retain information already when engaging in the issuance protocol. This retained information allows the IdP to create the update information. The credential holder uses the retained information from the issuance protocol, together with the update information from the IdP, to update the credential.

A challenge in updating a credential for the IdP may be to learn the new values to be certified. In the simplest case, this information is already known by the issuer. For instance, this is the case for an expiration date included in a credential. We can imagine more elaborate cases where the credential owner needs to provide the updated credential value to the IdP, which makes the update process interactive. To complete the update process, the IdP computes the update information that will allow the holder of a credential to update her credential, and deposits it at the pre-agreed location. Note that an interactive credential update process, due to the necessity of communicating the new credential values, may still be sensible, for example, if the credential resides on a tamper-resistant device and a new issuance process would involve additional hardware costs.

Proof Protocol Extensions

The basic proof protocol is run between the owner of a credential and an RP. It conveys very little information to the RP, namely, it allows the RP to verify that its communication partner owns a credential issued by a specific IdP. Since the credential holder issues a proof attesting to this fact, we also name her *prover* in the context of the proof protocol. We can imagine several extensions to the basic protocol. As in the description of the extensions to the issuance protocol, we outline the concepts and do not focus on a particular system or even

implementation. As mentioned before, usually companies or organisations assume the role of the RP (or verifier) and individuals (also called users) that of a prover.

Similar to certificates, credentials should allow their owner to disclose attribute values. Conversely to a certificate, an anonymous credential has the capability to disclose attributes selectively. For example, a credential owner with a credential certifying her first name, last name, address, and birth date may only release the birth date in a transaction that requires her age to be verified. Assuming that a credential contains several personally identifiable attributes, this *selective attribute disclosure* benefits the owner in her attempt to minimize the released data to the required minimum for a specific transaction. Determining such minimal set of attributes is a research question out of the scope of this thesis. Selective attribute disclosure allows a person having a comprehensive credential certifying many personally identifiable attributes to disclose a subset, which makes it attractive for eID scenarios.

In addition to selective disclosure, the holder of an anonymous credential can prove properties about the attribute values contained in a credential. Such *property proofs* include equality and inequality relations [28, 93] (i.e., the 'less than' and 'greater than' relation) of attribute values, or set membership proofs for finite-set attributes [33]. Using again the example where age verification is required by an RP, we can see that in such case the mere proof of the birth date being in a certain range is sufficient. The release of the exact date would transfer more information than strictly necessary. Property proofs become particularly interesting when several credentials are involved in a single proof protocol. Thereby, a credential holder can, for example, prove that the last name encoded in two different credentials bears the same value. Consequently, the combination of selective disclosure and property proofs about attribute values have huge potential for minimizing the data disclosed by a prover to an RP.

When seeking to minimize the information released to RPs, we have to acknowledge that a number of services require to recognise returning users. To this end, anonymous credentials allow a prover to establish a *pseudonym* with an RP in the context of the proof protocol. Such pseudonym is similar to a public key in a standard PKI but its scope is limited, that is, the same user may relate to a pseudonym at her discretion. Thereby, upon a future visit of an RP with whom a credential owner has established a pseudonym, she can prove knowledge of the same pseudonym again and relate to her actions within the previously established context. Note that a credential owner may generate an arbitrary number of pseudonyms. We speak of a (regular) pseudonym if an entity can have an arbitrary number of pseudonyms with one communication partner. However, an RP may want to assure that each entity can only establish one pseudonym for a specified scope, for example, in case it wants to guarantee that each entity only establishes one account. We denote pseudonyms that limit each credential holder to exactly one pseudonym per scope as *domain pseudonyms*. Using cryptographic committment schemes, such uniqueness can be enforced by an RP.

2.2.4 Disadvantages of Data Minimization

One benefit of data-minimizing authentication is the privacy that participating people maintain since they only prove statements in an unlinkable manner instead of leaking lots of information or even unique identifiers that allow a verifying entity to track each user. This privacy protection comes at the price of making the prosecution of misbehaving entities more difficult. In this section we focus on the methods that allow for accountable transactions, thus, withdrawal or revocation of the anonymity of a misbehaving entity. Furthermore, we investigate in the revocation of credentials, that is, withdrawal of the rights associated with a credential, and we discuss possibilities of limiting the misuse potential such as a user sharing her credential with a friend or even selling it for profit. We account for the different requirements that exist in this space (e.g., a government-issued eID and an online newspaper credential have different requirements).

Accountable Transactions

There are many scenarios that generally require little information, but may need additional information conditional on the occurrence of an event. For instance, a person renting a car may do so only proving that she owns a driver's license as well as that she paid for the rental car. In the case of an accident or the car not being returned, the car rental agency has a legitimate need for more information exposing the misbehaving entity and allowing the agency to claim a refund for the damage caused. The naive solution would be to provide the information in any case, which results in all well-behaving entities revealing more information than necessary because of the rare case where something goes wrong.

Verifiable encryption [4, 42] allows a prover to release attributes on behalf of a TTP during the proof protocol. That is, it provides a solution for releasing additional information conditional on some event. Specifically, if the specified event occurs the information is released but as long as the entity behaves well, its communication partner does not learn the information. More concretely, during the proof protocol, the credential holder can provide a verifiable encryption on behalf of a mutually trusted entity (i.e., the TTP) to the RP. The entities must agree on the condition under which the encrypted information is to be revealed and on the TTP they want to collaborate with. The RP will specify the attribute values that are to be included in the verifiable encryption and he can verify that those values are equal to the certified values. The latter could be available in the form of an anonymous credential. In case the specified condition is met, the RP will request the decryption of the verifiable encryption from the TTP, so it can take appropriate actions. The TTP is needed as the entity judging whether the condition is met and, in case it is, to decrypt the verifiable encryption since the RP should not be trusted to adhere to the condition. Consequently, verifiable encryption provides a mechanism to allow for accountable transactions in an anonymous credential system.

Credential Revocation

Revocation of a certification token is a key feature for any PKI because it allows the system to recover from situations such as a lost secret key, or a wrongly issued token. While revocation of certificates benefits from the unique identifier encoded into each certificate, such identifier is not present in an anonymous credential system. Still, there are several proposals for revoking credentials in a privacy-enhanced PKI [27, 30, 35, 85, 87]. We briefly discuss the main ideas and concepts of those proposals as well as the general settings for revocation.

Let us start with the revocation strategies that we can distinguish. First, *global* revocation corresponds to the scenario where a credential is made globally invalid. For example, this method is useful in case the IdP issued a credential by mistake or he wants to revoke all privileges associated with a credential. Second, in case a credential only needs to be invalidated for a specific scope (e.g., at one RP), *local* revocation is used. Both scenarios may use either a *whitelist* or a *blacklist* approach. The former denotes the method of listing all valid credentials, the latter corresponds to the situation of listing the revoked credentials. For instance, Certificate Revocation Lists (CRL) [53] used with X.509 certificates employ the blacklisting approach and can be used for global or local revocation. A similar approach would also work for anonymous credentials where the IdP would include a serial number as an attribute into each credential and publish a whitelist containing signatures on serial numbers of valid credentials. A credential owner then creates a proof that her serial number is contained in the list using the signature of the IdP. A further mechanism for X.509 certificates uses an online revocation authority that verifies each request against the revoked certificates [97]. Using verifiable encryption, a similar mechanism can be implemented for anonymous credentials again using a serial number. This approach, however, makes user transactions linkable for the revocation authority. There are more elaborate ideas, for example, using dynamic accumulators [35, 40], or the sequential ordering of signatures on serial numbers [85], which improve on the efficiency of the revocation solution. An appropriate revocation strategy depends on the number of credentials and the scenario in which those credentials are used. We refer to Lapon et al. [75] for an in-depth analysis of revocation strategies and their performance.

Note that an alternative view on credential revocation implies that the credential update mechanism (see Section 2.2.3) may be used for implementing short-lived credentials. Instead of revoking credentials of misbehaving entities, the IdP could not issue credential updates to those entities, which causes their credential to become invalid. The applicability of this approach heavily depends on the scenario, that is, for credentials that must be revoked timely it seems to be less applicable. Comparing it to traditional revocation mechanisms shows that the schedule of updating the CRL corresponds to the schedule necessary for credential updates.

Usage control

A concern arising from the privacy provided by anonymous credentials is the loss in controlling how and where they are used. More concretely, illegitimate use of credentials may be harder to detect since a simple proof of possession does not leak information that allows for detecting misbehaving entities. Let us consider a situation where the online version of a newspaper can be accessed using an anonymous credential. We assume that the newspaper agency provides an anonymous credential to its subscribers, where the credential allows them to access the newspaper's online content in a completely anonymous manner. If the credential is not bound to tamper-resistant hardware (e.g., a smart card), a subscriber can share her credential with a friend or even sell it for profit. The newspaper agency has an interest in detecting such re-use of the same credential by several entities but as the entities only issue a proof of possession of a legitimate credential, it simply cannot. There are several options that an RP, in our example the newspaper agency, can use to mitigate this problem: limited spending [34], device binding, or all-or-nothing shareability [39].

First, *limited spending* allows the credential issuer to limit the number of times a credential can be anonymously used within a certain scope. For example, the scope can be defined as a time span of one day and the credential can be used up to ten times during this time span. In the case of the newspaper agency such approach would limit the possibility of sharing the credential on a large scale without making it impossible to share it with a friend. Second, credentials that are bound to a tamper-resistant device make use of *device binding*. The tamper-resistance of the device hosting the credential makes the latter hard to copy, so it cannot easily be shared. Third, *all-or-nothing shareability* implements a binding among different credentials. The general assumption is that each user has some valuable credential, for example, an eID, which motivates her not to share any credential because sharing one credential implies sharing them all.

2.2.5 Implementation Aspects

Let us detail the general concepts related to implementing an anonymous credential system. In this section we focus on the *Idemix* anonymous credential system [98] as it provides an extensive feature set and it allows for multiple unlinkable proofs using one credential. Note that *Idemix* implements all-or-nothing usage control, which is reflected in the general setup.

Setup

The parties interacting in a protocol run must agree on general system parameters that define the security-relevant aspects of the system. More concretely, this includes the bit length of all relevant parameters as well as the group parameters, for example, the generator, moduli and order of the groups used within the system. In practice, this can be achieved by distributing the parameters together with the

application code or uploading them, or by authentically communicating them to the communication partner.

The setup procedure for IdPs and TTPs consists of generating public key pairs, creating a specification of the services they offer and publishing this specification as well as the public key. As an example, an issuer runs the issuer key generation and publishes the structure(s) of the credential(s) it is willing to issue together with her public key. Each entity willing to become a credential owner needs to choose her *master secret key* based on the group parameters of the system. This secret will be embedded as an attribute into all her credentials, which implements the all-or-nothing sharing prevention approach. Note that there is no enforcement mechanism to guarantee that a user only has one master secret key but an IdP can enforce that a newly issued credential uses the same master secret key as a credential the user already owns. The master secret key is also used to derive pseudonyms and domain pseudonyms, which are similar to public keys in a standard PKI. All the pseudonyms of an entity are unlinkable unless the user proves that they are based on the same master secret key.

Our Device Binding Implementation

We outlined in Section 2.2.4 that tamper-resistant devices may be used as a method to protect anonymous credentials from being illegitimately shared. Especially for the scenario of eID cards such an approach seems reasonable since they potentially have far-reaching authorisations linked to them. However, governments looking for an eID implementation have a concise set of requirements that a candidate system has to comply with. Implementation attempts on standard smart cards [10, 15] resulted in computation times that were far away from the expectations of the eID community. Especially, the mentioned implementations attained transaction times that are magnitudes larger than the extrapolated computation times put forth by Danes [56]. Furthermore, all of these results revert to a situation where the tamper-resistant device computes a proof of possession by outsourcing computation to the terminal.

In our paper [17] (see page 103) we describe the challenges that have to be solved in order to implement anonymous credentials on a severely resource-restricted device. In particular, we propose to use a small stack size, reduce write operations to EEPROM memory, and implement a method for generating pseudo-random numbers to reduce memory requirements in favour of additional computation operations. With those design principles we managed to implement anonymous credentials on a standard Java Card that computes a proof of possession without computational support from the terminal. Namely, we realize a variant of the Direct Anonymous Attestation (DAA) protocol [32] and compute an entire proof of knowledge on the tamper-resistant device. To get the performance in the proof protocol as specified by the eID community, we access the crypto co-processor using the RSA encryption interface. More concretely, we execute modular exponentiations by setting the RSA key to the value of the exponent

for each exponentiation and handle the base and modulus accordingly. There is one limitation arising from this approach. In a standard RSA operation of a Java Card using version 2.2 [102], the RSA key does not change frequently. Therefore it resides in EEPROM, which is persistent and protects its values in case of a power outage but it is very time consuming to update. With our approach, the exponent is changed frequently and updating the EEPROM is not an option. We thus need exponents to be stored in transient memory. The Java Card 3.0 standard [90] alleviates this issue by allowing RSA keys to be stored in transient memory. Using the RSA interface provides a dramatic speedup for exponentiations, which we also use for computing modular multiplications, that is, we calculate a multiplication of a and b modulo n as $ab \pmod n = ((a+b)^2 - a^2 - b^2)/2 \pmod n$. This method substantially reduces the computation time of a multiplication compared with computation on the Java Card application layer.

Our main result is the implementation of the *Idemix* anonymous credential system on a standard Java Card achieving a computation time for a proof protocol of less than ten seconds (using a 1536-bit RSA modulus), which fulfills eID specifications in terms of hardware, processing time, and security level. Indeed, our results are confirmed independently by Sterckx et al. [101]. In addition, we show that it is even possible to improve on the user experience by pre-computing several values. Specifically, we imagine a user arriving at a terminal where she would use her smart card to prove a statement as displayed by the terminal. She would presumably enter the smart card into the terminal before she has read and agreed with the policy displayed by the terminal. During the time in which the user assesses the policy, the smart card could already compute values that are independent of the policy. Using this method, our implementation achieves a (user-noticeable) computation time of approximately 2.6 seconds for a proof of possession of an anonymous credential using a modulus length of 1536 bits (see page 103 for details on system parameters and timings). Additional contributions are the minimal policy format that allows a terminal to communicate the requirements of a proof to a smart card, the integration of the smart card into the *Idemix* credential system and the measurements of computation times of the most important mathematical operations for implementing an anonymous credential system.

Our Idemix System Architecture

The requirements for an implementation of an anonymous credential system on a general-purpose computing system [43] differ significantly from the requirements of an implementation on a resource-constrained device such as a smart card. In [16] (see page 131) we propose a modular architecture that makes use of independent components and of specification languages that define the format of the exchanged messages as well as their combination on a local system. We provide an overview of our contributions by outlining the individual elements we propose and by showing

how the specification languages combine them to implement the functionality of a full-fledged anonymous credential system.

Components. The most important component is a credential, which consists of a number of attribute-value pairs as well as the cryptographic evidence of the issuer. A credential also references the relevant system parameters to make sure that different anonymous credential system configurations may co-exist. The further components are the master secret (including pseudonyms and domain pseudonyms), commitments, representations, verifiable encryptions, and messages. The master secret implements the all-or-nothing sharing protection mechanism as introduced in Section 2.2.4 and it is used to generate pseudonyms as well as domain pseudonyms. As a consequence, all pseudonyms are coupled to the master secret. Commitments can be used to transfer information gained in a protocol run to another protocol. For example, this component allows for the implementation of the extended issuing protocol in which a value certified in one credential can be carried over from a proof protocol run into an issuance protocol run. Specifically, *Idemix* uses the Damgård-Fujisaki-Okamoto commitment scheme [55], which is essentially a Pedersen commitment scheme [91] in groups of unknown order, to include an attribute value that will remain unknown to the IdP into the issuance protocol. A generalization of commitments are representations. Unlike commitments they allow for a specification of their bases. Therefore, representations are a very flexible element that, for example, allow a credential owner to prove that an attribute value is larger than another one without releasing either of the values. In the standard case, inequality relations [28, 93] can only be proven if one of the compared values is known to the verifier. The verifiable encryption component also enables to add accountability as it is the basic element for releasing information to a TTP. Finally, messages allow to specify a context in which the proof protocol is carried out, which is signed by the prover when interacting in a proof protocol. For instance, a common use of the message could be to tie a proof to specific terms and conditions which can be reviewed in case of a dispute between the prover and the verifier.

Specification Languages. We propose to steer the usage and combination of the proposed components by three specification languages, namely, the credential structure specification, the issuance specification, and the proof specification. The *credential structure* is used to define the attributes of a credential, which includes their name and whether or not they are known to the IdP. It plays an important role in the issuance protocol in showing which attributes a credential will certify once it will be issued. Moreover, it provides information to the recipient to decide whether or not to request a credential. In the proof protocol it provides the vital information on how to compute the verification since the credential itself must never be communicated to the verifier. The *issuance specification* in the general case only refers to the credential structure, which sufficiently defines the issuance protocol. However, in a case where information such as a commitment of a previous protocol run (e.g., a proof protocol) is to be incorporated into the issuance process, the issuance specification references such additional information.

As the proof protocol is very flexible and provides several extensions (see Section 2.2.3), the *proof specification* is the most versatile of the proposed languages. It enables a prover to precisely specify the contents of a proof. That is, it refers to the credential structures of the credentials involved in a proof, specifies which attribute values are disclosed, what properties are proved about undisclosed attributes, as well as what relations are fulfilled by the data encoded in further components such as commitments or representations. The proofs about attribute values comprise equality among attribute values, inequality relations, or set membership of finite-set attributes (e.g., using the Camenisch-Groß encoding [33]). Finally, the proof specification may refer to the pseudonym or domain pseudonym under which the prover releases the information, it may contain a message signed together with the proof, or it may refer to a verifiable encryption element released together with the proof.

We implemented the proposed architecture and released it under a special open source license as *Idemix* credential system [98]. At the core of the implementation is a signature scheme that supports the separation between issuance and proof protocol as well as several interactions in a proof protocol as outlined in Section 2.2.2. Our implementation uses the Camenisch-Lysyanskaya signature scheme [37] as it provides the richest set of features, for example, limited spending [34], efficient attribute encoding [33], credential updates [36], or verifiable encryption [42]. We refer to Table 1 on page 164 for an overview over the features that are currently implemented.

Our Approach to Ecosystem Integration

There exist already a number of technologies using the abstract concept of credentials, that is, a set of attributes with corresponding values for a specific entity. Examples of such technologies are X.509 [53], OpenID [52], SAML [88], LDAP [113], or anonymous credentials [31, 98]. To establish interoperability among such technologies we need (1) a policy language to communicate the requirements of an RP to a credential holder, (2) an abstract claim language to specify the contents of a proof, and (3) a technology-specific language for generating a proof. The first and last requirement can be met using the *Credential-based Authentication Requirements Language* (CARL) policy language proposed by Camenisch et al. [41] and the *Idemix* proof specification [16] language, respectively (see Section 7). Note that we cannot use standard languages for secure attribute transfer or specifying access control requirements (e.g., SAML [88], XACML [89]) as they do not provide primitives supporting data-minimizing authentication, which is an orthogonal problem tackled by Ardagna et al. [2].

In our paper [19] (see page 153), we propose a claim language based on CARL. This fills the gap between the CARL policy language and the *Idemix* proof specification as pointed out before. Demonstration of the mapping of general authentication concepts used in CARL to the specification language of a cryptographic library is an important step as (1) it contributes towards explaining

the concepts of data-minimizing authentication to a broader audience, and (2) it lowers the entry barrier for using the *Idemix* credential system implementation. As a further contribution, we extend the *Idemix* proof specification to make it more expressive and we outline how to use it appropriately to implement high-level functionalities. For instance, we show how arithmetic relations over several attribute values can be proven using the commitment and representation component of the *Idemix* proof specification. This relieves system architects from understanding the cryptographic tools used in the *Idemix* credential system and enables them to implement authentication systems with only the abstract functionality in mind. Finally, we provide an implementation of a complete data-minimizing authentication framework, with an architecture that allows for various credential technologies to be used. We add support for the *Idemix* credential system by implementing a technology-specific connector, where we selected *Idemix* for its broad feature-set in support of data-minimizing authentication.

A key component of our implementation is the user interface (UI) since presenting the choices that a user may have can be very complex. For instance, an RP may allow a user to access her service with a service-specific credential or a government credential and prove appropriate properties. Assuming that a user can fulfill both of the options, the UI has to convey this choice as well as their consequences to the user. We consider the possibilities to implement a suitable UI following the suggestions of Bichsel et al. [20] and Wästlund et al. [110] and present the first functional UI implementation. Note that due to counter-intuitive concepts and lack of experience of an average user with anonymous credentials, we assume that it will require a learning curve until users fully grasp the concepts.

Finally, we compare the data-minimization possibilities offered by different credential technologies. We do not limit ourselves to the situation where a user obtains a credential and can prove properties independently from the issuer but also include situations in which an entity requests certified attributes after they have been requested by the RP. This is currently the most widely deployed approach with protocols such as OpenID and it can offer far-reaching functionality, assuming the existence of attribute exchange protocols supporting statements over attributes. However, compared to anonymous credentials, those protocols suffer from the IdP being informed about each transaction. This requires a very high level of trust in the IdP and the controversy of data retention efforts in Europe shows that such high levels of trust may not be acceptable [65, 86].

Authentication Environment

Data-minimizing authentication systems build on public-key cryptography, which couples it with a strong requirement. Namely, public keys must be authentically retrieved before they can be used for any purpose such as signature verification or encryption. Today, probably the most prominent use case of public-key cryptography are X.509 certificates used for SSL/TLS encryption for e-business services. In this context, companies own certificates and use them for browser-based interactions. Authentic key retrieval is tackled using a PKI in combination with distributing a set of "trustworthy" public keys with the browser software. While this approach has severe shortcomings [64, 104], it is used for establishing (server-side) authentication and message confidentiality in e-commerce applications. However, in a configuration where each person possesses a set of certified tokens that can not only be used in browser-based transactions, the current approach is clearly not sufficient. Moreover, while a large-scale company can be expected to implement appropriate protection and update procedures for its secret keys, for a normal user such assumption is too strong. Consequently, we need appropriate strategies to tackle the key life-cycle management for a broad audience but we believe that the problem is bigger, we need to understand what characterizes the digital representation of people or organizations.

A main goal of data-minimizing authentication is to make transactions unlinkable, that is, to prevent service providers from gathering excessive information on their users. On the other hand, the success of ESNs shows that people want to communicate with each other using the Internet. For this purpose, they need to be able to recognize their friends also in the digital domain. This chapter elaborates on the aspects of entities representing themselves online and on how they may be recognised. Note that the while we use the term "identity" in the following sections, we want to point out that identity is an elusive concept and

researchers from several fields have contributed towards defining what constitutes a person's identity. In this work, we do not aim at contributing to such definition, but we want to provide a conceptual framework that will assist us in identifying similarities and differences between the recognition of persons and organizations in the offline and online world. Therefore, we consider a simplified view of identities where an identity consists of attribute-value pairs. In particular, we define the communication of an entity to be an attribute of the latter, thus identity is strictly dependent on communication.

In Section 3.1 we sketch the properties of identities that manifest themselves in the digital domain. This discussion builds on our concepts introduced in Bichsel et al. [21] (see page 181). Through a comparison between authentication in the offline and online world, we illustrate the requirements for successful authentication of individuals. Furthermore, in Section 3.2 we discuss our method for bootstrapping trust in the digital domain [22] (see page 193). The opportunity of our approach lies mainly in using the efforts of people in maintaining a network of friends in Electronic Social Networks (ESNs) to bootstrap security properties.

3.1 Digital Identities

In this section we present the fundamental identity concepts we use when describing authentication situations. We start with a summary of different views on digital identity concepts relating to authentication in Section 3.1.1. The entities, that is, *persons* and *organizations*, are characterized by attributes, which comprise an entity's name, date of birth (or establishment) just as well as the communication they participate in. Therefore, we need to investigate the fundamental role that communication between *entities* plays in our definition of an identity. To this end, we introduce characteristics of communication in Section 3.1.2. In Section 3.1.3 we discuss the definition of our identity concepts, which we illustrate in the offline and online world in Section 3.1.4. Finally, we present our mechanisms to recognize "friends" in a digital context [21] (see page 181).

3.1.1 Views on Digital Identity

In the physical world, a person has some characteristics or attributes (e.g., name, hair color and length, or facial features) that enable others to identify her. The same happens with entities like organizations: some characteristics (e.g., name, date of establishment, address) define them and allow others to refer to them. Recognition of people or organizations is tightly bound to those identifying attributes. A question, then, rises on the role of these attributes in the definition of an identity, and it is legitimate to wonder whether the same concepts and mechanisms work on the Internet. We will not get into long philosophical debates on essential characteristics of entities [79], or attempt to classify them based on whether they change over time [111]. We think that such distinctions are not

very significant in the usual practice of identity management: in our view, a characteristic or attribute c can enable us to identify an entity E, as long as it has not changed since the encounter when we registered c as belonging to E. In addition, it must be uniquely characterizing E in the multitude of other entities from which we are singling out E.

Let us summarize on some related efforts in establishing an understanding of digital identity. Cameron [44] lists identity aspects that he calls "laws of identity" instead of a unified definition. This work aims at describing the necessity and mechanisms required to build an Internet identity layer. Windley [111] also considers such an identity infrastructure a necessity for successful service delivery over the Internet and provides several pointers to existing proposals and standards. In the context of distributed system research, standards such as OpenID [94] propose to support the expression of identity attributes for authentication and access control purposes. In addition, an increasing number of these works, see for instance Ardagna et al. [3], consider privacy issues as fundamental.

3.1.2 Characteristics of Communication

In general, communication serves the purpose of exchanging information and can take place in several different forms. Example of ways in which people communicate are: talking in person, sending an e-mail, making a phone call, leaving a voicemail, having a video chat, or writing a letter. All these possibilities can be exploited for the same objective, but they come with different characteristics, which affect the way communicating entities perceive each other's attributes. In our view, the *attributes* of a person E are comprised of not only personally identifiable information such as her first name, last name, or date of birth, but also all impressions and attitudes transferred during communication processes she is involved in.

Proposition 1 (Attribute)**.** Information that is linked to an entity E and helps other entities in distinguishing E from a set of entities.

Communication can be distinguished into *synchronous* and *asynchronous*, depending on whether it takes place in simultaneous presence of all communicating entities or not, respectively. Synchronous communication is probably the most common way for people to know and get to know each other better: for example, during a conversation at a bar, at a work meeting, or talking on the phone. On the contrary, asynchronous communication does not require the presence of the communicating parties at the same time. It is implemented by one entity persisting a set of attributes that gets observed by another entity at a later time. Examples of this type of communication are letters, memos, and photos. Synchronous communication allows for faster exchange of information among participants and usually provides a broader variety of attributes, so that identities are presented and perceived in a more effective way. For instance, we can think of how much faster we can get to know a person thanks to several phone conversations and dates

rather than by exchanging letters as pen-friends. On the other hand, asynchronous communication has its merits in preservation of messages since it relies on objects that can store information usually for a much longer time than the faulty memory of a human being. For instance, think of how we recall details of an event because of photos, rather than our memories.

Another important distinction between types of communication depends on whether it takes place *offline* or *online*. The increase of computational power, availability of computing devices, and the introduction of the Internet provide a number of new communication possibilities. In the area of synchronous communication it started with scant text-based chats and evolved to fully-fledged personal conversations thanks to the likes of Skype. However, asynchronous communication plays an even more important role in the Internet era, as shown by the unprecedented success of electronic social networks or blogs like Facebook, Twitter, or Tumblr. The service of those Web sites is to enable entities to create a persistent description of their attributes, including physical features, personal taste and opinions, for other users. The immense impact of this kind of service shows that a new way to communicate with people and organizations has been established. We intend to investigate how such communication can be enhanced and improved, therefore, we first need to analyze its foundations by means of an adequate conceptual framework.

3.1.3 Identity Concepts

Let us begin with the basic concepts that characterize identity and that we illustrate using examples from the offline world. As said before, in our view, identity is strictly dependent on communication, and the simplest and most common form in which offline communication can take place is a physical meeting between two persons or a telephone conversation. Whether intentionally or not, a person E shows some of her attributes during communication, enabling the other person to associate them with E, and this association is the fundamental component of what it means to know, remember, or recognize E. Thus, we intuitively define the *identity* of a person E as the set of all attributes E communicates in any way to any other entity.

The same basic principle of associating attributes with a communication partner for recognizing her holds in more complex examples of communication. For instance, we can think of scenarios where more than two parties are involved in a message exchange, or where one of the parties is not a person but an organization. In fact, although a person cannot communicate with an organization in the same way she would communicate with another person, an organization is also characterized by attributes that people use to identify and distinguish it from others. For organizations, the presentation of such attributes to other entities is often delegated to a specific person, usually from the public relations office of the organization.

In general, during the communication of two entities E and U, some of the attributes that contribute to the definition of the identity of E are shown to U.

Proposition 2 (Facet)**.** We call a *facet* of the identity of an entity E a set of attributes which describe E, and are presented by E in a well-defined communication context, and we symbolize it with f^E.

This view closely relates to how Berg and Leenes [109] imagine to realize audience segregation in ESNs. It is a very general view, according to which whatever a person looks like, says, or does in front of other people can be seen as defining part of her identity: our appearance, our opinions, our actions indeed (partially) define our identity. We indicate with $\mathcal{F}^E$ the set of all facets ($\{f_i^E\}$) that entity E is presenting. $\mathcal{F}^E$ is a dynamic set, in that the attributes that E shows vary continuously. Temporal issues are out of the scope of this work, but let us point out again the role played by communication in the definition of identity, so that it is not fundamental whether $\mathcal{F}^E$ changes quickly or remains constant, but rather its effects on the other entities that perceive, record, and remember a subset of $\mathcal{F}^E$. As a matter of fact, not all attributes made available by E are gathered and retained by her communication partner U. Rather, people can be distracted and miss some particulars, or forget some information due to an imperfect memory and the passing of time. Hence, the result of the communication between E and U is not equivalent to a simple transmission of the information of the facet E is showing: we need to introduce the concept of a *perceived identity* of E by U.

Proposition 3 (Perceived Identity)**.** Entity U's *perceived identity* of entity E, indicated with $\mathcal{I}^U(E)$, is the set of all the information on E's identity that is kept in U's memory or stored in her memory devices.

Where E's facets contribute to her identity, a perceived identity is rather the impression that those facets have made on U, that is, it subsumes all the attributes U retains and associates with E. $\mathcal{I}^U(E)$ is a dynamic set as well, but it changes slower than $\mathcal{F}^E$, as only situations when E and U are communicating affect it. The reason for the slow change is that a person E can radically change the facet she represents towards other people from one day to another, but E's friends will not simply replace all memories that they have of E. Note that, while a synchronous communication that affects $\mathcal{I}^U(E)$ affects also $\mathcal{I}^E(U)$, asynchronous communication typically updates only either of the two.

Let us illustrate our concepts using an example of the facets of a person "John Doe" in Figure 3.1. It depicts the different facets John shows to his employer, the state he lives in, the legal authorities, or his biking friends. For example, the employer and the state both get to know the first name, last name, and the salary of John Doe. Apart from those entities, John does not want anybody else to know his salary, thus it is not part of John's facet with any other entity. The perceived identities do not directly show in the picture but let us point out that even though John communicates a single facet towards a group of people,

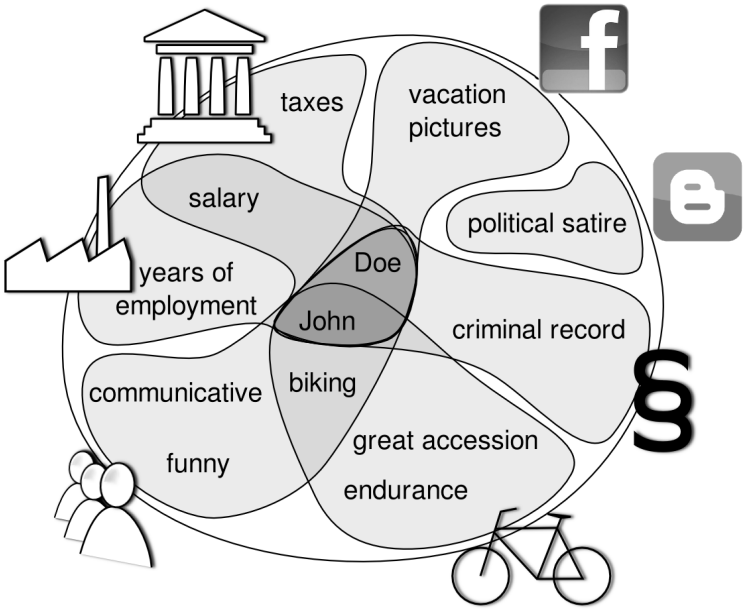

Figure 3.1: Facets of a person called "John Doe" depicted as the attributes the person shares with entities, such as (from top left, counterclockwise) the state, his employer, his friends, his cycling mates, and legal authorities. The picture includes digital facets of John, namely, a political blog and his Facebook profile.

the resulting perceived identities would all differ. This is also notable with his friends who remember different parts of a conversation and come to different conclusions. In addition, John clearly has different conversations with his friends that contribute to the diversity of the perceived identities. Another example where we see the differences between a facet and the resulting perceived identities is the company where John works: although John consistently shows his "professional facet", we can be certain that his boss, his closest colleague, and the human resources manager who interviewed him have all different perceptions of John. In fact, John has a different perceived identity with each individual within each group of people.

3.1.4 Comparing the Offline and Online World

Let us analyze our identity concepts in the context of the online world to be able to compare the situation of the offline and online world. As a person communicates with others and shows some of her characteristics in the offline world, so can a person describe some side of herself on a blog or a social network. A digital facet of

an entity E, f^E, is any set of attributes or information in a digital format referring to E, created and managed by E or an entity authorized by E, presented in a consistent way on a computer system. E's Facebook page or an email from E are examples of digital facets. The information provided by a digital facet, possibly in the form of text, pictures, or multimedia files, can be considered as the digital counterpart of a person's facet, that is, what she presents when people meet her in the physical world. Analogously, as entity U builds up a perceived identity of entity E by communicating with it in the real world, she can form a perceived digital identity of E (i.e., $\mathcal{I}^U(E)$) by checking some of E's digital facets from $\mathcal{F}^E$, the set of all data that has been published by E or by an authorized entity.

So far, we have focused on the similarities between the offline and online world when it comes to dealing with identity, but there are indeed important differences. Digital facets are scalable, that is, persistent, easily accessible, and potentially address a broad audience, as opposed to offline facets, which are typically volatile, and limited to a very restricted audience. In addition to the scalability of digital facets, their digital nature makes them easy to copy, which marks an important difference with the offline world where biometric, hard to copy features typically pertain to a facet. Further, ever improving search algorithms let the retrieval of digital information become more and more effective. In combination with storage becoming cheaper it causes information on the Internet virtually becoming unerasable, in other words, the *Internet does not forget*. This can be a desired feature in the case of a blog whose publisher seeks to convince people, spread ideas, or simply entertain. However, it may not be true for social networks, where users possibly publish personal information intended for a small group of very close friends (e.g., vacation pictures that are not meant to be shared with work colleagues and superiors).

Persistence and retrieval of digital facets play a significant role when entities change their attributes. It is normal that the interests and opinions of people change over time. However, since people's memories fade with time, in the offline world such changes hardly ever undermine the coherence of a perceived identity of a person. Similar, while a bad temper may be forgotten and forgiven in the offline world, online statements made in the far past have an impact on the present.

3.1.5 Recognition of Identities

In the offline world, we usually recognize the people we communicate and interact with on the basis of biometric properties such as facial features, voice, or handwriting. This happens automatically, and we only get aware of this process whenever a mismatch occurs, for example, when a person calls us and pretends to be someone we know, but their different voice gives the scam away. People's physical features change so slowly that they can be considered constant over a significant interval of time, where we abstract from cases where a person has plastic surgery. This allows a person U, for instance, meeting E to match the currently presented facet f_i^E with all stored perceived identities and recognize

E. In summary, in the offline world, the physical features of persons provide a fundamental support in the recognition process.

Contrary to the offline world, the online world lacks such direct contact, that is, it does not offer biometric features to compare against. In our paper [21] we illustrate the challenges that arise from such situation. We shed light on the security properties we want to achieve and compare attacks that may be mounted. One security property that is of interest to us is the authenticity of a facet. An observer should be able to determine whether or not a facet is published by the entity it represents. More concretely, we discuss the role of individual attributes in such authentication process and illustrate how (1) intrinsically authentic attributes, (2) attributes certified by an IdP, and (3) community-certified attributes are of use. Intrinsically authentic attributes, such as the writing style of a person, have the advantage that the preserve their binding to a person even when digitized. Therefore, such attributes are similar to biometric attributes in the offline world and allow a person to match against her perceived identity. Certified attributes use the trust of an observer U of a facet in an IdP to convince person U of the authenticity of the certified attributes. Finally, community-certified attributes use a similar principle but instead of a (trusted) IdP the certification is provided by a community of entities that attest to the authenticity of some attributes.

The digital nature of communication and the fact that nobody is tightly bound to her physical features anymore are not only limiting factors with respect to human communication. On the Internet, where the immediate recognition processes that take place in the offline world are not inevitable, *anonymity* and *pseudonymity* have become possible. We investigate how a person E can build up a digital facet for a fictional entity F. This includes mechanisms to manage such facet f^F, for example, we describe a mechanism to authenticate towards a service provider hosting f^F without the latter learning information on E. Furthermore, we show how to use verifiable encryption to pass on the control over f^F to a person E' without the host learning about this change.

3.2 Trust Management

Public-key cryptography requires a system enabling authentic retrieval of public keys, that is, a trust management infrastructure. For instance, before an entity E can use a public key pk, for example, for establishing a confidential connection to the owner of pk, it needs to verify that the owner of pk corresponds to the intended entity. Using the identity concepts introduced before, if E wants to communicate to U, she must verify that the candidate public key pk belongs to f^U. We illustrate current solutions to establish such match in Section 3.2.1.

The introduction of data-minimizing authentication increases each individual's exposure to public-key cryptography. As a consequence, a person not only needs the possibility to authentically retrieve public keys but also a set of algorithms for the management of her own (secret) keys. In Section 3.2.2 we discuss our

approach [22] of using ESNs for both, the management of personal keys as well as the trust management in further entities' public keys.

3.2.1 Public Key Authentication

The *public-key infrastructure* (PKI) paradigm uses a set of (trusted) entities who create certificates that bind the public key to a set of attributes for a given entity. Intuitively, such binding can be used to authenticate a public key pk of an entity E using (1) a certificate issued by an entity I attesting to the binding of E's attributes with pk, (2) an authentic copy of the public key of I, and (3) trust in I. For a more formal treatment of such authentication processes we refer to Maurer and Schmid [81].

We distinguish two implementations of a PKI, namely, a hierarchical and a distributed approach. First, the hierarchical approach is used today for confidential communication with service providers in browser-based transactions. Each service provider uses an X.509 [53] certificate from a certificate authority (CA) that is retrieved and verified in a user's browser. A significant issue is the authenticity verification of the public key of a CA. In the current deployment those keys are distributed together with browser software. The most prominent use of X.509 certificates happens in e-business applications, where a certificate is used to bind attributes including a URL to the public key of an entity. Such a certificate can be verified without the involvement of the CA, which allows her to remain offline after the issuance of a certificate. Current browsers support users in the verification of the certificates by providing warning messages if, for example, the certificate expired, or the certified URL does not match the connected. However, there is the issue of users not paying attention to the warnings [104] and as the certificates of CAs are distributed with the browser software, a successful attack on a certificate takes long until it can be fixed.

Second, a distributed approach to authenticate public keys is the Web of Trust (WoT) [114]. In a WoT each individual may issue a certificate on the public key of another entity, and thereby assert the correspondence between public key and (personal) attributes. Trust in such certificate is established through a path of already trusted entities. Here, the problem of initial trust relations is solved using physical verification of a witness such as an identity card or a driver's licence. The latter is costly and has proved not to be appealing to a broad audience. Consequently, we investigate on possibilities to innovate the process of establishing trust in a public key and corresponding attributes.

3.2.2 ESN-based Key and Trust Management

In our context the management of the keys of a person consists of two aspects: (1) the generation and update of her own key(s), and (2) the process of authentic key retrieval of public keys of other parties. In our paper [22] (see page 193), we describe algorithms to solve both problems using ESNs. The main idea behind

our approach, which has independently been hinted at by Hogben [70], is that people use ESNs to communicate with their peers on a daily basis. Thereby, they (implicitly) authenticate their friends and can discover fraudsters in the long term. Our methods uses this behavious to provide a low-overhead solution for bootstrapping trust in the digital domain. Even though this method is most interesting on the level of individuals, the presence of companies in ESNs may enable a transition from the current, hierarchical PKI to a distributed approach also for e-business applications.

In detail, we start our discussion with an ESN-based algorithm used for *trust assessment*. This algorithm is heavily inspired by processes happening in the offline world and integrates well into our digital identity concepts. That is, it details on how the matching between a perceived identity and a digital facet can be leveraged to finally reach the state of *trust declaration*. We propose to declare trust by issuing a certificate on the user's attributes including a public key. This WoT can be further used for *trust propagation* and also builds an integration point in case several ESNs implement our trust management proposals. The latter aspect is especially interesting as the trust relations build using an ESN need to be useful outside of the ESN. We propose to use the WoT infrastructure as a backend system and replace the current, manual signing process with our ESN-based approach. People using several ESN could sign keys from within all ESN and use the keys within their Web browser or email application. Therefore, we propose mechanisms that allow a user to finally easily authenticate public key information, which has a huge potential in the current e-business context.

Conclusion and Open Problems

In the process of working on this thesis, we have looked at various problems, found solutions, but also discovered new issues. In this chapter we discuss two aspects: First, in Section 4.1 we draw our conclusions on the results presented in this thesis; second, we elaborate on opportunities for further research in Section 4.2.

4.1 Conclusion

A main goal of our work has been to make data-minimizing authentication more practical. One aspect of the practicality of a technology lies in its readiness for deployment in a specific scenario. We envision one particularly interesting setup for deploying data-minimizing authentication to include a secure element such as a smart card. Our group signature scheme [18] features the shortest group signatures known to date and it comes with an extremely efficient signature generation algorithm. Both factors are very important when using a resource-constrained device for signature generation. The linear complexity in revoking the anonymity of signatures is acceptable to a computationally powerful service provider, especially since this is an operation that will presumably only rarely be used.

A smart card with group signature functionality would, for example, be of use in a corporate environment where employees could sign documents on behalf of the company or access the employer's premises. The limited set of features, however, caused us to shift focus and investigate in the possibility of having anonymous credentials implemented on a smart card. To this end, we were able to demonstrate a DAA-like protocol on a standard Java Card [17] that meets eID requirements. Clearly, anonymous credentials are most useful when

the implementation supports a more extensive set of features than our DAA version. In particular, we imagine the possibility to issue a proof stating that the credential holder is of age, for example, using an eID, to be an interesting scenario for using anonymous credentials. The reason being that service providers are legally required to verify this fact, for instance, before selling alcohol. The rapid development in smart card technology as well as further results, such as the U-Prove implementation of Mostowski and Vullers [83], suggest that implementing further functionality on smart cards is already feasible today.

Our modular architecture for *Idemix* [16] builds a first step in coping with the complexity of the extensive feature set of *Idemix*. On the one hand, it allows for an easier integration of partial implementations of the *Idemix* protocols such as our own Java Card implementation. Further, it offers extension possibilities such as the generalization of zero-knowledge proof computations. On the other hand, this helps us in educating developers on how the functionality of *Idemix* can be accessed, thus making the library easier to understand and use. We profit from our proposed API when we integrate *Idemix* into a more general entity authentication framework [19]. The resulting implementation is a framework that can be easily extended with further authentication technologies. We simplified such extensions by releasing our implementation under the Eclipse Public License (EPL), a popular open source license. In addition, by extending the SAML and XACML standards as proposed by Ardagna et al. [2], our implementation could be made fully standards-compliant.

Successful deployment of an authentication solution requires a precise understanding of the properties that should be achieved. In this light, our identity concepts [21] illustrate the characteristics of the implicit authentication in the offline world, they show the differences to the digital domain. Thereby, they manifest the shortcomings we have to overcome in the online world to attain authentication assurances as in the offline world. Using those concepts we can describe the goals of mechanisms such as our technique of bootstrapping trust in the absence of security assumptions [22]. Our use of ESNs makes clever use of current user behaviour to establish security properties in personal attributes and public keys. Usually, people rather see security as an obstacle and even accept a less secure system with better usability.

4.2 Open Problems

The road for a research result into people's daily lives is long and unpaved. In the case of data-minimizing authentication, some of the biggest challenges are not of technical nature but rather have a political and marketing component. In Section 4.2.1 we discuss general problems that need to be overcome before data-minimizing authentication can be deployed, before providing a list of issues that directly follow up on our results in Section 4.2.2.

4.2.1 General Challenges

Today, one category of service providers use business models that require extensive knowledge of personal information about users, another category collects this information for security or accountability reasons. While a company using models of the first category needs to profile its customers to create its revenue, it is a very important task to convince the latter category of the merits of data minimization leading to improvements in the strength of their authentication system and mitigating data quality issues. A persuasive aspect towards this goal is a *complete* demonstrator for a specific scenario, in particular featuring (1) an efficient implementation on state-of-the-art hardware, (2) a complete but minimal set of features with respect to the chosen use case, and (3) a well-tested User Interface (UI). In addition, we can assist the persuasion process with an analysis of the benefits, risks, and costs of data collection. However, it is an open question how reliable data can be gathered about problems such as: How many users indicate wrong information towards service providers, or what are the costs of a successful attack on a service provider (e.g., the Sony incident [8, 9, 107]).

Independent to the efforts in convincing service providers, we can foster the adoption of a new technology through standardization. Ardagna et al. [2] are following this trail of thought with their addition of data-minimization principles to the SAML and XACML standards. Another effort in a similar direction would be the specification of a technology-independent API for a basic set of features relevant to data-minimizing authentication. Such API would benefit the interoperability of U-Prove and *Idemix*, the most well-known anonymous credential systems to date. Further, it would improve on our architecture [16], which is specific to *Idemix*.

A further direction that contributes towards adoption of data minimization could be via regulation such as data privacy legislation (e.g., EU Data Protection Directive (Directive 95/46/EC)). Legislators have the obligation to help protect individuals from privacy invasion but they also have a requirement to set the grounds for a flourishing industry. Data-minimization has the potential to satisfy both goals that today are conflicting when it comes to requiring privacy-friendly service provisioning. Unfortunately, there are not many implementations (let alone products) that offer data-minimizing authentication, which makes legislators hesitant because it may force companies to adopt a specific technology, rather than a concept.

Apart from those aspects that contribute towards establishing data-minimizing authentication as a new authentication paradigm there are some technical challenges that are critical to its success. For example, the UI must be closely studied to ensure that a user can easily assess the information that will be exchanged in an authentication transaction. The research of Wästlund et al. [110] explores this direction but we also need to understand the learning process associated with this technology. This is particularly important as people are not familiar with the concept of a zero-knowledge proof in their current environment,

thus it is hard to convey to them the functionality that can be achieved through this technology. Furthermore, a problem related to the UI is that the information displayed on a computer screen may be (maliciously) modified. There exists technology (e.g., the TPM [108]) that may be employed to achieve this property for the screen of a general computing system. However, we need to consider that credentials may reside on a smart card that can be used with any terminal and we need appropriate support for transactions also in such situations. Finally, the current *Idemix* cryptographic library has not been verified to implement the functionality published as research papers. This is an important piece in making sure that the security proofs given for some construction does still hold with respect to to a specific piece of software.

4.2.2 Specific Issues

During the past four years we have looked at numerous challenges, however, usually solving one problem resulted in several others becoming apparent. Let us list a selection of interesting challenges we identified. We structure this list by the contribution that brought up the question.

Get Shorty with Group Signatures without Encryption

- Our group signature scheme [18] combines the group manager and the opener into one entity. This setting requires stronger trust of a group member in this more powerful entity, which we can see in the trust required to prove the individual security properties. In addition, the combined entity is a single point of failure and an interesting attack target. It is desirable to achieve weaker trust assumptions by separating those two entities as formalised by Bellare et al. [13].

- Our proposed group signature scheme does not come with a corresponding generic construction. An interesting question lies in finding such construction using the combination of security properties we propose. The hope would be that this leads to further (even more) efficient group signature schemes.

- Group signatures according to the definition of Bellare et al. [12] can be shown to imply encryption. By changing the anonymity notion, the same argument does not hold for our scheme. That is, the secret signing keys of group members cannot be made available to an attacker. An interesting problem therefore would be to investigate whether our scheme (or a generic construction implying our scheme) still implies message encryption functionality.

Anonymous Credentials on a Standard Java Card

- Our implementation of anonymous credentials on a Java Card [17] demonstrates the feasibility of such an approach. As we have not used a modern smart card, we could not implement our solution in a fully standards-compliant way. The reason is that RSA secret keys can only be stored in RAM starting with the Java Card 3.0 standard [103]. It would be interesting to analyse the performance of a state-of-the-art smart card running a standards-compliant implementation of an anonymous credential system.

- The DAA-like protocol as implemented in our prototype only has limited applications. It would be interesting to assess the performance of an approach using a user-owned device, such as a mobile phone, in combination with an embedded smart card. The advantage of such approach not only lies in the additional computational power that the mobile phone provides but also in its I/O capabilities that allow for an interaction with the user. Such implementation comes along with the challenges of designing a suitable UI and reasoning why the user should trust the display to show the information to be processed by the system.

Mixing Identities with Ease

- Regarding the architecture and implementation of *Idemix* [16], there are a number of interesting questions. For example: How can we make better use of the multiple cores (and threads) of modern CPUs? What is the corresponding performance benefit? How can we make the computation of proofs more modular? Could we even use a compiler for zero-knowledge proofs such as the one proposed by Almeida et al. [1]?

- We believe that the concepts underlying anonymous credentials are not yet well understood by a broad audience. An open question is how the (counter-intuitive) functionality of anonymous credential systems can be efficiently explained to a broad audience. A didactic model abstracting the cryptographic details and focusing on the general ideas would be of help.

A Comprehensive Framework Enabling Data-Minimizing Authentication

- Integration of *Idemix* into a more general authentication framework [19] has been a huge step towards demonstrating how data-minimizing authentication can be implemented. Extending the resulting framework with further technologies such as U-Prove [31] or X.509 would be a way to validate the general applicability of our approach.

- Ardagna et al. [2] describe the extension of SAML and XACML necessary to incorporate data-minimizing authentication functionality. Our framework currently uses the CARL policy language [41] to communicate authentication

requirements from a service provider to a user. Extending our framework to function with those (extended) standards would validate the proposed changes.

- While there are many research contributions extending the functionality of data-minimizing authentication technology, there are only very few studies on corresponding UIs. There are some ideas [20, 110] on how to visualize capabilities of zero-knowledge proofs but there are no results on the learning curve a user undergoes when using data-minimizing authentication technology such as anonymous credentials. Since the adoption of this technology is heavily depending on the acceptance of users, we see many questions in UI design that should be analyzed.

Recognizing Your Digital Friends

- We only have a coarse understanding of how people interpret relations and personal communication in the digital domain. There are interesting questions: How good is the general understanding of problems associated with digital attributes, for example, email addresses that show up in the "From" field are not authenticated? How is the situation with attributes in an ESN? Does this understanding differ with respect to age, education, or provenance? Analysis of such questions would provide information on whether or not the gap between recognition processes in the offline vs. online world are understood and influence the behaviour of people.

Security and Trust through Electronic Social Network-based Interactions

- Formally, bootstrapping of security properties without pre-shared information is not possible [81]. Our proposal of a trust establishment and management mechanism combining digital information from ESNs with additional, out-of-band information [22] establishes the confidence of a person in attribute values of her communication partner. Could we express the properties achieved with such approach in a formal way?

- We assume our approach to be efficient because it leverages current user behaviour. An implementation could confirm or refuse such claim. In addition it could answer the question of whether or not a user does realize the difference between merely "friending" a person vs. signing a public key.

- Our assumption that the establishment process of a WoT alone is responsible for the poor adoption can be challenged. It would be interesting to learn if the establishement is a main factor or if the lack of potential use for a set of authentic public keys is an even stronger incentive not to invest in this cumbersome process. If the latter were the case, it would be challenging to find convincing use cases for authenticated public keys for an "average" person.

- The current error messages warning users from establishing an SSL/TLS encrypted connection in e-business transactions has been shown to be ineffective, that is, many users ignore those warnings and establish communication with a malicious entity [104]. This shows that the mechanisms of the currently deployed PKI are not apparent to users and they do not act appropriately (e.g., stop interaction upon a certificate warning). We would be interested in the metrics for a comparison of the current approach with our proposal assuming that users can indicate their trust in public keys on corporate ESN pages.

Bibliography

[1] José B. Almeida, Endre Bangerter, Manuel Barbosa, Stephan Krenn, Ahmad-Reza Sadeghi, and Thomas Schneider. A certifying compiler for zero-knowledge proofs of knowledge based on Σ-protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Proc. of the 15th European Symposium on Research in Computer Security (ESORICS)*, volume 6345 of *Lecture Notes in Computer Science*, pages 151–167, Athens, Greece, September 2010. Springer.

[2] Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Gregory Neven, Stefano Paraboschi, Franz-Stefan Preiss, Pierangela Samarati, and Mario Verdicchio. Enabling privacy-preserving credential-based access control with XACML and SAML. In *3rd IEEE International Symposium on Trust, Security and Privacy for Emerging Applications (TSP)*, Proc. of the 10th IEEE International Conference on Computer and Information Technology (CIT), pages 1090–1095, Bradford, UK, July 2010. IEEE Computer Society Press.

[3] Sabrina Ardagna, Claudio A. and De Capitani di Vimercati, Stefano Paraboschi, Eros Pedrini, and Pierangela Samarati. An XACML-based privacy-centered access control system. In *Proc. of the 1st ACM Workshop on Information Security Governance (WISG)*, pages 49–58, Chicago, IL, USA, November 2009. ACM Press.

[4] Giuseppe Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *Proc. of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 138–146, Kent Ridge Digital Labs, Singapore, November 1999. ACM Press.

[5] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *Advances in Cryptology: CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270, Santa Barbara, CA, USA, August 2000. Springer.

[6] Giuseppe Ateniese, Dawn Song, and Gene Tsudik. Quasi-efficient revocation of group signatures. In Matt Blaze, editor, *Proc. of the 6th International*

Conference on Financial Cryptography (FC), volume 2357 of *Lecture Notes in Computer Science*, pages 183–197, Southampton, Bermuda, March 2002. Springer.

[7] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. In Matthew Franklin, editor, *Proc. of the 3rd International Conference on Financial Cryptography (FC)*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211, Anguilla, British West Indies, February 1999. Springer.

[8] Liana B. Baker. Sony suffers second major user data theft. `http://www.reuters.com/article/2011/05/02/us-sony-idUSTRE73R0Q320110502`, May 2011. *accessed: 22 August 2011.*

[9] Liana B. Baker and Jim Finkle. Sony PlayStation suffers massive data breach. `http://www.reuters.com/article/2011/04/26/us-sony-stoldendata-idUSTRE73P6WB20110426`, April 2011. *accessed: 22 August 2011.*

[10] Josep M. Balasch. Smart card implementation of anonymous credentials. Master's thesis, K.U.Leuven, Belgium, Leuven, Belgium, 2008.

[11] BEA, IBM, Microsoft, RSA Security, and VeriSign. Web services federation language (WS-federation). `http://schemas.xmlsoap.org/ws/2003/07/secext/`, July 2003. Specification.

[12] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology: EUROCRYPT '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 2003. Springer.

[13] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153, San Francisco, CA, USA, February 2005. Springer.

[14] Mihir Bellare and Moti Yung. Certifying permutations: Non-interactive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, 9(3):149–166, 1996.

[15] Patrik Bichsel. Theft and misuse protection for anonymous credentials. Master's thesis, ETH Zürich, Switzerland, November 2007.

[16] Patrik Bichsel and Jan Camenisch. Mixing identities with ease. In Evelyne De Leeuw, Simone Fischer-Hübner, and Lothar Fritsch, editors, *IFIP Working Conference on Policies & Research in Identity Management*

(IDMAN), volume 343 of *IFIP Advances in Information and Communication Technology*, pages 1–17, Oslo, Norway, November 2010. Springer.

[17] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 600–610, Chicago, IL, USA, November 2009. ACM Press.

[18] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get Shorty via Group Signatures without Encryption. In Juan A. Garay and Roberto De Prisco, editors, *Proc. of the 7th Conference on Security and Cryptography for Networks (SCN)*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398, Amalfi, Italy, September 2010. Springer.

[19] Patrik Bichsel, Jan Camenisch, and Franz-Stefan Preiss. A comprehensive framework enabling data-minimizing authentication. In Thomas Groß and Kenji Takahashi, editors, *Proc. of the 7th ACM Workshop on Digital Identity Management (DIM)*, pages 13–22, Chicago, IL, USA, November 2011. ACM Press.

[20] Patrik Bichsel, Jan Camenisch, Franz-Stefan Preiss, and Dieter Sommer. Dynamically-changing interface for interactive selection of information cards satisfying policy requirements. *IBM Technical Report RZ 3756 (# 99766)*, IBM Research – Zurich, December 2009.

[21] Patrik Bichsel, Jan Camenisch, and Mario Verdicchio. Recognizing your digital friends. In *1st International Workshop on Security and Privacy in Social Networks (SPSN)*, IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT), and IEEE International Confernece on Social Computing (SocialCom), pages 1310–1313, Boston, MS, USA, October 2011. IEEE Computer Society Press.

[22] Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. Security and trust through electronic social network-based interactions. In *Workshop on Security and Privacy in Online Social Networking (SPOSN)*, volume 4 of *International Conference on Computational Science and Engineering (CSE)*, pages 1002–1007, Vancouver, Canada, August 2009. IEEE Computer Society Press.

[23] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology: CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Santa Barbara, CA, USA, August 1998. ACM Press.

[24] Manuel Blum, Alfredo De Santis, Silvio Micali, and Guiseppe Persiano. Non-interactive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[25] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology: EUROCRYPT '04*, volume 3027 of *Lecture Notes in Computer Science*, pages 54–73, Interlaken, Switzerland, May 2004. Springer.

[26] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology: CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[27] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 168–177, Washington, DC, USA, October 2004. ACM Press.

[28] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology: EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Bruges, Belgium, May 2000. Springer.

[29] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.

[30] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Proc. of the 12th Australasian Conference on Information Security and Privacy (ACISP)*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415, Townsville, Australia, July 2007. Springer.

[31] Stefan Brands and Christian Paquin. U-Prove cryptographic specification v1.0. Technical report, Microsoft Research, March 2010.

[32] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. IBM Research Report RZ 3450, IBM Research – Zurich, March 2004.

[33] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul Syverson, and Somesh Jha, editors, *Proc. of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 345–356, Alexandria, VA, USA, November 2008. ACM Press.

[34] 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, *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 201–210, Alexandria, VA, USA, October 2006. ACM Press.

[35] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *Proc. of the 12th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500, Irvine, CA, USA, March 2009. Springer.

[36] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. Solving revocation with efficient update of anonymous credentials. In Juan A. Garay and Roberto De Prisco, editors, *Proc. of the 7th Conference on Security and Cryptography for Networks (SCN)*, pages 454–471, Amalfi, Italy, September 2010. Springer.

[37] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, March 2001. Springer.

[38] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. `http://eprint.iacr.org/2001/019`, 2001.

[39] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Proc. of the 3th Conference on Security and Cryptography for Networks (SCN)*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, September 2003. Springer.

[40] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology: CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 2004. Springer.

[41] Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-preserving access control. In Barbara Carminati James B. D. Joshi, editor, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 119–128, Pittsburgh, PA, USA, June 2010. ACM Press.

[42] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology: CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144, Santa Barbara, CA, USA, August 2003. Springer.

[43] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In Vijay Atluri, editor, *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, Washington, DC, USA, November 2002. ACM Press.

[44] Kim Cameron. The laws of identity. `http://www.identityblog.com/?page_id=354`, May 2005. *accessed: 15 December 2011.*

[45] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[46] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: CRYPTO 1982*, pages 199–203, Santa Barbara, CA, USA, August 1983. Plenum Press.

[47] David Chaum. Blind signature systems. In David Chaum, editor, *Advances in Cryptology: CRYPTO 1983*, page 153, Santa Barbara, CA, USA, August 1984. Plenum Press.

[48] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[49] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology: CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Santa Barbara, CA, USA, August 1990. Springer.

[50] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology: EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265, Brighton, UK, April 1991. Springer.

[51] Lidong Chen and Torben P. Pedersen. New group signature schemes. In Alfredo De Santis, editor, *Advances in Cryptology: EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181, Perugia, Italy, May 1995. Springer.

[52] OpenID Consortium. OpenID authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`, December 2007. Specification.

[53] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[54] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology: CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Santa Barbara, CA, USA, August 1994. Springer.

[55] Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology: ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, December 2002. Springer.

[56] Luuk Danes. Smart card integration in the pseudonym system Idemix. Master's thesis, University of Groningen, Mathematics Department, Groningen, Netherlands, 2007.

[57] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proc. of the 24th IEEE Symposium on Security and Privacy*, pages 2–15, Berkeley, CA, USA, May 2003. IEEE Computer Society Press.

[58] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - 1st International Conference on Cryptology in Vietnam (VIETCRYPT)*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210, Hanoi, Vietnam, September 2006. Springer.

[59] Rachna Dhamija, J. D. Tygar, and Marti Hearst. Why phishing works. In Rebecca Grinter, Thomas Rodden, Paul Aoki, Ed Cutrell, Robin Jeffries, and Gary Olson, editors, *Proc. of the SIGCHI conference on Human Factors in computing systems (CHI '06)*, pages 581–590, Montréal, Québec, Canada, April 2006. ACM Press.

[60] Claudia Diaz. *Anonymity and Privacy in Electronic Services*. PhD thesis, K.U.Leuven, Belgium, Leuven, Belgium, December 2005.

[61] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246 (Proposed Standard), January 1999. Obsoleted by RFC 4346, updated by RFCs 3546, 5746, 6176.

[62] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[63] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proc. of the 13th USENIX Security Symposium*, pages 303–320, San Diego, CA, USA, August 2004. USENIX Association.

[64] Peter Eckersley and Jesse Burns. Is the ssliverse a safe place? `https://www.eff.org/files/ccc2010.pdf`, December 2010. Talk at 27th Chaos Communication Congress.

[65] European Digital Rights (EDRi). Data retention regime in discussion all over europe. `http://www.edri.org/edrigram/number8.22/data-retention-regime-discussion`, November 2010. *accessed: 21 February 2012.*

[66] The Age (Fairfax Media). The hack of the year. `http://www.theage.com.au/news/security/tor-hack/2007/11/12/1194766589522.html`, November 2007. *accessed: 15 January 2012.*

[67] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, September 2008.

[68] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):84–88, February 1999.

[69] Michael Herrmann and Christian Grothoff. Privacy-implications of performance-based peer selection by onion-routers: A real-world case study using i2p. In Simone Fischer-Hübner and Nicholas Hopper, editors, *Proc. of the 11th Privacy Enhancing Technologies Symposium (PETS)*, volume 6794 of *Lecture Notes in Computer Science*, pages 155–174, Waterloo, Canada, July 2011. Springer.

[70] G. Hogben. Security issues in the future of social networking. In *W3C Workshop on the Future of Social Networking*, 2009.

[71] I2P Team. I2P anonymous network. `http://www.i2p2.de/`, January 2012. *accessed: 15 January 2012.*

[72] Blake Ives, Kenneth R. Walsh, and Helmut Schneider. The domino effect of password reuse. *Communications of the ACM*, 47(4):75–78, April 2004.

[73] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), February 2003.

[74] Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. Remote physical device fingerprinting. In *Proc. of the 26th IEEE Symposium on Security and Privacy*, pages 211–225, Oakland, CA, USA, May 2005. IEEE Computer Society Press.

[75] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Analysis of revocation strategies for anonymous *idemix* credentials. In Bart De Decker, Jorn Lapon, Vincent Naessens, and Andreas Uhl, editors, *12th IFIP TC 6 / TC 11 International Conference on Communications and Multimedia Security (CMS)*, volume 7025 of *Lecture Notes in Computer Science*, pages 3–17, Ghent, Belgium, October 2011. Springer.

[76] Jorn Lapon, Kristof Verslype, Pieter Verhaeghe, Bart De Decker, and Vincent Naessens. PetAnon: a fair and privacy-preserving petition system. In Vashek Matyáš, Simone Fischer-Hübner, Daniel Cvrcek, and Petr Švenda, editors, *Pre-Proceedings of The Future of Identity in the Information Society*, pages 73–78, Brno, Czech Republic, September 2008. FIDIS/IFIP Internet Security & Privacy Summer School.

[77] Ben Laurie. Improving SSL certificate security. `http://googleonlinesecurity.blogspot.com/2011/04/improving-ssl-certificate-security.html`, April 2011. *accessed: 10 January 2012*.

[78] Anna Lysyanskaya. Pseudonym systems. Master's thesis, MIT, Cambridge, MA, USA, 1999.

[79] Gareth B. Matthews. Aristotelian essentialism. *Philosophy and Phenomenological Research*, 50:251–262, 1990.

[80] Ueli M. Maurer and Pierre E. Schmid. A calculus for secure channel establishment in open networks. In Dieter Gollmann, editor, *Proc. of the 3rd European Symposium on Research in Computer Security (ESORICS)*, volume 875 of *Lecture Notes in Computer Science*, pages 173–192, Brighton, UK, November 1994. Springer.

[81] Ueli M. Maurer and Pierre E. Schmid. A calculus for security bootstrapping in distributed systems. *Journal of Computer Security*, 4(1):55–80, 1996. Preliminary version in: *ESORICS'94*, vol. 875 of *LNCS*.

[82] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol Version 2. `http://tools.ietf.org/html/draft-sassaman-mixmaster-03`, December 2004. IETF Internet Draft.

[83] Wojciech Mostowski and Pim Vullers. Efficient U-Prove implementation for anonymous credentials on smart cards. In George Kesidis and Haining Wang, editors, *Proc. of the 7th International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, Lecture Notes in Computer Science, London, UK, September 2011. Springer.

[84] Steven Murdoch and George Danezis. Low cost traffic analysis of Tor. In *Proc. of the 26th IEEE Symposium on Security and Privacy*, pages 183–195, Oakland, CA, USA, May 2005. IEEE Computer Society Press.

[85] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Stanislaw Jarecki and Gene Tsudik, editors, *Proc. of the 12th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480, Irvine, CA, USA, March 2009. Springer.

[86] Dave Neal. Privacy groups want to rethink eu data retention plans. `http://s.tt/14jA3`, September 2011. *accessed: 21 February 2012.*

[87] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292, San Francisco, CA, USA, February 2005. Springer.

[88] OASIS. Assertions and protocols for the OASIS Security Assertion Markup Language (SAML) v2.0. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`, March 2005. OASIS Standard.

[89] OASIS. eXtensible Access Control Markup Language (XACML) V2.0. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`, March 2005. OASIS Standard.

[90] Oracle. Java Card platform specification 3.0.1. `http://java.sun.com/javacard/3.0.1/specs.jsp`, December 2011. Specification.

[91] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology: CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 1992. Springer.

[92] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis C. Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, Soazig Guillou, and Thomas A. Berson. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology: CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631, Santa Barbara, CA, USA, August 1990. Springer.

[93] Michael O. Rabin and Jeffrey O. Shallit. Randomized algorithms in number theory. *Communications in Pure and Applied Mathematics*, 39:239–256, 1986.

[94] David Recordon and Drummond Reed. OpenID 2.0: A platform for user-centric identity management. In Ari Juels, Marianne Winslett, and Atsuhiro Goto, editors, *Proc. of the 2nd ACM Workshop on Digital Identity Management (DIM)*, pages 11–16, Alexandria, VA, USA, November 2006. ACM Press.

[95] Michael Reiter and Aviel Rubin. Crowds: Anonymity for Web transactions. *Communications of the ACM*, 1(1):66–92, June 1998.

[96] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[97] S. Santesson and P. Hallam-Baker. Online Certificate Status Protocol Algorithm Agility. RFC 6277 (Proposed Standard), June 2011.

[98] Security Team, IBM Research – Zurich. Specification of the Identity Mixer cryptographic library (a.k.a. cryptographic protocols of the Identity Mixer library). *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, April 2010.

[99] Andrei Serjantov and George Danezis. Towards an information theoretic metric for anonymity. In Roger Dingledine and Paul Syverson, editors, *Proc. of the 2nd Workshop on Privacy Enhancing Technologies (PETS)*, volume 2482 of *Lecture Notes in Computer Science*, pages 259–263, San Francisco, CA, USA, April 2002. Springer.

[100] Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[101] Michaël Sterckx, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede. Efficient implementation of anonymous credentials on Java Card smart cards. In *First IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 106–110, London, UK, December 2009. IEEE Computer Society Press.

[102] Sun Microsystems. Java Card platform specification 2.2.1. `http://java.sun.com/javacard/specs.html`, October 2003. Specification.

[103] Sun Microsystems. Java Card platform specification 3.0. `http://java.sun.com/javacard/3.0/specs.jsp`, April 2008. Specification.

[104] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: an empirical study of SSL warning effectiveness. In *Proc. of the 18th USENIX Security Symposium*, pages 399–416, San Diego, CA, USA, June 2009. USENIX Association.

[105] Hiroko Tabuchi. Sony says parts of PlayStation network will be back online this week. `http://www.nytimes.com/2011/05/02/technology/02sony.html`, May 2011. *accessed: 12 September 2011.*

[106] The JAP Team. Project: AN.ON - anonymity.online. `http://anon.inf.tu-dresden.de/index_en.html`, 2012. *accessed: 15 January 2012.*

[107] TheFirewall.co.uk. Research reveals the folly of sony's handling of its data breach double whammy. `http://www.thefirewall.co.uk/news/71/`, May 2011. *accessed: 22 August 2011.*

[108] Trusted Computing Group. TCG TPM specification 1.2. `www.trustedcomputinggroup.org`, 2003. Specification.

[109] Bibi van den Berg and Ronald Leenes. Keeping up appearances: Audience segregation in social network sites. In Serge Gutwirth, Yves Poullet, Paul De Hert, and Ronald Leenes, editors, *Computers, Privacy and Data Protection: An Element of Choice*, pages 211–231. Springer, 1st edition, 2011.

[110] Erik Wästlund, Julio Angulo, and Simone Fischer-Hübner. Evoking comprehensive mental models of anonymous credentials. In *Open Problems in Network Security: IFIP WG 11.4 International Workshop*, Lucerne, Switzerland, June 2011. Springer.

[111] Phillip Windley. *Digital Identity.* O'Reilly Media, Inc., Sebastopol, CA, USA, 2005.

[112] Jeff Yan, Alan Blackwell, Ross Anderson, and Alasdair Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31, September 2004.

[113] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510 (Proposed Standard), June 2006.

[114] Philip R. Zimmermann. *The official PGP user's guide.* MIT Press, Cambridge, MA, USA, 1995.

Publications

List of Publications

Conference Publications

1. Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get Shorty via Group Signatures without Encryption. In Juan A. Garay and Roberto De Prisco, editors, *Proc. of the 7th Conference on Security and Cryptography for Networks (SCN)*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398, Amalfi, Italy, September 2010. Springer.

   – See p. 69.

2. Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 600–610, Chicago, IL, USA, November 2009. ACM Press.

   – See p. 103.

3. Patrik Bichsel and Jan Camenisch. Mixing identities with ease. In Evelyne De Leeuw, Simone Fischer-Hübner, and Lothar Fritsch, editors, *IFIP Working Conference on Policies & Research in Identity Management (IDMAN)*, volume 343 of *IFIP Advances in Information and Communication Technology*, pages 1–17, Oslo, Norway, November 2010. Springer.

   – See p. 131.

4. Patrik Bichsel, Jan Camenisch, and Franz-Stefan Preiss. A comprehensive framework enabling data-minimizing authentication. In Thomas Groß and Kenji Takahashi, editors, *Proc. of the 7th ACM Workshop on Digital Identity Management (DIM)*, pages 13–22, Chicago, IL, USA, November 2011. ACM Press.

   – See p. 153.

5. Patrik Bichsel, Jan Camenisch, and Dieter Sommer. A calculus for privacy-friendly authentication. In *Proc. of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT)*, Newark, NJ, USA, June 2012. ACM Press.

6. Patrik Bichsel, Jan Camenisch, and Mario Verdicchio. Recognizing your digital friends. In *1st International Workshop on Security and Privacy in Social Networks (SPSN)*, IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT), and IEEE International Confernece on Social Computing (SocialCom), pages 1310–1313, Boston, MS, USA, October 2011. IEEE Computer Society Press.

   – See p. 181.

7. Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. Security and trust through electronic social network-based interactions. In *Workshop on Security and Privacy in Online Social Networking (SPOSN)*, volume 4 of *International Conference on Computational Science and Engineering (CSE)*, pages 1002–1007, Vancouver, Canada, August 2009. IEEE Computer Society Press.

   – See p. 193.

Book Chapter

1. Patrik Bichsel, Jan Camenisch, and Mario Verdicchio. Recognizing your digital friends. In Yuval Elovici, Yaniv Altshuler, Armin Cremers, Nada Aharony, and Alex Pentland, editors, *Security and Privacy in Social Networks*. Springer, 2012.

Patents

1. Patrik Bichsel, Jan Camenisch, and Thomas Groß. Transaction auditing for data security devices. *International Patent Application WO 2011/104654 A1 (PCT/IB2011/050638)*, World Intellectual Property Organization (WIPO), September 2011.

2. Patrik Bichsel, Franz-Stefan Preiss, and Mario Verdicchio. Set definition in data processing systems. *United States Patent Application US 2011/0087999 A1 (12/924,600)*, April 2011.

Technical Reports

1. Security Team, IBM Research – Zurich. Specification of the Identity Mixer cryptographic library (a.k.a. cryptographic protocols of the Identity Mixer library). *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, April 2010.

2. Patrik Bichsel, Jan Camenisch, Franz-Stefan Preiss, and Dieter Sommer. Dynamically-changing interface for interactive selection of information cards satisfying policy requirements. *IBM Technical Report RZ 3756 (# 99766)*, IBM Research – Zurich, December 2009.

3. Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Thomas Heydt-Benjamin, Dieter Sommer, and Greg Zaverucha (Contributors). Cryptographic protocols of the Identity Mixer library. *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, March 2009.

Get Shorty via Group Signatures without Encryption

Publication Data

Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get Shorty via Group Signatures without Encryption. In Juan A. Garay and Roberto De Prisco, editors, *Proc. of the 7th Conference on Security and Cryptography for Networks (SCN)*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398, Amalfi, Italy, September 2010. Springer.

Contributions

- Principal author.
- Security definitions, scheme, comparison, proof of Theorem 5.1.

Copyright

Original version: `http://dx.doi.org/10.1007/978-3-642-15317-4_24`

Get Shorty via Group Signatures without Encryption

P. Bichsel[1], J. Camenisch[1], G. Neven[1], N.P. Smart[2], and B. Warinschi[2]

[1] IBM Research – Zurich,
Switzerland.
{pbi,jca,nev}@zurich.ibm.com
[2] Dept Computer Science,
Univeristy of Bristol,
United Kingdom.
{nigel,bogdan}@cs.bris.ac.uk

Abstract. Group signatures allow group members to anonymously sign messages in the name of a group such that only a dedicated opening authority can reveal the exact signer behind a signature. In many of the target applications, for example in sensor networks or in vehicular communication networks, bandwidth and computation time are scarce resources and many of the existent constructions simply cannot be used. Moreover, some of the most efficient schemes only guarantee anonymity as long as no signatures are opened, rendering the opening functionality virtually useless.

In this paper, we propose a group signature scheme with the shortest known signature size and favorably comparing computation time, whilst still offering a strong and practically relevant security level that guarantees secure opening of signatures, protection against a cheating authority, and support for dynamic groups. Our construction departs from the popular sign-and-encrypt-and-prove paradigm, which we identify as one source of inefficiency. In particular, our proposal does not use standard encryption and relies on re-randomizable signature schemes that hide the signed message so as to preserve the anonymity of signers.

Security is proved in the random oracle model assuming the XDDH, LRSW and SDLP assumptions and the security of an underlying digital signature scheme. Finally, we demonstrate how our scheme yields a group signature scheme with verifier-local revocation.

Key words: Group signatures, pairings, group signature security definition.

1 Introduction

Group signatures, introduced in 1991 by Chaum and van Heyst [19], allow members of a group to anonymously sign messages on behalf of the whole group. For

example, they allow an employee of a company to sign a document in such a way that the verifier only learns that it was signed by an employee, but not by which employee. Group membership is controlled by a *Group Manager*, who can add users (called *Group Members*) to the group. In addition, there is an *Opener* who can reveal the identity of signers in the case of disputes. In some schemes, such as the one we propose, the tasks of adding members and revoking anonymity are combined into a single role. In the systems proposed in [3, 16, 34], group membership can be selectively revoked, i.e., without affecting the signing ability of the remaining members.

Security notions.

Since 1991 a number of security properties have been developed for group signatures including unforgeability, anonymity, traceability, unlinkability, and non-frameability. In 2003 Bellare, Micciancio, and Warinschi [4] developed what is now considered the standard security model for group signatures. They propose two security properties for static groups called *full anonymity* and *full traceability* and show that these capture the previous security requirements of unforgeability, anonymity, traceability, and unlinkability. Bellare, Shi, and Zhang [7] extended the notions of [4] to dynamic groups and added the notion of *non-frameability* (or exculpability), by which the Group Manager and Opener together cannot produce a signature that can be falsely attributed to an honest Group Member.

Boneh and Shacham [11] proposed a relaxed anonymity notion called *selfless anonymity* where signers can trace their own signatures, but not those of others. This weakening, however, leads to the following feature: if a group member signed a message but forgot that she signed it, then she can recover this information from the signature itself. Other schemes [10, 12, 13] weaken the anonymity notion by disallowing opening oracle queries, providing only so-called CPA-anonymity. This is a much more serious limitation: in practice it means that all security guarantees are lost as soon as a single signature is opened, thereby rendering the opening functionality virtually useless. As we've witnessed for the case of encryption [8], CCA2-security is what can make it into practice.

In this work, we consider a hybrid between the models of [7] and [11] that combines the dynamic group setting and the non-frameability notion of [7] with the selfless anonymity notion and the combined roles of Group Manager and Opener of [11]. We stress however that we prove security under the practically relevant CCA2-anonymity notion, rather than the much weaker CPA-anonymity notion. Yet still, our scheme compares favourably with all known schemes that offer just CPA-anonymity.

Construction paradigms.

Many initial group signature schemes were based on the Strong-RSA assumption [2, 3, 16]. In recent years the focus has shifted to schemes based on bilinear

maps [10,11,17,26,33], which are the most efficient group signatures known today, both in terms of bandwidth and computational efficiency.

Most existing group signature schemes follow the construction paradigm where a group signature consists of an anonymous signature, an encryption of the signer's identity under the Opener's public key, and a non-interactive zero-knowledge (NIZK) proof that the identity contained in the encryption is indeed that of the signer. While very useful as an insight, this construction paradigm seems to stand in the way of more efficient schemes. In this paper, we depart from the common paradigm and construct a group signature scheme that consists solely of an anonymous signature scheme and a NIZK proof, removing the need to encrypt the identity of the signer. We thereby obtain the most efficient group signature scheme currently known, both in terms of bandwidth and computational resources (see Appendix 6).

It is surprising that we can do without a separate encryption scheme, given that group signatures as per [4] are known to imply encryption [1]. This implication however does not hold for group signatures with selfless anonymity, giving us the necessary slack to construct more efficient schemes while maintaining a practically relevant security level.

Our scheme.

In our construction each Group Member gets a Camenisch-Lysyanskaya (CL) [17] signature on a random message as a secret key. To produce a group signature, the Group Member re-randomizes this signature and produces a NIZK proof that she knows the message underlying the signature. The novel feature is that the Opener (alias Group Manager) can use information collected during the joining phase to test which user created the signature, without the need for a separate encryption.[3] A disadvantage is that opening thereby becomes a linear operation in the number of Group Members. Since opening signatures is a rather exceptional operation and is performed by the Group Manager who probably has both the resources and the commercial interest to expose traitors, we think that this is a reasonable price to pay.

CL signatures and NIZK proofs have been combined before to produce "group-like" signatures, most notably in the construction of pairing-based DAA schemes [14, 21, 22]. DAA schemes are not genuine group signatures, however, as there is no notion of an Opener.

Finally, we note that from a certain class of group signature schemes as per our definitions (that includes our scheme), one can build a group signature scheme with verifier-local revocation (VLR) [11]. Such a scheme allows verifiers to check whether a signature was placed by a revoked group member by matching it against

[3] If the random messages were known to the Group Manager, he could open group signatures simply by verifying the re-randomized signatures against the issued random messages. To achieve non-frameability, however, the random message is only known to the Group Member, so opening in our scheme is slightly more involved.

a public revocation list. The converse is not true, i.e., a VLR scheme does not automatically yield a group signature as per our definitions, as it does not provide a way to open individual signatures (rather than revoking all signatures by one signer). We refer to Section 3.2 for details.

2 Preliminaries

Notation.

If S is a set, we denote the act of sampling from S uniformly at random and assigning the result to the variable x by $x \leftarrow S$. If S consists of a single element $\{s\}$, we abbreviate this to $x \leftarrow s$. We let $\{0,1\}^*$ and $\{0,1\}^t$ denote the set of binary strings of arbitrary length and length t respectively, and let ε denote the empty string. If A is an algorithm, we denote the action of obtaining x by invoking A on inputs $y_1, \ldots, y_n$ by $x \leftarrow A(y_1, \ldots, y_n)$, where the probability distribution on x is determined by the internal coin tosses of A. We denote an interactive protocol P as $P = (P_0, P_1)$. Executing the protocol on input in_0 and in_1, resulting in the respective output out_0 and out_1, we write as $\langle out_0; out_1 \rangle \leftarrow \langle P_0(in_0); P_1(in_1) \rangle$. If **arr** is an array or list we let **arr**$[i]$ denote the ith element in the array/list.

Digital Signature Scheme.

We will use a digital signature scheme consisting of three algorithms, namely a key generation algorithm $\mathsf{DSKeyGen}$, a signing algorithm DSSign, and a signature verification algorithm $\mathsf{DSVerify}$. In our setting the key generation will be executed between a user and a certification authority (CA). It might be an interactive algorithm leading to the user getting a secret key sk and the CA as well as the user get the public key pk corresponding to the secret key. The signing algorithm accepts a secret key sk and a message m as input and returns a signature $\bar{\sigma} \leftarrow \mathsf{DSSign}(sk, m)$. The signature is constructed such that the verification algorithm upon input a message m', a public key pk, and a signature $\bar{\sigma}$ returns $\mathsf{DSVerify}(pk, m', \bar{\sigma})$, which is true if both $m' \equiv m$, and sk corresponds to pk and false otherwise. The signature scheme must satisfy the notion of unforgeability under chosen-message attacks [29].

Number-Theoretic Background.

Our construction will make extensive use of asymmetric pairings on elliptic curves. In particular we will use the following notation, for a given security parameter η,

- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order $q = \Theta(2^\eta)$.
- We write the group operations multiplicatively, and elements in $\mathbb{G}_1$ will generally be denoted by lower case letters, elements in $\mathbb{G}_2$ by lower case

letters with a "tilde" on them, and elements in $\mathbb{Z}_q$ by lower case Greek letters.

- We fix a generator g (resp. $\tilde{g}$) of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$).
- There is a computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:
  - For all $x \in \mathbb{G}_1$, $\tilde{y} \in \mathbb{G}_2$ and $\alpha, \beta \in \mathbb{Z}_q$ we have $\hat{e}(x^\alpha, \tilde{y}^\beta) = \hat{e}(x, \tilde{y})^{\alpha\beta}$.
  - $\hat{e}(g, \tilde{g}) \neq 1$.

Following [28] we call a pairing of Type-1 if $\mathbb{G}_1 = \mathbb{G}_2$, of Type-2 if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists a computable homomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and of Type-3 if $\mathbb{G}_1 \neq \mathbb{G}_2$ and no such homomorphism exists. In addition, in [20,32] a further Type-4 pairing is introduced in which $\mathbb{G}_2$ is a group of order q^2, namely the product of $\mathbb{G}_1$ with the $\mathbb{G}_2$ used in the Type-3 pairing setting. In practice Type-3 pairings offer the most efficient implementation choices, in terms of both bandwidth and computational efficiency.

Associated to pairings are the following computational assumptions, which we shall refer to throughout this paper:

Assumption 1 (LRSW)**.** With the notation above we let $\tilde{x}, \tilde{y} \in \mathbb{G}_2$, with $\tilde{x} = \tilde{g}^\alpha$, $\tilde{y} = \tilde{g}^\beta$. Let $O_{\tilde{x},\tilde{y}}(\cdot)$ be an oracle that, on input of a value $\mu \in \mathbb{Z}_q$, outputs a triple $A = (a, a^\beta, a^{\alpha+\mu\alpha\beta}) \in \mathbb{G}_1^3$ for a randomly chosen $a \in \mathbb{G}_1$. Then for all probabilistic polynomial time adversaries $\mathcal{A}$, the quantity $\nu(\eta)$, defined as follows, is a negligible function:

$$\nu(\eta) := \Pr[\alpha \leftarrow \mathbb{Z}_q; \beta \leftarrow \mathbb{Z}_q; \tilde{x} \leftarrow \tilde{g}^\alpha; \tilde{y} \leftarrow \tilde{g}^\beta; (\mu, a, b, c) \leftarrow \mathcal{A}^{O_{\tilde{x},\tilde{y}}(\cdot)}(\tilde{x}, \tilde{y}) : \mu \notin Q \wedge a \in \mathbb{G}_1 \wedge b = a^\beta \wedge c = a^{\alpha+\mu\alpha\beta}]$$

where Q is the set of queries passed by $\mathcal{A}$ to its oracle $O_{\tilde{x},\tilde{y}}(\cdot)$.

This assumption was introduced by Lysyanskaya et al. [30], in the case $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ for groups that are not known to admit an efficient bilinear map. The authors showed in the same paper, that this assumption holds for generic groups, and is independent of the decisional Diffie-Hellman (DDH) assumption. However, it is always applied in protocols for which the groups admit a pairing, and the above asymmetric version is the version that we will require.

Assumption 2 (XDDH; SXDH)**.** We say XDDH to hold in the pairing groups if DDH is hard in $\mathbb{G}_1$, i.e., if given a tuple $(g, g^\mu, g^\nu, g^\omega)$ for $\mu, \nu \leftarrow \mathbb{Z}_q$ it is hard to decide whether $\omega = \mu\nu \mod q$ or random. We say SXDH holds if DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Note that neither XDDH nor SXDH hold in the case of Type-1 pairings. For the others types of pairings XDDH is believed to hold, and only for Type-3 pairings SXDH is believed to hold.

To demonstrate the non-frameability of our scheme we require an additional assumption, which we call the symmetric Discrete Logarithm Assumption (SDLP).

Assumption 3 (SDLP)**.** Given the tuple $(g^\mu, \tilde{g}^\mu) \in \mathbb{G}_1 \times \mathbb{G}_2$ computing μ is a hard problem.

This is a non-standard assumption which, however, implicitly underlies many asymmetric pairing versions of protocols in the literature that are described in the symmetric pairing setting only. Note that the input to the SDLP problem can always be checked to be a valid input, as given $(h, \tilde{h})$ one can always check whether $\hat{e}(g, \tilde{h}) = \hat{e}(h, \tilde{g})$.

The above three assumptions are what we require to prove our scheme secure. However, for comparison with other schemes in the literature, we recap on the following two problems, introduced in [9] and [10], respectively.

Assumption 4 (q-SDH)**.** In a pairing situation as above, this assumption implies that given a q-tuple $(\tilde{g}^\gamma, \tilde{g}^{\gamma^2}, \ldots, \tilde{g}^{\gamma^q})$ for some hidden value of γ, it is hard to output a pair $(g^{1/(\gamma+\alpha)}, \alpha)$ for some $\alpha \in \mathbb{Z}_q$.

Assumption 5 (DLIN)**.** Given $a, b, c, a^\alpha, b^\beta, c^\gamma \in \mathbb{G}_1$, this assumption says it is hard to determine whether $\alpha + \beta = \gamma$.

CL Signatures.

Our group signature scheme is based on the pairing-based Camenisch-Lysyanskaya (CL) signature scheme [17] (Scheme A in their paper), which is provably secure under the LRSW assumption. The scheme assumes three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of prime order $q = \Theta(2^\eta)$, with a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and two generators $g \in G_1$ and $\tilde{g} \in \mathbb{G}_2$.

The secret key of the CL signature scheme consists of $\alpha, \beta \leftarrow \mathbb{Z}_q$ and the public key is defined as $(\tilde{x}, \tilde{y}) \leftarrow (\tilde{g}^\alpha, \tilde{g}^\beta) \in \mathbb{G}_2^2$. Computing a signature $s \in \mathbb{G}_1^3$ on a message $m \in \mathbb{Z}_q$ is done by choosing $a \leftarrow \mathbb{G}_1$, calculating $b \leftarrow a^\beta$ and $c \leftarrow a^{\alpha+m\alpha\beta}$, and setting $s \leftarrow (a, b, c)$. Finally, a tuple $(a, b, c) \in \mathbb{G}_1^3$ is a valid signature on a message $m \in \mathbb{Z}_q$ if both $\hat{e}(a, \tilde{x}) = \hat{e}(b, \tilde{g})$ and $\hat{e}(a, \tilde{x}) \cdot \hat{e}(b, \tilde{x})^m = \hat{e}(c, \tilde{g})$ hold.

Theorem 2.1 ([17])**.** The CL signature scheme A is existentially unforgeable against adaptive chosen message attacks [29] under the LRSW assumption.

CL signatures are re-randomizable, i.e., given a valid signature $(a, b, c) \in \mathbb{G}_1^3$ on a message m, the signature $(a^r, b^r, c^r) \in \mathbb{G}_1^3$ will also be valid for any $r \in \mathbb{Z}_q^*$. This re-randomization property is central to our new group signature scheme.

Sigma Protocols.

We will use a number of protocols to prove knowledge of discrete logarithms (and, more generally, of pre-images of group homomorphisms) and properties about them. This section recaps some basic facts about such protocols and the notation we will use.

Let $\phi : \mathbb{H}_1 \to \mathbb{H}_2$ be a group homomorphism with $\mathbb{H}_1$ and $\mathbb{H}_2$ being two groups of order q and let $y \in \mathbb{H}_2$. We will use additive notation for $\mathbb{H}_1$ and

multiplicative notation for $\mathbb{H}_2$. By $PK\{(x) : y = \phi(x)\}$ we denote the Σ-protocol for a zero-knowledge proof of knowledge of x such that $y = \phi(x)$ [15, 18]. Σ-protocols for group homomorphisms are three-move protocols where the prover chooses $\mathsf{rnd} \leftarrow \mathbb{H}_1$ and sends $\mathsf{Comm} \leftarrow \phi(\mathsf{rnd})$ to the verifier; the verifier sends back a random $\mathsf{Cha} \leftarrow \mathbb{H}_1$; the prover then sends $\mathsf{Rsp} = \mathsf{rnd} - \mathsf{Cha} \cdot x$; and the verifier checks that $\phi(\mathsf{Rsp})\phi(x)^{\mathsf{Cha}} = \mathsf{Comm}$. It is well-known that basic Σ-protocols for group homomorphisms are honest-verifier zero-knowledge proofs of knowledge of the pre-image of the group homomorphism. There is a number of different ways to turn any honest-verifier Σ-protocol into a protocol that is full zero-knowledge with perfect simulation and negligible soundness error (e.g., [23, 25]). We denote the full zero-knowledge variant of a Σ-protocol $PK\{\ldots\}$ as $FPK\{\ldots\}$.

The well-known Schnorr identification protocol is the special case $PK\{(x) : y = g^x\}$, i.e., $\phi(x) = g^x$ where g is a generator of a subgroup of order q of $\mathbb{Z}_p$. Let $\phi_1 : \mathbb{H}_1 \rightarrow \mathbb{H}_2$ and $\phi_2 : \mathbb{H}_1 \rightarrow \mathbb{H}_2$. We often write $y_1 = \phi_1(x_1) \wedge y_2 = \phi_2(x_2)$ to denote $\phi(x_1, x_2) := (\phi_1(x_1), \phi_2(x_2))$ or $y_1 = \phi_1(x) \wedge y_2 = \phi_2(x)$ to denote $\phi(x) := (\phi_1(x), \phi_2(x))$.

The "signature" variant of a Σ-protocol is obtained by applying the Fiat-Shamir heuristic [27] to the above Σ-protocol. We denote such a "signature-proof-of-knowledge" on a message $m \in \{0,1\}^*$ by, $SPK\{(x) : y = \phi(x)\}(m)$. That is, when we say that $\Sigma \leftarrow SPK\{(x) : y = \phi(x)\}(m)$ is computed, we mean that a random $\mathsf{rnd} \leftarrow \mathbb{H}_1$ is chosen and the pair $\Sigma \leftarrow (\mathsf{Cha}, \mathsf{Rsp})$ is computed where $\mathsf{Cha} \leftarrow \mathcal{H}(\phi \| y \| \phi(\mathsf{rnd}) \| m)$, $\mathsf{Rsp} \leftarrow \mathsf{rnd} - \mathsf{Cha} \cdot x$ and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is a suitable hash function. Note that $\Sigma \in \mathbb{Z}_q \times \mathbb{H}_1$. We say that $\Sigma = (\mathsf{Cha}, \mathsf{Rsp})$ is valid with respect to y and ϕ if $\mathsf{Cha} = \mathcal{H}(\phi \| y \| y^{\mathsf{Cha}} \phi(\mathsf{Rsp}) \| m)$ holds; typically y and ϕ will be clear from the context and we will just say that "Σ is valid." We further note that a unique specification of the statement (e.g., $(x) : y = \phi(x)$) that SPK "proves" needs to be included as an argument to the hash function, i.e., here $\phi \| y$, where ϕ stands for the description of the whole algebraic setting. In the random oracle model [6], one can use the forking lemma [5, 31] to extract the secrets from these SPKs if correct care is taken that the prover can indeed be efficiently rewound. Moreover, in the random oracle model one can simulate SPKs for unknown secrets by choosing $\mathsf{Cha}, \mathsf{Rsp} \leftarrow \mathbb{Z}_q$ at random and programming the random oracle so that $\mathcal{H}(\phi \| y \| y^{\mathsf{Cha}} \phi(\mathsf{Rsp}) \| m) = \mathsf{Cha}$.

3 Definitions

As mentioned in the Introduction, we propose a notion that builds a hybrid between [7] and [11]. Consequently, our definitions describe a dynamic group signature scheme with a combined role of Group Manager and Opener that obtains selfless anonymity, traceability, and non-frameability.

3.1 Syntax

A group signature scheme consists of a set of users with a unique index i who can produce signatures on behalf of the group. Initially users must interact with a trusted party to establish a public key pair. Users can become Group Members via an interaction with the Group Manager. After the interaction the user obtains a secret signing key that she can use to produce signatures on behalf of the group. The Group Manager obtains a piece of information that he can later use to identify signatures created by the user. In addition, both parties obtain some piece of publicly available information, which certifies the fact that the particular user has joined the group.

As remarked earlier, in our models we put more trust in the Group Manager by requiring that he is also in charge of opening signatures. The syntax that we require is as follows.

Definition A group signature scheme GS extended by a PKI is given by a tuple

$$(\mathsf{GSetup}, \mathsf{PKIJoin}, (\mathsf{GJoin}_U, \mathsf{GJoin}_M), \mathsf{GSign}, \mathsf{GVerify}, \mathsf{GOpen}, \mathsf{GJudge})$$

where:

1. GSetup is a setup algorithm. It takes as input a security parameter 1^η and produces a tuple (gpk, $gmsk$), where gpk is a group public key and $gmsk$ is the Group Manager's secret key. To simplify notation we assume that $gmsk$ always includes the group public key. Note that the group public key contains system parameters, which need to be checked by all entities not involved in there generation.

2. PKIJoin is an algorithm executed by a user to register with a certification authority (CA). It takes as input the index of the user i and the security parameter 1^η. The output of the protocol is the key pair $(\mathbf{usk}[i], \mathbf{upk}[i])$ consisting of user secret key and user public key or $\perp$ in case of a failure. The user public key $\mathbf{upk}[i]$ is sent to the CA, who makes it available such that anyone can get an authentic copy of it.

3. $\mathsf{GJoin} = (\mathsf{GJoin}_M, \mathsf{GJoin}_U)$ is a two-party interactive protocol used to add new users to the group. The input for the user is $(i, \mathbf{usk}[i], gpk)$, i.e., the index of the user, the user secret key, and the group public key. The input for the Group Manager is $(i, \mathbf{upk}[i], gmsk)$, i.e., the user index, the user public key, and the Group Manager's secret key.

   As a result of the interaction, the user obtains her group signing key $\mathbf{gsk}[i]$, and the Group Manager obtains some registration information $\mathbf{reg}[i]$ (which will later be used to trace signatures of i). If the protocol fails, the output of both parties is set to $\perp$.

4. GSign is the algorithm users employ to sign on behalf of the group. It takes as input an individual user signing key $\mathbf{gsk}[i]$ and the message $m \in \{0,1\}^*$

to be signed, and outputs a signature σ. We write $\sigma \leftarrow \mathsf{GSign}(\mathbf{gsk}[i], m)$ for the process of obtaining signature σ on m with secret key $\mathbf{gsk}[i]$.

5. GVerify is the signature verification algorithm. It takes as input (gpk, m, σ), i.e., the group public key, a message and a group signature, and returns 0 if the signature is deemed invalid and 1 otherwise.

6. GOpen is the algorithm for opening signatures. It takes as input $(gmsk, m, \sigma, \mathbf{reg})$, i.e., the Group Manager's secret key, a message, a valid group signature on the message, and the registration information table $\mathbf{reg}$, and returns a user index $i \in [n]$ and a proof π that user i produced signature σ, or it returns $\perp$, indicating that opening did not succeed.

   We assume that the opening algorithm, before outputting (i, π), always checks that the user i is registered, i.e., that $\mathbf{reg}[i] \neq \perp$, and that the proof π passes the judging algorithm (see the next item). If either of these checks fails, the opening algorithm outputs $\perp$.

7. GJudge is the judging algorithm. It takes as input a message m, a group signature σ on m, the group public key gpk, a user index i, the user public key $\mathbf{upk}[i]$, and a proof π and outputs 1 or 0, expressing whether the proof shows that user i created signature σ or not.

   We assume that the judging algorithm verifies the signature using the GVerify algorithm on input gpk, m, and σ.

3.2 Security notions

In this section we give the security definitions that we require from group signature schemes. We describe the oracles that are involved in our definitions, as well as the restrictions that we put on their uses. These oracles use some shared global state of the experiments in which they are provided to the adversary. In particular, at the time of their use, the sets of honest and corrupt users are defined. Also the oracles have access to the global information contained in $\mathbf{upk}$. For honest users the oracles have access to $\mathbf{gsk}$ and if the Group Manager is uncorrupted they also have access to $\mathbf{reg}$. We assume that at the beginning of the execution, the content of each entry in these arrays is set to $\perp$ (uninitialized).

We consider a setting with n users divided (statically) into sets $\mathcal{HU}$ and $\mathcal{DU}$ of honest and dishonest users, respectively. Even though our definitions appear to consider static corruptions only, one can easily see (by taking an upper bound on the number of users for n and guessing the indices of "target" users upfront) that they actually imply security in the dynamic case. However, the latter comes at the cost of losing a factor n in reduction tightness for traceability and non-frameability, and of $n^2/2$ for anonymity. For some notions the adversary $\mathcal{A}$ is actually a pair of algorithms $(\mathcal{A}_0, \mathcal{A}_1)$; we implicitly assume that $\mathcal{A}_0$ can pass state information to $\mathcal{A}_1$. Our security notions make use of the following oracles:

- $\mathsf{Ch}(b, \cdot, \cdot, \cdot)$ is the challenge oracle for defining anonymity. It accepts as input a triple formed from two identities $i_0, i_1 \in \mathcal{HU}$ and a message m, and returns a signature $\sigma^* \leftarrow \mathsf{GSign}(\mathbf{gsk}[i_b], m)$ under the signing key of user i_b, where b is a parameter of the experiment. This oracle can only be called once.

- $\mathsf{SetUPK}(\cdot, \cdot)$ takes an input the index of a user $i \in \mathcal{DU}$ and a value upk. If $\mathbf{reg}[i] \equiv \perp$ it sets the user's public key $\mathbf{upk}[i] \leftarrow upk$. The oracle can only be called before user i joins the group.

- $\mathsf{GJoin}_{UD}(\cdot)$ is an oracle that takes as input an honest user index $i \in \mathcal{HU}$ and executes the user side of the join protocol for i, i.e., $\mathsf{GJoin}_U(i, \mathbf{usk}[i], gpk)$. The local output of the protocol is stored in $\mathbf{gsk}[i]$. This oracle can be used by an adversary to execute the registration protocol with an honest user, the adversary playing the role of the Group Manager (when the latter is corrupt).

- $\mathsf{GJoin}_{DM}(\cdot)$ is an oracle that takes as input the index of a corrupt user $i \in \mathcal{DU}$ and simulates the execution of the join protocol for the (honest) Group Manager, i.e., $\mathsf{GJoin}_M(i, \mathbf{upk}[i], gmsk)$. The local output of the protocol is stored in $\mathbf{reg}[i]$. This oracle can be used by an adversary to execute the registration protocol with the (honest) Group Manager on behalf of any corrupt user.

- $\mathsf{GSign}(\cdot, \cdot)$ accepts as input pairs $(i, m) \in \mathcal{HU} \times \{0,1\}^*$ and obtains a signature on m under $\mathbf{gsk}[i]$ if the user is not corrupt, and its signing key is defined.

- $\mathsf{GOpen}(\cdot, \cdot)$ accepts as input a message-signature pair (m, σ) and returns the result of the function call $\mathsf{GOpen}(gmsk, m, \sigma, \mathbf{reg})$. The oracle refuses to open the signature attained through a call to the Ch oracle, i.e., $\sigma \equiv \sigma^*$.

Note that, depending on the precise group signature scheme, the oracles $\mathsf{GJoin}_{UD}(\cdot)$ and $\mathsf{GJoin}_{DM}(\cdot)$ may require multi-stages, i.e., interaction between the oracle and the adversary to complete the functionality. If this is the case we assume that these stages are executed by the adversary in a sequential order, as if the oracles are a single stage. Thus, we do not allow the adversary to interleave separate executions of the GJoin protocol, or execute multiple of them in parallel.

Correctness.

We define the correctness of a group signature scheme GS through a game in which an adversary is allowed to requests a signature on some message by any of the honest players. The adversary wins if either (1) the resulting signature does not pass the verification test, (2) the signature is opened as if it were produced by a different user, or (3) the proof produced by opening the signature does not pass the judging algorithm. The experiment is detailed in Figure 1. We say that GS is correct if for any adversary $\Pr[\mathbf{Exp}^{\text{corr}}_{\mathsf{GS},A}(\eta) = 1]$ is 0.

$\mathbf{Exp}^{\text{corr}}_{\mathsf{GS},\mathcal{A}}(\eta)$
$\mathcal{HU} \leftarrow \{1,\ldots,n\}$; $\mathcal{DU} \leftarrow \emptyset$
$(gpk, gmsk) \leftarrow \mathsf{GSetup}(1^\eta)$
For $i \in \mathcal{HU}$
$(\mathbf{usk}[i], \mathbf{upk}[i]) \leftarrow \mathsf{PKIJoin}(i, 1^\eta)$
$\langle \mathbf{reg}[i]; \mathbf{gsk}[i] \rangle \leftarrow \langle \mathsf{GJoin}_M(i, \mathbf{upk}[i], gmsk); \mathsf{GJoin}_U(i, \mathbf{usk}[i], gpk) \rangle$
$(i, m) \leftarrow \mathcal{A}^{\mathsf{GSign}(\cdot,\cdot),\mathsf{GOpen}(\cdot,\cdot)}(gpk)$
If $i \notin \mathcal{HU}$ then return 0
$\sigma \leftarrow \mathsf{GSign}(\mathbf{gsk}[i], m)$
If $\mathsf{GVerify}(gpk, m, \sigma) = 0$ then return 1
$(j, \pi) \leftarrow \mathsf{GOpen}(gmsk, m, \sigma, \mathbf{reg})$
If $i \neq j$ or $\mathsf{GJudge}(m, \sigma, gpk, i, \mathbf{upk}[i], \pi) = 0$ then return 1
Return 0

$\mathbf{Exp}^{\text{anon-}b}_{\mathsf{GS},\mathcal{A}}(\eta)$
$\mathcal{DU} \leftarrow \mathcal{A}_0(1^\eta)$
$\mathcal{HU} \leftarrow \{1,\ldots,n\} \setminus \mathcal{DU}$
$(gpk, gmsk) \leftarrow \mathsf{GSetup}(1^\eta)$
For $i \in \mathcal{HU}$
$(\mathbf{usk}[i], \mathbf{upk}[i]) \leftarrow \mathsf{PKIJoin}(i, 1^\eta)$
$\langle \mathbf{reg}[i]; \mathbf{gsk}[i] \rangle \leftarrow \langle \mathsf{GJoin}_M(i, \mathbf{upk}[i], gmsk); \mathsf{GJoin}_U(i, \mathbf{usk}[i], gpk) \rangle$
$b' \leftarrow \mathcal{A}_1^{\mathsf{Ch}(b,\cdot,\cdot,\cdot),\mathsf{SetUPK}(\cdot,\cdot),\mathsf{GJoin}_{DM}(\cdot),\mathsf{GSign}(\cdot,\cdot),\mathsf{GOpen}(\cdot,\cdot)}(gpk)$
Return b'

$\mathbf{Exp}^{\text{trace}}_{\mathsf{GS},\mathcal{A}}(\eta)$
$\mathcal{DU} \leftarrow \{1,\ldots,n\}$; $\mathcal{HU} \leftarrow \emptyset$
$(gpk, gmsk) \leftarrow \mathsf{GSetup}(1^\eta)$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{SetUPK}(\cdot,\cdot),\mathsf{GJoin}_{DM}(\cdot),\mathsf{GOpen}(\cdot,\cdot)}(gpk)$
If $\mathsf{GVerify}(gpk, m, \sigma) = 1$ and $\mathsf{GOpen}(gmsk, m, \sigma, \mathbf{reg}) = \perp$ then return 1
Else return 0

$\mathbf{Exp}^{\text{nf}}_{\mathsf{GS},\mathcal{A}}(\eta)$
$(\mathcal{DU}, gpk) \leftarrow \mathcal{A}_0(1^\eta)$
$\mathcal{HU} \leftarrow \{1,\ldots,n\} \setminus \mathcal{DU}$
For $i \in \mathcal{HU}$
$(\mathbf{usk}[i], \mathbf{upk}[i]) \leftarrow \mathsf{PKIJoin}(i, 1^\eta)$
$(i, m, \sigma, \pi) \leftarrow \mathcal{A}_1^{\mathsf{SetUPK}(\cdot,\cdot),\mathsf{GJoin}_{UD}(\cdot),\mathsf{GSign}(\cdot,\cdot)}(1^\eta)$
If $i \notin \mathcal{HU}$ or $\mathsf{GVerify}(gpk, m, \sigma) = 0$ then return 0
If σ was oracle output of $\mathsf{GSign}(i, m)$ then return 0
If $\mathsf{GJudge}(m, \sigma, gpk, i, \mathbf{upk}[i], \pi) = 0$ then return 1
Return 0

Figure 1: Experiments for defining the correctness and security of a group signature scheme. The particular restrictions on the uses of the oracles are described in Section 3.2.

Anonymity.

Anonymity requires that group signatures do not reveal the identity of the signer. In the experiment that we consider, the adversary controls all of the dishonest users. The adversary has access to a challenge oracle $\mathsf{Ch}(b,\cdot,\cdot,\cdot)$, which he can call only once with a triple (i_0, i_1, m), where i_0 and i_1 are the indices of two honest signers, and m is some arbitrary message. The answer of the oracle is a challenge signature $\sigma^* \leftarrow \mathsf{GSign}(\mathbf{gsk}[i_b], m)$. During the attack the adversary can (1) add corrupt users to the group of signers (via the $\mathsf{SetUPK}(\cdot,\cdot)$ and $\mathsf{GJoin}_M(\cdot)$ oracles), (2) require signatures of honest users on arbitrary messages via the GSign oracle, and (3) require opening of arbitrary signatures (except the signature σ^* obtained from the challenge oracle) via the GOpen oracle. The experiment is described in Figure 1. For any adversary that obeys the restrictions described above we define its advantage in breaking the anonymity of GS by

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathsf{GS},A}(\eta) = \Pr[\mathbf{Exp}^{\text{anon-1}}_{\mathsf{GS},A}(\eta) = 1] - \Pr[\mathbf{Exp}^{\text{anon-0}}_{\mathsf{GS},A}(\eta) = 1]$$

We say that the scheme GS satisfies the anonymity property if for any probabilistic polynomial-time adversary, its advantage is a negligible function of η.

Traceability.

Informally, *traceability* requires that no adversary can create a valid signature that cannot be traced to some user that had already been registered. We model the strong but realistic setting where all of the signers are corrupt and work against the group manager. In the game that we define, the adversary can add new signers using access to the GJoin_{DM} oracle and can request to reveal the signers of arbitrary signatures via the GOpen oracle. The goal of the adversary is to produce a valid message-signature pair (m, σ) that cannot be opened, i.e., such that the opening algorithm outputs $\perp$. For any adversary A we define its advantage in breaking traceability of group signature scheme GS by:

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GS},A}(\eta) = \Pr[\mathbf{Exp}^{\mathrm{trace}}_{\mathsf{GS},A}(\eta) = 1]$$

We say that GS is traceable if for any probabilistic polynomial-time adversary, its advantage is a negligible function of the security parameter.

Non-frameability.

Informally, *non-frameability* requires that even a cheating Group Manager cannot falsely accuse an honest user of having created a given signature. We model this property through a game that closely resembles that for traceability. The difference is that the adversary has the Group Manager's secret key (who is corrupt). During his attack the adversary can require honest users to join the group via the oracle GJoin_{UD}, and can obtain signatures of honest users through oracle GSign. The goal of the adversary is to produce a signature and a proof that this signature was

created by an honest user (who did not actually create the signature). For any adversary A we define its advantage against non-frameability of group signature scheme GS by

$$\mathbf{Adv}^{\mathrm{nf}}_{\mathsf{GS},A}(\eta) = \Pr[\mathbf{Exp}^{\mathrm{nf}}_{\mathsf{GS},A}(\eta) = 1]$$

We say that scheme GS is non-frameable if for any probabilistic polynomial-time adversary, its advantage is a negligible function of η.

Remarks.

The security definitions that we present depart from the more established ones in several ways that we describe and justify now. First, we repeat that even though our definitions appear to consider static corruptions only, they imply security in a dynamic setting.

Second, we borrow the *selfless anonymity* notion from [11] that departs from the one of [4] in that it does not allow the adversary access to the signing keys of the two signers involved in the query to the challenge oracle. Thus, we cannot grant the adversary access to the secret information of any honest user. This is a natural, mild restriction which, as discussed in the introduction, may lead to significantly more efficient schemes.

Third, our notion of traceability seems different than the notion of traceability of [4]. Indeed, according to our definition an attacker that creates a signature that opens as some honest identity is not considered an attack! We look at this scenario as a framing attack, however, and it is therefore covered under our non-frameability notion, a notion that was not modeled in [4].

Fourth, a detailed comparison of our security notion with the notion of [7] reveals that we do not provide a read and write oracle for the registration table **reg**. This follows from the fact that we combine the Group Manager with the opening authority. Thereby, the entities cannot be corrupted individually, thus, the adversary has either full access (i.e., when the Group Manager is corrupted) or he does have no access.

Group Signatures with Verifier-Local Revocation.

Let us discuss the relation of our scheme and definition with the group signature scheme with verifier-local revocation by Boneh and Shacham [11]. They define a group signature scheme with verifier-local revocation (VLR) as a scheme that has the additional feature of a revocation list. Essentially, VLR-verification of a group signature contains, in addition to the signature verification as described before, a check for each item in the revocation list whether or not it relates to the group signature at hand. If it does, then the signature is deemed invalid.

The scheme and definitions of Boneh and Shacham (1) do not have an open (or tracing) procedure and (2) assume that the group manager is fully trusted. The latter makes sense because if there is no open procedure, it is not possible to falsely blame a user for having produced a specific group signature. However, Boneh and

Shacham point out that any VLR scheme has an implicit opening algorithm: one can make a revocation list consisting of only a single user and then run the VLR group signature verification algorithm. Thus, the verification fails only in the case where the user who generated the signature is (the only) entity in the revocation list, which leads to her identification. This shows that we can convert a VLR-scheme into a group signature scheme with an Opener, however, we stress that the obtained scheme does not satisfy non-frameability.

We now point out that the opposite direction also works: for a sub-class of group signature schemes according to our definition one can construct a group signature scheme with verifier local revocation. The subclass is the schemes for which the GOpen algorithm takes as input $(gpk, m, \sigma, \mathbf{reg})$ instead of $(gmsk, m, \sigma, \mathbf{reg})$ as per our definition (i.e., it does not need to make use of the group manager's secret key). We note that the scheme we propose in this paper falls into this sub-class. Now, the idea for obtaining a VLR group signature scheme is as follows. The new key generation consists of the GSetup, PKIJoin, and $(\mathsf{GJoin}_U, \mathsf{GJoin}_M)$ where the group manager runs the users' parts as well and then just hands them their keys. The VLR group signing algorithm is essentially GSign. To revoke user i, the group manager adds $\mathbf{reg}[i]$ to the revocation list. Finally, the VLR-verification consist of GVerify and GOpen, i.e., it accepts a signature if GVerify accepts and if GOpen fails for all entries $\mathbf{reg}[i]$ in the revocation list. The security notions for VLR group signatures, namely selfless anonymity and traceability, follow from our notions of anonymity and traceability for group signatures. We do not give the precise formulation, but we note that a security model for VLR dynamic group signatures follows by combining our dynamic security model above, with the static VLR model from [11]. We also note that VLR group signatures do not provide forward-anonymity: a new revocation list can also be used on old signatures.

4 Our Group Signature Scheme

Overview of Our Scheme.

Our group signature scheme is based on two special properties of CL signatures, namely on their re-randomizability and on the fact that the signature "does not leak" the message that it authenticates. Intuitively, a user's group signing key is a CL signature on a random message ξ that only the user knows. To create a group signature for a message m, the user re-randomizes the CL signature and attaches a signature proof of knowledge of ξ on m.

If non-frameability were not a requirement, we could simply let the Group Manager choose ξ, so that he can open group signatures by checking for which of the issued values of ξ the re-randomized CL signature is valid. To obtain non-frameability, however, the Group Manager must not know ξ itself. Hence, in our scheme ξ is generated jointly during an interactive GJoin protocol between the user and the Group Manager. Essentially, this protocol is a two-party computation

where the user and the Group Manager jointly generate ξ, a valid CL signature on ξ, and a key derived from ξ that allows the Group Manager to trace signatures, but not to create them.

System Specification.

We now present the algorithms that define our efficient group signature scheme. We assume common system parameters for a given security parameter η. Namely, we assume that an asymmetric pairing is fixed, i.e., three groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of order $q > 2^\eta$ with an efficiently computable map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, together with generators g and $\tilde{g}$ of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Further, two hash functions $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$, $\mathcal{G} : \{0,1\}^* \to \mathbb{Z}_q$ are defined.

$\mathsf{GSetup}(1^\eta)$**:** The Group Manager chooses random $\alpha, \beta \leftarrow \mathbb{Z}_q$, and computes $\tilde{x} \leftarrow \tilde{g}^\alpha$ and $\tilde{y} \leftarrow \tilde{g}^\beta$. It then sets the group public key of the scheme to $gpk \leftarrow (\tilde{x}, \tilde{y})$ and the group secret key to $gmsk \leftarrow (\alpha, \beta)$.

$\mathsf{PKIJoin}(i, 1^\eta)$**:** The CA certifies public keys of a digital signature scheme as defined in Section 2. The user generates $(\mathbf{upk}[i], \mathbf{usk}[i]) \leftarrow \mathsf{DSKeyGen}(1^\eta)$ and sends $\mathbf{upk}[i]$ to the CA for certification.

$\mathsf{GJoin} = (\mathsf{GJoin}_M(i, \mathbf{upk}[i], gmsk), \mathsf{GJoin}_U(i, \mathbf{usk}[i], gpk))$**:** When a user i wants to join the group, she must have already run the $\mathsf{PKIJoin}$ algorithm. Then she runs the following protocol with the Group Manager. We assume that this protocol is run over secure channels and, for simplicity, that the parties only run one instance at a time. We also assume that if a verification for a party fails, the party informs the other party about the failure and the protocol is aborted.

1. The Group Manager chooses a random $\kappa \leftarrow \mathbb{Z}_q$, computes $t \leftarrow \mathcal{G}(\kappa)$, and sends t to the user.
2. The user i chooses $\tau \leftarrow \mathbb{Z}_q$, computes $s \leftarrow g^\tau$, $\tilde{r} \leftarrow \tilde{x}^\tau$, $k \leftarrow \hat{e}(g, \tilde{r})$, as well as $\bar{\sigma} \leftarrow \mathsf{DSSign}(\mathbf{usk}[i], k)$, sends $(s, \tilde{r}, \bar{\sigma})$ to the Group Manager and executes $FPK\{(\tau) : s = g^\tau \wedge \tilde{r} = \tilde{x}^\tau\}$ with the Group Manager.
3. The Group Manager uses $\mathsf{DSVerify}(\mathbf{upk}[i], \hat{e}(g, \tilde{r}), \bar{\sigma})$ to verify the signature. If it verifies correctly he computes $z \leftarrow s \cdot g^\kappa$ and $\tilde{w} \leftarrow \tilde{r} \cdot \tilde{x}^\kappa$, stores $(\tilde{w}, \tilde{r}, \kappa, \bar{\sigma})$ in $\mathbf{reg}[i]$, chooses $\rho \leftarrow \mathbb{Z}_q$, computes $a \leftarrow g^\rho$, $b \leftarrow a^\beta$, and $c \leftarrow a^\alpha \cdot z^{\rho\alpha\beta}$, and sends (a, b, c, κ) to the user. In addition, he executes

$$FPK\{(\alpha, \beta, \rho, \gamma) : c = a^\alpha z^\gamma \wedge a = g^\rho \wedge \tilde{x} = \tilde{g}^\alpha \wedge \tilde{y} = \tilde{g}^\beta \wedge 1 = b^\alpha / g^\gamma\}$$

with her, where $\gamma = \rho\alpha\beta$. Note that this proof allows the user to verify that $\alpha, \beta \neq 0$.

4. The user computes $\xi \leftarrow \tau + \kappa \bmod q$, and checks whether $t = \mathcal{G}(\kappa)$. She also verifies $\hat{e}(a, \tilde{y}) = \hat{e}(b, \tilde{g})$ and, if the verification is successful, stores the entry $\mathbf{gsk}[i] \leftarrow (\xi, (a, b, c))$.

Remarks: The value of ω stored in $\mathbf{reg}[i]$ allows the Opener to identify a user within the group signature scheme. In addition, the Opener can provably attribute this ω to $k = \hat{e}(g, \tilde{r})$. Consequently, a group signature can be provably attributed to k. By the unforgeability of the external signature scheme, the signature on k allows to attribute a group signature to a user public key $\mathbf{upk}[i]$. Furthermore, the FPK protocol that the Group Manager and the user execute in Step 3 of the protocol indeed proves that c was computed correctly w.r.t. a, b, $\tilde{x}$, and $\tilde{y}$. To this end, note that because of $\hat{e}(a, \tilde{y}) = \hat{e}(b, \tilde{g})$, we know that $b = a^{\beta}$ and thus $b = g^{\beta\rho}$. Subsequently, from $1 = b^{\alpha}/g^{\gamma}$ we can conclude that $\gamma = \rho\alpha\beta$ and hence that c was computed correctly by the Group Manager.

$\mathsf{GSign}(\mathbf{gsk}[i], m)$**:** Let a user i with signing key $\mathbf{gsk}[i] = (\xi, (a, b, c))$ sign the message m. She first re-randomizes the signature by choosing $\zeta \leftarrow \mathbb{Z}_q$ and computing $d \leftarrow a^{\zeta}$, $e \leftarrow b^{\zeta}$, and $f \leftarrow c^{\zeta}$, and then computes the SPK

$$\Sigma \leftarrow SPK\{(\xi) : \frac{\hat{e}(f, \tilde{g})}{\hat{e}(d, \tilde{x})} = \hat{e}(e, \tilde{x})^{\xi}\}(m)$$

proving that she knows the "message" for which (d, e, f) is a valid CL-signature. Finally, she outputs $\sigma \leftarrow (d, e, f, \Sigma) \in \mathbb{G}_1^3 \times \mathbb{Z}_q^2$ as the group signature on m.

$\mathsf{GVerify}(gpk, m, \sigma)$**:** To verify a signature $\sigma = (d, e, f, \Sigma)$ on the message m, the verifier first checks that $\hat{e}(d, \tilde{y}) = \hat{e}(e, \tilde{g})$, where $\tilde{g}, \tilde{y}$ are retrieved from gpk. Secondly, the verifier checks that the proof Σ is valid. If either of the checks fail, output 0; otherwise output 1.

$\mathsf{GOpen}(gmsk, m, \sigma, \mathbf{reg})$**:** Given signature $\sigma = (d, e, f, \Sigma)$ on m, the Group Manager verifies the signature using $\mathsf{GVerify}$. Then, for all entries $\mathbf{reg}[i] = (\tilde{w}_i, \tilde{r}_i, \kappa_i, \bar{\sigma}_i)$ he checks whether $\hat{e}(f, \tilde{g}) = \hat{e}(d, \tilde{x}) \cdot \hat{e}(e, \tilde{w}_i)$ holds. For the $\tilde{w}_i$ where the equation holds, the Group Manager retrieves κ_i and $\bar{\sigma}_i$, computes $k_i \leftarrow \hat{e}(g, \tilde{r}_i)$ and the SPK

$$\Pi \leftarrow SPK\{(\tilde{w}_i, \kappa_i) : \frac{\hat{e}(f, \tilde{g})}{\hat{e}(d, \tilde{x})} = \hat{e}(e, \tilde{w}_i) \wedge k_i = \frac{\hat{e}(g, \tilde{w}_i)}{\hat{e}(g, \tilde{x})^{\kappa_i}}\} \ ,$$

and outputs $(i, \pi = (k_i, \bar{\sigma}_i, \Pi))$.

Note that $\phi(\tilde{w}) := (\hat{e}(e, \tilde{w}), \hat{e}(g, \tilde{w}))$ is a group homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_T \times \mathbb{G}_T$ and therefore π can be obtained from applying the Fiat–Shamir transform to the underlying Σ-protocol as discussed earlier. Also note that the opening operation is linear in the number of users in the system, but we

consider this reasonable as in most practical applications opening is a rather exceptional operation performed by a resourceful Group Manager.

$\mathsf{GJudge}(gpk, m, \sigma, i, \mathbf{upk}[i], \pi)$**:** The signature of the external signature scheme is verified using the signature verification algorithm $\mathsf{DSVerify}(upk[i], k, \bar{\sigma})$. If the signature verifies, use input gpk, m, $\sigma = (d, e, f, \Sigma)$, and π, to output 1 if algorithm $\mathsf{GVerify}(gpk, m, \sigma) = 1$ and Π is valid. Otherwise output 0.

Remarks.

Following the explanations in Section 3.2, we can build a VLR scheme as follows. Transformation of the key generation and the signing algorithm are straightforward. To revoke a user i, the Group Manager publishes the corresponding entry $\tilde{w}_i$ from $\mathbf{reg}[i]$ to the revocation list **rlist**. Finally, we modify the $\mathsf{GVerify}$ algorithm so that it checks not only that $\hat{e}(d, \tilde{y}) = \hat{e}(e, \tilde{g})$ and the proof Σ is valid, but also whether

$$\hat{e}(f, \tilde{g}) = \hat{e}(d, \tilde{x}) \cdot \hat{e}(e, \tilde{w}_i)$$

for any entry $\tilde{w}_i$ in **rlist**. If this is the case, it rejects the signature. Thus, the verifier performs what has been a part of the tasks of the Opener in our basic group signature scheme.

5 Security Results

Verifying our scheme's correctness is not hard from its description (and the comments we made there). We now present our results that the scheme satisfies our anonymity, traceability, and non-frameability requirements. Proofs of the following theorems can be found in Appendix B.

Theorem 5.1. In the random oracle model the group signature scheme is anonymous under the XDDH and the SDLP assumptions.

Theorem 5.2. In the random oracle model the group signature scheme is traceable under the LRSW assumption.

Theorem 5.3. In the random oracle model the group signature scheme is non-frameable under the SDLP assumption and the unforgeability of the underlying digital signature scheme.

Security of our scheme as a VLR group signature scheme, in the random oracle model, follows from the above theorems.

6 Comparison With Previous Schemes

We compare efficiency of several schemes with respect to (1) signature size, (2) computational costs of signature generation, and (3) computational costs of signature verification. Let us begin with a detailed discussion of our scheme. The signature algorithm outputs the randomized CL signature (d, e, f), as well as the data needed to verify the signature proof of knowledge Σ. When looking at Σ in more detail, we can set

$$A \leftarrow \frac{\hat{e}(f,\tilde{g})}{\hat{e}(d,\tilde{x})} = \left(\frac{\hat{e}(c,\tilde{g})}{\hat{e}(a,\tilde{x})}\right)^{\zeta} \text{ and } B \leftarrow \hat{e}(e,\tilde{x}) = \hat{e}(b,\tilde{x})^{\zeta},$$

then the SPK is to prove knowledge of ξ such that $A = B^{\xi}$. Applying our description of SPK's obtained from Sigma protocols (see Section 2), the signer needs to compute, for random $\mathsf{rnd} \leftarrow \mathbb{Z}_q$,

$$\mathsf{Comm} \leftarrow B^{\mathsf{rnd}}, \quad \mathsf{Cha} \leftarrow \mathcal{H}(\phi \| A \| \mathsf{Comm} \| m), \quad \mathsf{Rsp} \leftarrow \mathsf{rnd} - \mathsf{Cha} \cdot \xi \pmod{q}.$$

The SPK is then given by the pair $(\mathsf{Cha}, \mathsf{Rsp})$, and hence verification is performed by checking whether $\mathsf{Cha} = \mathcal{H}\left(\phi \| A \| (B^{\mathsf{Rsp}} \cdot A^{\mathsf{Cha}}) \| m\right)$. Thus, a signature consists of three elements in $\mathbb{G}_1$ (d,e, and f) and two elements in $\mathbb{Z}_q$ (Cha and Rsp).

We now turn to computational cost, which we denote by the following type of expression $1 \cdot P^2 + 2 \cdot P + 3 \cdot \mathbb{G}_T^2 + 1 \cdot \mathbb{G}_1$ denotes a cost of one product of two pairing values, two pairings, three multi-exponentiations in $\mathbb{G}_T$ with two terms, and one exponentiation in $\mathbb{G}_1$. Unfortunately, it is very hard to assign conversion factors between the different operations. The reason being that such factors heavily depend, for example, on the elliptic curve underlying a scheme, on the security parameters, or even optimisation of the implementation. Still, to provide a better readability we sort the operations with presumably decrementing complexity and cost.

With the above formulation of the required SPK, the cost for signing would be $2 \cdot \mathbb{G}_T + 3 \cdot \mathbb{G}_1$, since $\hat{e}(a,\tilde{x}), \hat{e}(b,\tilde{x})$ and $\hat{e}(c,\tilde{g})$, can be precomputed. Now we want to optimize the computation of the hash to further shorten the computation time of signing. Keep in mind that doing so, would require corresponding changes to the proofs as well. In our case, the change for optimization actually simplifies the proof. Thus, if we adapt the computation of the challenge to include d, e and f instead of A, i.e., $\mathsf{Cha} \leftarrow \mathcal{H}(\phi \| d \| e \| f \| \mathsf{Comm} \| m)$, the signer does not need to compute A. Consequently, the cost for signing accounts to $1 \cdot \mathbb{G}_T + 3 \cdot \mathbb{G}_1$. Note that this slight change in the computation of the challenge not only benefits the signer but also the verifier of the signature saves on computational costs.

Verification in our scheme requires to check whether $\hat{e}(d,\tilde{y}) = \hat{e}(e,\tilde{g})$ holds. This is computed as one product of two pairings, which is more efficient than computing two pairings separately. In addition, verification consists of verifying the SPK, which amounts to $1 \cdot P^2 + 1 \cdot \mathbb{G}_1^2 + 1 \cdot \mathbb{G}_1$, assuming the calculation of the verification value as $\hat{e}(f^c,\tilde{g})/\hat{e}(d^c e^{s_\xi},\tilde{x})$. Note that we use here the adapted

computation of the challenge as described before. We apply similar changes to the calculation of the challenge value for all schemes in our comparison to reduce the required numbered of computations. In addition, we assume precomputation of pre-known values.

To have an conclusive comparison of the verification costs, it is essential to know the exact cost for each operation. This follows from the fact that there exist various possibilities to verify a NIZK proof. For example, verification of the BBS* (as described in Appendix A) NIZK proof requires the calculation of the following value

$$\left(\frac{\hat{e}(u,v)}{\hat{e}(T_3,w)}\right)^c \cdot \frac{\hat{e}(T_3,v)^x \hat{e}(h,v)^y}{\hat{e}(e,w)^r \hat{e}(e,v)^\delta}$$

where u, v, w, h, and e are pre-known. We can calculate this value as

$$\frac{\hat{e}(u^c T_3^x h^y e^{-\delta}, v)}{\hat{e}(T_3^c e^r, w)} \qquad \text{or as} \qquad \hat{e}(T_3, v^x w^{-c}) \frac{\hat{e}(u,v)^c \hat{e}(h,v)^y}{\hat{e}(e,w)^r \hat{e}(e,v)^\delta} \ ,$$

where the first computation amounts to $1 \cdot P^2 + 1 \cdot \mathbb{G}_1^4 + 1 \cdot \mathbb{G}_1^2$ operations and the the second one accounts for $1 \cdot P + 1 \cdot \mathbb{G}_T^4 + 1 \cdot \mathbb{G}_2^2$ operations. We can see that a direct comparison of those different methods of computing the same value is very hard. Such difficulties, however, mostly arise in the verification equation and make the verification costs less transparent. Still, the numbers in Table 1 show that all compared schemes require similar computation efforts in verification.

We now compare our scheme with the current best schemes w.r.t. signature length. We only consider pairing-based schemes as RSA-based schemes need much larger groups to attain the same security level. Consequently, we can focus on just a small number of schemes.

- The CL scheme from [17] shares many similarities with our own. The basic security is based on the LRSW and the DDH assumption in $\mathbb{G}_T$. The basic construction is in the case of Type-1 pairings, and combines the CL-signature scheme with a Cramer-Shoup encryption. It is this Cramer-Shoup based component which creates the main divergence from our own scheme. Translating the construction to the Type-2 or Type-3 setting we obtain a more efficient construction based on the LRSW and the XDDH assumption.

- The DP scheme of Delerablée and Pointcheval [26] is based on the XDDH assumption and q-SDH. It is shown to provide full-anonymity under the XDDH assumption w.r.t. the so-called CCA attack, which is achieved by combining two ElGamal encryptions. The scheme is also shown to provide full-traceability under the q-SDH assumption.

- The BBS group signature scheme [10] is similar to the DP scheme [26]. However, it provides full-anonymity under the DLIN assumption only with respect to a so-called CPA attack (i.e., the adversary is not allowed to make any Open oracle queries). As we strive to provide a comparison between

systems that have similar security guarantees, we consider a variant of the BBS scheme that we call BBS*. The modification is obtained from the BSS scheme by following remarks in that same paper to obtain exculpability. As all schemes that we compare are secure under XDDH, we can use standard Cramer-Shoup encryption instead of the linear variant proposed in [33] which is secure also if XDDH does not hold. We provide the details of BBS* in Appendix A.

We summarize the efficiency discussion in Table 1. Note that all schemes provide anonymity w.r.t. the CCA attack, are based on the random oracle model, and provide strong exculpability. As pointed out in the discussion before, they use slightly different underlying assumptions, namely q-SDH or LRSW. A further difference is that our scheme, as opposed to the schemes we compare against, combines Group Manager and Opener into one entity.

Table 1 shows that our scheme compares favourably with the other schemes in the signature length and the signature generation operation. In particular, it reduces the signature size by almost a factor of two. Comparing verification costs shows all schemes on an approximately equal level. Note that short signatures and small signature computation costs are particularly interesting as there are many scenarios where the group signature has to be generated and communicated by a resource constrained device.

Scheme	Size of Sig.		Sign Cost						Verification Cost							
	$\mathbb{G}_1$	$\mathbb{Z}_q$	$\mathbb{G}_T^5$	$\mathbb{G}_T^3$	$\mathbb{G}_T^2$	$\mathbb{G}_T$	$\mathbb{G}_1^2$	$\mathbb{G}_1$	P^2	P	$\mathbb{G}_T^3$	$\mathbb{G}_2^2$	$\mathbb{G}_1^4$	$\mathbb{G}_1^3$	$\mathbb{G}_1^2$	$\mathbb{G}_1$
Ours	3	2				1		3	2						1	1
CL	7	4			1		1	11	2			1		2	2	1
DP	4	5		1			1	6		1	1	1		1	2	
BBS*	4	5	1				3	5	1				1	1	4	

Table 1: Comparison of signature lengths, signature generation costs and signature verification costs.

We end this section by considering our VLR group signature variant as explained in Section 3.2. This version of our scheme has the same signature size as above, namely $3 \cdot \mathbb{G}_1 + 2 \cdot \mathbb{Z}_q$. The signing cost is also the same, namely $1 \cdot \mathbb{G}_T + 3 \cdot \mathbb{G}_1$, but verification includes the opening computations for all revoked users, thus verification requires time

$$|\mathbf{rlist}| \cdot P + 3 \cdot P^2 + 1 \cdot P + 1 \cdot \mathbb{G}_T^2.$$

As noted previously the BS VLR group signature scheme from [11] requires Type-4 pairings to implement it, as is explained in [32]. Security in their scheme is based on the q-SDH and DLIN assumptions, with DLIN being required to prove selfless-anonymity. A signature requires two elements in $\mathbb{G}_1$ and five elements in

$\mathbb{Z}_q$ and this can be computed in $1 \cdot \mathbb{G}_T^3 + 1 \cdot \mathbb{G}_1^2 + 3 \cdot \mathbb{G}_1$. Verification costs depend on the size of the revocation list $|\mathbf{rlist}|$, and are given by

$$(2 + |\mathbf{rlist}|) \cdot P + 1 \cdot \mathbb{G}_T + 2 \cdot \mathbb{G}_2^2 + 2 \cdot \mathbb{G}_1^2.$$

These are slightly faster times than those quoted in [11] as we assume an efficient Type-4 representation of $\mathbb{G}_2$ is used. Note we have not counted the cost of hashing into $\mathbb{G}_2$ which could be expensive depending on the precise elliptic curve chosen. However, we note that our scheme is significantly more efficient in terms of bandwidth and computational resources than that of [11], even before considering the time needed to hash onto the Type-4 $\mathbb{G}_2$ group in the BS scheme.

7 Acknowledgements

The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II and ICT-2007-216483 PRIMELIFE. The fourth author was supported by a Royal Society Wolfson Merit Award and a grant from Google. All authors would like to thank the referee's of a prior version of this paper.

References

[1] Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. In Javier López, Sihan Qing, and Eiji Okamoto, editors, *ICICS 04*, volume 3269 of *LNCS*, pages 1–13, Malaga, Spain, October 27–29, 2004. Springer, Berlin, Germany.

[2] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 255–270, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.

[3] Giuseppe Ateniese, Dawn Xiaodong Song, and Gene Tsudik. Quasi-efficient revocation in group signatures. In Matt Blaze, editor, *FC 2002*, volume 2357 of *LNCS*, pages 183–197, Southampton, Bermuda, March 11–14, 2002. Springer, Berlin, Germany.

[4] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Germany.

[5] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and

Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 390–399, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[6] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[7] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153, San Francisco, CA, USA, February 14–18, 2005. Springer, Berlin, Germany.

[8] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 1–12, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Berlin, Germany.

[9] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

[10] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

[11] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04*, pages 168–177, Washington D.C., USA, October 25–29, 2004. ACM Press.

[12] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.

[13] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 1–15, Beijing, China, April 16–20, 2007. Springer, Berlin, Germany.

[14] Ernie Brickell, Liqun Chen, and Jiangtao Li. A new direct anonymous attestation scheme from bilinear maps. In *Trusted Computing - Challenges and Applications – TRUST 2008*, volume 4968 of *LNCS*, pages 1–20. Springer-Verlag, 2008.

[15] Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the portability of generalized schnorr proofs. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 425–442, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.

[16] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[17] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

[18] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 410–424, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.

[19] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265, Brighton, UK, April 8–11, 1991. Springer, Berlin, Germany.

[20] Liqun Chen, Michael Cheng, and Nigel P. Smart. Identity-based key agreement protocols from pairings. *International Journal of Information Security*, 6:213–241, 2007.

[21] Liqun Chen, Paul Morrissey, and Nigel P. Smart. Pairings in trusted computing (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 1–17, Egham, UK, September 1–3, 2008. Springer, Berlin, Germany.

[22] Liqun Chen, Paul Morrissey, and Nigel P. Smart. DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198, 2009. `http://eprint.iacr.org/`.

[23] Ronald Cramer, Ivan Damgård, and Philip D. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 354–372, Melbourne, Victoria, Australia, January 18–20, 2000. Springer, Berlin, Germany.

[24] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[25] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Germany.

[26] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210, Hanoi, Vietnam, September 25–28, 2006. Springer, Berlin, Germany.

[27] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Germany.

[28] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156:3113–3121, 2008.

[29] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[30] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199, Kingston, Ontario, Canada, August 9–10, 2000. Springer, Berlin, Germany.

[31] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[32] Hovav Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, 2005.

[33] Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/`.

[34] Gene Tsudik and Shouhuai Xu. Accumulating composites and improved group signing. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 269–286, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Berlin, Germany.

A Sketch of BBS*

This variant of the BBS group signature scheme is based on remarks by Boneh et al. [10] (general scheme and non-frameability) and [33] (CCA anonymity). In

particular, the variant we consider attains exculpability by an interactive protocol between Group Manager and user for the joint computation of a triple (A_i, x_i, y_i) such that $A_i^{x_i+\gamma} h^{y_i} = u$. Here y_i is secret to the user, γ is the Group Manager's secret, and $u, h \in \mathbb{G}_1$ are public parameters. Given that our scheme assumes XDDH, we employ standard Cramer-Shoup encryption [24] instead of the linear Cramer-Shoup encryption proposed by Shacham [33].

Let us now provide details of the BBS* scheme. The setup and key generation algorithms produce $u, v \leftarrow \mathbb{G}_1$ and $c \leftarrow u^{\chi_1} v^{\chi_2}, d \leftarrow u^{\mu_1} v^{\mu_2}$, as well as $e \leftarrow v^{\iota}$. As the results of the join protocol, each user gets a tuple (A_i, x_i, y_i) fulfilling $A_i^{x_i+\gamma} h^{y_i} = u$ and the Group Manager uses his secret value γ to compute $w \leftarrow v^{\gamma}$. The group public key consists of (u, v, c, d, e, h, w) and the secret key of the Opener contains $(\chi_1, \chi_2, \mu_1, \mu_2)$.

To sign a message, user i chooses $r \leftarrow \mathbb{Z}_q$, $\delta \leftarrow r \cdot x_i$, and computes $T_1 \leftarrow u^r$, $T_2 \leftarrow v^r$, $T_3 \leftarrow e^r A_i$, $T_4 \leftarrow c^r d^{r\mathcal{H}(T_1,T_2,T_3)}$. Moreover, she computes the proof

$$\Sigma \leftarrow SPK\{(r, x_i, \delta, y_i) : T_1 = u^r \wedge T_2 = v^r \wedge T_4 = c^r d^{r\mathcal{H}(T_1,T_2,T_3)} \wedge 1 = T_1^{x_i} u^{-\delta} \wedge \frac{\hat{e}(u,v)}{\hat{e}(T_3,w)} = \frac{\hat{e}(T_3,v)^{x_i} \hat{e}(h,v)^{y_i}}{\hat{e}(e,w)^r \hat{e}(e,v)^{\delta}}\} \ ,$$

and outputs the signature $\sigma \leftarrow (T_1, T_2, T_3, T_4, \Sigma)$.

The verification of a signature consists of checking the validity of the proof Σ. Opening a signature can be performed by the Opener using his secret key to decrypt the Cramer-Shoup encryption of the value A_i.

B Security Proofs

B.1 Proof of Theorem 5.1

We show that a simulator $\mathcal{B}$, given an adversary $\mathcal{A}$ having a non-negligible advantage in the anonymity game $\mathbf{Adv}^{\text{anon}}_{\mathsf{GS},A}(\eta) \geq \epsilon$, can solve a DDH problem in $\mathbb{G}_1$. We denote the DDH challenge given to $\mathcal{B}$ as $(g_0, g_1, g_2, g_3) = (g, g^{\mu}, g^{\nu}, g^{\omega})$, where $\mathcal{B}$ will output a guess δ' indicating whether $\omega \leftarrow \mu\nu$, i.e., $\delta = 1$, or $\omega \leftarrow \mathbb{Z}_q$, i.e., $\delta = 0$.

The proof idea is to let honest users sign with signatures on a secret value $\xi_i = \mu r_i$, where $r_i \leftarrow \mathbb{Z}_q$. More concretely, $\mathcal{B}$ generates $(gpk, gmsk)$ in the normal way and creates a tuple $(a_i, b_i, c_i) = (g^{\rho_i}, a_i^{\beta}, a_i^{\alpha+\alpha\beta\mu r_i})$ for each honest user i. When $\mathcal{A}$ asks the simulator for a challenge signature, $\mathcal{B}$ uses its knowledge of $gmsk$ and $\mathbf{gsk}[i], \forall i \in \mathcal{HU}$ to create a signature of the form $(d^*, e^*, f^*) = (g^{\nu}, g^{\nu\beta}, g^{\nu\alpha}(g^{\omega})^{\alpha\beta r_i})$. This constitutes a valid group signature for a user with $\xi_i = \mu r_i$ and randomization parameter ν, assuming $\mathcal{B}$ has been given a DDH tuple with $\omega = \mu\nu$. Consequently, $\mathcal{A}$ only has an advantage in solving the anonymity game, if the DDH challenge was a correct DDH tuple. Otherwise, $\mathcal{A}$ does not gain any information from the given signature.

In more detail, given the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, $\mathcal{B}$ retrieves the sets $\mathcal{HU}, \mathcal{DU} \subseteq \{1, \ldots, n\}$ from $\mathcal{A}$. He uses $g \leftarrow g_0$, selects a generator $\tilde{g} \in \mathbb{G}_2$ as well as $\alpha, \beta \leftarrow \mathbb{Z}_q$, and computes $\tilde{x} \leftarrow \tilde{g}^{\alpha}$ and $\tilde{y} \leftarrow \tilde{g}^{\beta}$. Then, he sets the group manager secret key $gmsk \leftarrow (\alpha, \beta)$ and the group public key $gpk \leftarrow (g, \tilde{g}, \tilde{x}, \tilde{y})$, which he supplies to $\mathcal{A}$.

For all honest users $i \in \mathcal{HU}$, the simulator $\mathcal{B}$ generates a PKI key pair $(\mathbf{usk}[i], \mathbf{upk}[i]) \leftarrow \mathsf{PKIJoin}(i, 1^{\eta})$. Then $\mathcal{B}$ chooses $\rho_i, r_i \leftarrow \mathbb{Z}_q$ and calculates their group signing key as $(a_i, b_i, c_i) = (g_0^{\rho_i}, a^{\beta}, a^{\alpha} g_1^{\rho_i \alpha \beta r_i})$. Note, that this tuple has the same distribution as the one specified by the protocol in Section 4. Moreover, $\mathcal{B}$ chooses a random $k_i \in \mathbb{G}_T$ and computes the signature $\bar{\sigma}_i \leftarrow \mathsf{DSSign}(\mathbf{usk}[i], k_i$, which is distributed as in the proposed scheme. He stores $(a_i, b_i, c_i, r_i, k_i, \bar{\sigma}_i)$ in $\mathbf{gsk}[i]$.

Then $\mathcal{B}$ runs $\mathcal{A}$ and simulates the oracle queries as follows:

- $\mathcal{G}(\kappa)$: $\mathcal{B}$ maintains a list $L_{\mathcal{G}}$ of previous random oracle responses storing (κ, t). It returns t to each query κ, assigning a fresh random value $t \leftarrow \{0,1\}^{\ell}$ if (κ, t) is undefined.

- $\mathcal{H}(S)$: $\mathcal{B}$ maintains a list $L_{\mathcal{H}}$ of previous random oracle responses storing (S, Cha). It returns Cha to each query S, assigning a fresh random value $\mathsf{Cha} \leftarrow \{0,1\}^{\ell}$ if (S, Cha) is undefined.

- $\mathsf{SetUPK}(i, upk)$: $\mathcal{B}$ executes the role of the CA and publishes $\mathbf{upk}[i] \leftarrow upk$. Note that everyone can get an authentic copy of $\mathbf{upk}$ (which can be achieved by letting the simulator create a signature on the user public key using the CA's public key).

- $\mathsf{GJoin}_{DM}(i)$: For dishonest users $i \in \mathcal{DU}$, $\mathcal{B}$ simulates the Group Manager's side of the protocol as prescribed in the scheme in Section 4 using the knowledge of $gmsk$. The simulator stores the local output $(\tilde{w}, \tilde{r}, \kappa, \bar{\sigma})$ in $\mathbf{reg}[i]$.

- $\mathsf{GSign}(i, m)$: Given user $i \in \mathcal{HU}$ and a message m from $\mathcal{A}$, $\mathcal{B}$ retrieves $\mathbf{gsk}[i] = (a_i, b_i, c_i, r_i, k_i, \bar{\sigma}_i)$.

  It chooses $\zeta \leftarrow \mathbb{Z}_q$ to re-randomize the group signing key to obtain $(d, e, f) = (a_i^{\zeta}, b_i^{\zeta}, c_i^{\zeta})$. By programming the random oracle $\mathcal{H}(\cdot)$, $\mathcal{B}$ can simulate the signature of knowledge Σ and returns $\sigma = (a, b, c, \Sigma)$ to $\mathcal{A}$. He adds (i, m, σ) to a list $\mathbf{sgn}$.

- $\mathsf{GOpen}(m, \sigma)$: As the signature can originate from an honest or a dishonest user, the simulator distinguishes two cases.

  1. If the signature stems from a dishonest user, it can be opened and the proof π created using the information from $\mathbf{reg}$ as in the real scheme.
  2. In case the signature was created on the behalf of an honest user, then either σ is a previous output of the $\mathsf{GSign}(\cdot, \cdot)$ oracle produced by $\mathcal{B}$, or

it is a forgery produced by $\mathcal{A}$. The latter case is excluded by the non-frameability property of Theorem 5.3. In the former case, $\mathcal{B}$ can simply look up the corresponding user i from the list **sgn** using the message m and σ, and look up the corresponding tuple $\mathbf{gsk}[i] = (a_i, b_i, c_i, r_i, k_i, \bar{\sigma}_i)$. Then, by programming the random oracle $\mathcal{H}(\cdot)$ $\mathcal{B}$ simulates a signature of knowledge

$$\Pi \leftarrow SPK\{(\tilde{w}_i, \kappa) : \frac{\hat{e}(f, \tilde{g})}{\hat{e}(d, \tilde{x})} = \hat{e}(e, \tilde{w}_i) \wedge k_i = \frac{\hat{e}(g, \tilde{w}_i)}{\hat{e}(g, \tilde{x})^{\kappa}}\} \ .$$

He sends $(i, \pi = (k_i, \bar{\sigma}, \Pi))$ to $\mathcal{A}$.

- $\mathsf{Ch}(\cdot, i_0, i_1, m)$: First, $\mathcal{B}$ chooses $b \leftarrow \{0, 1\}$ and looks up the group signing key $\mathbf{gsk}[i_b] = (a_{i_b}, b_{i_b}, c_{i_b}, r_{i_b}, k_{i_b}, \bar{\sigma}_{i_b})$. Secondly, he constructs

$$(d^*, e^*, f^*) \leftarrow (g_2, g_2^{\beta}, g_2^{\alpha} g_3^{\alpha\beta r_i}) \ .$$

Assuming that the given DDH challenge is a DDH tuple, i.e., $\delta = 1$, implies $f^* = g_0^{\alpha\nu + \alpha\beta\nu\mu r_i}$, which has the same distribution compared to a real signature tuple. If $\delta = 0$, f^* is uniformly distributed in $\mathbb{G}_1$, independent of the choice of b. Finally, $\mathcal{B}$ simulates the signature of knowledge Σ by programming the random oracle $\mathcal{H}(\cdot)$.

At the end of its execution, the adversary $\mathcal{A}$ will output a guess b'. The simulator $\mathcal{B}$ outputs $\delta' = 1$ if the adversary $\mathcal{A}$ output $\gamma' = \gamma$, and outputs $\delta' = 0$ otherwise. We calculate the advantage of $\mathcal{B}$ in solving the DDH challenge as

$$\mathbf{Adv}_{\mathcal{B}}^{XDDH} = \Pr[\delta' = 1 | \delta = 1] - \Pr[\delta' = 1 | \delta = 0].$$

When $\delta = 0$, f^* is a uniformly distributed value in $\mathbb{G}_1$ independent of b, so that $\mathcal{A}$ outputs $b' = b$ with probability $\Pr[b' = b | \delta = 0] = \frac{1}{2}$. As $\mathcal{B}$ guesses $\delta' = 1$ when $b' = b$, we get:

$$\Pr[\delta' = 1 | \delta = 0] = \frac{1}{2} \ .$$

If $\delta = 1$ then the challenge signature is identically distributed as in a real attack scenario. Due to $\mathcal{B}$'s choice of δ' we see that $\Pr[\delta' = 1 | \delta = 1] = \Pr[b' = b | \delta = 1]$, which stands for the adversary $\mathcal{A}$ winning the anonymity game. Thus,

$$\begin{aligned} \Pr[b' = b | \delta = 1] &= \frac{1}{2} \left(\Pr[\mathbf{Exp}_{\mathsf{GS},A}^{\text{anon-1}}(\eta) = 1] + \Pr[\mathbf{Exp}_{\mathsf{GS},A}^{\text{anon-0}}(\eta) = 0] \right) \\ &= \frac{\mathbf{Adv}_{\mathsf{GS},A}^{\text{anon}}(\eta) + 1}{2}, \end{aligned}$$

lets us conclude that:

$$\mathbf{Adv}_{\mathcal{B}}^{XDDH} \geq \frac{\epsilon}{2} \ .$$

B.2 Proof of Theorem 5.2

We prove that given any adversary $\mathcal{A}$ winning the traceability game, one can construct an adversary $\mathcal{B}$ breaking the existential unforgeability of CL signatures. The theorem then follows from the security proof of CL signatures [17].

In a first step, we transform adversary $\mathcal{A}$ into an adversary $\mathcal{A}'$ to which the general forking lemma due to Bellare and Neven [5] can be applied; we recall the lemma in Appendix C. The lemma will then yield a forking algorithm $\mathcal{F}_{\mathcal{A}'}$ that produces two different but related untraceable group signatures, based on which $\mathcal{B}$ can compute a forgery for the CL signature scheme.

Given adversary $\mathcal{A}$, consider the following algorithm $\mathcal{A}'$. On input $(\tilde{x}, \tilde{y})$ and random tape R, it runs $\mathcal{A}$ on input $gpk = (\tilde{x}, \tilde{y})$ and random tape R' derived from R, simulating its oracle queries as follows while maintaining a counter ctr and lists $L_{\mathcal{G}}$, $L_{\mathcal{H}}$, and an associative array **reg**:

- $\mathcal{G}(\kappa)$: $\mathcal{A}'$ looks up a tuple (κ, t) in the list $L_{\mathcal{G}}$ and returns t; if no such tuple is found, it chooses a random value $t \leftarrow \{0,1\}^{\ell}$ and adds (κ, t) to $L_{\mathcal{G}}$.

- $\mathcal{H}(S)$: $\mathcal{A}'$ looks up a tuple (S, j, Cha) in the list $L_{\mathcal{H}}$ and returns Cha. If no such tuple is found, it increases ctr, chooses a random value $\mathsf{Cha} \leftarrow \{0,1\}^{\ell}$ and adds (S, ctr, Cha) to $L_{\mathcal{H}}$.

- $\mathsf{SetUPK}(i, upk)$: $\mathcal{A}'$ sets $\mathbf{upk}[i] \leftarrow upk$ as the certified public key associated to user i.

- $\mathsf{GJoin}_{DM}(i)$: $\mathcal{A}'$ chooses $\kappa \leftarrow \mathbb{Z}_q$, computes $t \leftarrow \mathcal{G}(\kappa)$, and sends t to $\mathcal{A}$. After receiving $(s, \tilde{r}, \bar{\sigma})$, $\mathcal{A}'$ rewinds $\mathcal{A}$ to extract τ from the $FPK\{(\tau)\}$; when it fails, which only happens with probability of the knowledge error $1/q$, $\mathcal{A}'$ halts with output $(0, \varepsilon)$. Otherwise, it computes $\xi \leftarrow \tau + \kappa \bmod q$ and queries its signing oracle for a CL signature (a, b, c) on message ξ. It then sends (a, b, c, κ) to $\mathcal{A}$ and uses the zero-knowledge simulator to simulate $FPK\{(\alpha, \beta, \rho, \gamma)\}$, which it can do without any probability loss due to the perfect zero-knowledge property. Finally, it saves the tuple $(\tilde{w} \leftarrow \tilde{x}^{\xi}, \tilde{r}, \kappa, \bar{\sigma})$ in $\mathbf{reg}[i]$.

- $\mathsf{GOpen}(m, \sigma)$: If $\mathsf{GVerify}(gpk, m, \sigma) = 0$ then $\mathcal{A}'$ returns $\perp$. Else, it parses σ as (d, e, f, Σ) and looks for a tuple $(\tilde{w}, \tilde{r}, \kappa, \bar{\sigma}) \in \mathbf{join}$ such that $\hat{e}(f, \tilde{g}) = \hat{e}(d, \tilde{x}) \cdot \hat{e}(e, \tilde{w})$. If such a tuple is found, it constructs a proof π using values $\tilde{w}$, $\tilde{r}$ and κ as in the real GOpen algorithm and returns $(i, (k, \bar{\sigma}, \pi))$.

When $\mathcal{A}$ outputs its forgery $(m, \sigma = (d, e, f, (\mathsf{Cha}, \mathsf{Rsp}))$ we distinguish two cases depending on the validity of a forgery. If the forgery is invalid, meaning that $\mathsf{GVerify}(gpk,\ m, \sigma) = 0$ or σ can be opened by the procedure described in the GOpen oracle using one of the $\tilde{w}$ values in $\mathbf{join}$, then $\mathcal{A}'$ halts with output $(0, \varepsilon)$. Otherwise, it looks up the index j so that $(S, j, \mathsf{Cha}) \in L_{\mathcal{H}}$ for $S = A \| B \| C \| m$ where $A = \hat{e}(f, \tilde{g})/\hat{e}(d, \tilde{x})$, $B = \hat{e}(e, \tilde{x})$, and $C = A^{\mathsf{Cha}} B^{\mathsf{Rsp}}$. Such a tuple must

exist, since at the very latest $\mathcal{A}'$ would have forced its creation during the final verification $\mathsf{GVerify}(gpk, m, \sigma)$. We call the j-th $\mathcal{H}(\cdot)$ query made by $\mathcal{A}$ the "crucial" hash query. $\mathcal{A}'$ halts with output (j, σ).

Consider the general forking lemma with algorithm $\mathcal{A}'$ and an input generator $\mathcal{IG}$ that outputs gpk. If $\mathcal{A}$ wins the traceability game with probability ϵ, i.e., $\mathbf{Adv}^{\text{trace}}_{\mathsf{GS},\mathcal{A}}(\eta) = \epsilon$, then the probability acc that $\mathcal{A}'$ outputs (j, σ) with $j \geq 1$ is

$$acc \;\geq\; \epsilon - n/q \;,$$

where the latter term is due to premature halting because of an extraction failure during any of the GJoin protocols.

By applying the general forking lemma, we obtain an algorithm $\mathcal{F}_{\mathcal{A}'}$ that with probability

$$frk \;\geq\; \frac{acc^2}{q_{\mathrm{H}}} - \frac{1}{q} \;\geq\; \frac{\epsilon^2}{q_{\mathrm{H}}} - \frac{3n^2+1}{q} \tag{1}$$

outputs a tuple $(1, \sigma_1, \sigma_2)$, where q_{H} is (at most) the number of $\mathcal{H}(\cdot)$ queries made by $\mathcal{A}$.

Based on this algorithm $\mathcal{F}_{\mathcal{A}'}$, consider algorithm $\mathcal{B}$ that forges CL signatures by running $\mathcal{F}_{\mathcal{A}'}$ to obtain two signatures $\sigma_1 = (d_1, e_1, f_1, (\mathsf{Cha}_1, \mathsf{Rsp}_1))$ and $\sigma_2 = (d_2, e_2, f_2, (\mathsf{Cha}_2, \mathsf{Rsp}_2))$. Let

$$\begin{array}{llllll} A_1 & = \hat{e}(f_1, \tilde{g})/\hat{e}(d_1, \tilde{x}) & B_1 & = \hat{e}(e_1, \tilde{x}) & C_1 & = A_1^{\mathsf{Cha}_1} B_1^{\mathsf{Rsp}_1} \\ A_2 & = \hat{e}(f_2, \tilde{g})/\hat{e}(d_2, \tilde{x}) & B_2 & = \hat{e}(e_2, \tilde{x}) & C_2 & = A_2^{\mathsf{Cha}_2} B_2^{\mathsf{Rsp}_2} \;. \end{array}$$

Since the two executions of $\mathcal{A}$ are identical in inputs, random tape, and oracle responses up to the point where the "crucial" hash queries are made, the arguments of these hash queries must be identical too, so that $A_1 = A_2$, $B_1 = B_2$, and $C_1 = C_2$. Because $B_1 = B_2$ we have that $e_1 = e_2$. Since both signatures are valid we have that $\hat{e}(d_1, \tilde{y}) = \hat{e}(e_1, \tilde{g})$ and $\hat{e}(d_2, \tilde{y}) = \hat{e}(e_2, \tilde{g})$, so that $d_1 = d_2$. Finally, because $A_1 = A_2$ we also have that $f_1 = f_2$. The forking algorithm however guarantees us that the responses to these queries Cha_1 and Cha_2 are different and smaller than q, so that $\mathsf{Cha}_1 - \mathsf{Cha}_2 \neq 0 \bmod q$. By putting the equations for C_1 and C_2 together one can see that $\xi = (\mathsf{Rsp}_2 - \mathsf{Rsp}_1)/(\mathsf{Cha}_1 - \mathsf{Cha}_2) \bmod q$ satisfies the equation $\hat{e}(f_1, \tilde{g})/\hat{e}(d_1, \tilde{x}) = \hat{e}(e_1, \tilde{x})^{\xi}$, which is the second verification equation of CL signatures. The validity of the group signature σ_1 ensures that the first CL verification equation is also satisfied, so that (d_1, e_1, f_1) is a valid CL signature on message ξ. Moreover, ξ does not occur in a tuple of **reg**, because in that case the opening of σ_1 at the end of the execution of $\mathcal{A}'$ would have succeeded. Since the only messages ξ for which $\mathcal{B}$ previously queried CL signatures are those occurring in **reg**, $\mathcal{B}$ can output $(\xi, (d_1, e_1, f_1))$ as its own forgery; its overall probability of doing so is at least the probability frk depicted in Equation (1).

B.3 Proof of Theorem 5.3

The goal of the adversary $\mathcal{A}$ in the non-frameability game is to create a group signature σ on a message m together with a valid proof π that attributes σ to an

honest user i even though σ was never output by the GSign oracle on inputs i, m. We distinguish between two types of attacks. In the first type, the value for k in $\pi = (k, \bar{\sigma}, \Pi)$ is different from the value that user i signed during the join protocol. It is easy to see that this type of attack is impossible by the unforgeability of the underlying signing algorithm DSSign, as $\bar{\sigma}$ is a valid signature under $\mathbf{upk}[i]$ that was never signed by user i. We now focus on the second type of attacks, where the value for k in π is the same as signed by user i when joining.

We construct a simulator $\mathcal{B}$ that solves a given SDLP problem $(g_1, \tilde{g}_1) = (g^\mu, \tilde{g}^\mu)$. Note, that w.l.o.g. we assume that the bases in the SDLP problem match the bases that the group signature scheme is using, as an SDLP problem using different bases can be transformed into the problem denoted before. The simulator makes use of an adversary $\mathcal{A}$ having a non-negligible advantage in the non-frameability game $\mathbf{Adv}^{\mathrm{nf}}_{\mathsf{GS},A}(\eta) \geq \epsilon$. The simulator's output is ε, if he was not able to solve the DL problem, or μ being the solution to the problem.

The proof idea is to create an algorithm $\mathcal{A}'$ to which we will apply the general forking lemma. Through the lemma we attain an algorithm $\mathcal{F}_{\mathcal{A}'}$. This algorithm will output two signatures (σ_0, σ_1) that are related such that the simulator can extract the "message" that they have been issued upon. Assuming that $\mathcal{B}$ manages to construct those messages dependent on μ will allow him to solve the SDLP problem.

In more detail, the algorithm $\mathcal{A}'$ gets gpk and a random tape R. It derives R' from R and runs $\mathcal{A}$ on input (gpk, R'). Algorithm $\mathcal{A}'$ maintains a counter ctr and lists $L_{\mathcal{G}}$, $L_{\mathcal{H}}$.

- $\mathcal{G}(\kappa)$: $\mathcal{A}'$ looks up a tuple (κ, t) in the list $L_{\mathcal{G}}$ and returns t; if no such tuple is found, it chooses a random value $t \leftarrow \{0,1\}^\ell$ and adds (κ, t) to $L_{\mathcal{G}}$.

- $\mathcal{H}(S)$: $\mathcal{A}'$ looks up a tuple (S, j, Cha) in the list $L_{\mathcal{H}}$ and returns Cha. If no such tuple is found, it increases ctr, chooses a random value $\mathsf{Cha} \leftarrow \mathbb{Z}_q$ and adds (S, ctr, Cha) to $L_{\mathcal{H}}$.

- $\mathsf{SetUPK}(i, upk)$: $\mathcal{A}'$ sets $\mathbf{upk}[i] \leftarrow upk$ as the certified public key associated to user i.

- $\mathsf{GJoin}_{UD}(i)$: Given an honest user $i \in \mathcal{HU}$, $\mathcal{A}'$ extracts the value of κ from the random oracle $\mathcal{G}(\cdot)$ by looking for a pair $(\kappa, t) \in L_{\mathcal{G}}$. (For large enough values of ℓ, exactly one such pair must exist, because otherwise $\mathcal{A}$ must either have created a collision on $\mathcal{G}$, or predicted an output of $\mathcal{G}$ before querying it.) Then he chooses $r_i \leftarrow \mathbb{Z}_q$ and computes $s \leftarrow g_1^{r_i}/(g^\kappa)$ as well as $\tilde{r} \leftarrow \tilde{g}_1^{r_i}/(\tilde{g}^\kappa)$. Then he computes the signature $\bar{\sigma} \leftarrow \hat{e}(g, \tilde{r})$ and sends $(s, \tilde{r}, \bar{\sigma})$ to $\mathcal{A}$. The algorithm simulates the proof of knowledge without any probability loss due to the perfect zero-knowledge property. $\mathcal{A}$ will supply the value of κ and will prove the correct computation of the values (a, b, c). Through the verification of this proof, $\mathcal{A}'$ is assured that the received signature constitutes a signature on $\xi = \mu r_i$. $\mathcal{A}'$ stores (ξ, a, b, c) in $\mathbf{gsk}[i]$.

- $\mathsf{GSign}(i, m)$: Given user $i \in \mathcal{HU}$ and a message m from $\mathcal{A}$, $\mathcal{A}'$ retrieves the signing key from $\mathbf{gsk}$. If $\mathbf{gsk}[i] = \perp$ does not exists, then $\mathcal{A}'$ return $\perp$. Otherwise, it computes $\sigma \leftarrow \mathsf{GSign}(\mathbf{gsk}[i], m)$, adds (m, σ) to the list $\mathbf{sgn}$, and returns σ.

Algorithm $\mathcal{A}$ will output the forged signature $\sigma = (d, e, f, (\mathsf{Cha}, \mathsf{Rsp}))$ on a message m, a proof π and an i indicating for which user $\mathcal{A}$ forged the signature. If any of $\mathsf{GVerify}(gpk, m, \sigma) = 0$, $i \notin \mathcal{HU}$, $(m, (d, e, f)) \in \mathbf{sgn}$, or $\mathsf{GJudge}(m, \sigma, i, \mathbf{upk}[i], \pi) = 0$ fails, then $\mathcal{A}'$ will halt and output $(0, \varepsilon)$.

Otherwise, $\mathcal{A}'$ looks up the tuple $(A \| B \| C \| m, j, \mathsf{Cha})$ where $A = \hat{e}(f, \tilde{g})/\hat{e}(d, \tilde{x})$, $B = \hat{e}(e_1, \tilde{x})$, and $C = A^{\mathsf{Cha}} B^{\mathsf{Rsp}}$. Such a tuple exists, since it would be created at the latest during the final execution of $\mathsf{GVerify}$ by $\mathcal{A}'$. We call the j-th hash query the "crucial" hash query. $\mathcal{A}'$ outputs (j, σ).

We use the forking lemma as given in Section C with the algorithm $\mathcal{A}'$ and input generator $\mathcal{IG}$ outputting gpk. Provided $\mathcal{A}$ wins the non-frameability game with probability ϵ, makes the probability of $\mathcal{A}'$ finding (j, σ) with $j \geq 1$ become $acc = \epsilon$. Through the general forking lemma we get an algorithm $\mathcal{F}_{\mathcal{A}'}$, which succeeds with probability

$$frk \geq \frac{acc^2}{q_{\mathcal{H}}} - \frac{1}{q} = \frac{\epsilon^2}{q_{\mathcal{H}}} - \frac{1}{q}$$

to produce a tuple $(1, \sigma_0, \sigma_1)$. The number of queries of $\mathcal{A}$ has the upper bound $q_{\mathcal{H}}$.

Running the algorithm $\mathcal{F}_{\mathcal{A}'}$, the simulator $\mathcal{B}$ obtains two signatures $\sigma_1 = (d_1, e_1, f_1, (\mathsf{Cha}_1, \mathsf{Rsp}_1))$ and $\sigma_2 = (d_2, e_2, f_2, (\mathsf{Cha}_2, \mathsf{Rsp}_2))$. By a similar reasoning as in the traceability proof in Appendix B.2, we have that $(d_1, e_1, f_1) = (d_2, e_2, f_2)$, $\mathsf{Cha}_1 \neq \mathsf{Cha}_2$, and $\mathsf{Cha}_1 \neq \mathsf{Cha}_2 \bmod q$. The simulator $\mathcal{B}$ computes $\xi = (\mathsf{Rsp}_2 - \mathsf{Rsp}_1)/(\mathsf{Cha}_1 - \mathsf{Cha}_2) \bmod q$, which satisfies the equation $\hat{e}(f_1, \tilde{g})/\hat{e}(d_1, \tilde{x}) = \hat{e}(e_1, \tilde{x})^{\xi}$. Due to our construction of the signatures, we have that $\xi = \mu r_i$ holds, where r_i can be looked up in $\mathbf{rand}$. We therefore have that $g^{\xi/r_i} = g_1$ and $\tilde{g}^{\xi/r_i} = \tilde{g}_1$, so that ξ/r_i is a solution to the SDLP problem. The simulator obtains this value with probability at least frk.

C Forking Lemma

We recall here the general forking lemma due to Bellare and Neven [5]. In the following, think of x as a public key, q_{H} as the number of queries to a random oracle, and $h_1, \ldots, h_{q_{\mathrm{H}}}$ as the responses.

Lemma Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \ldots, h_{q_{\mathrm{H}}}$ returns a pair $(j, \sigma) \in \{0, \ldots, q_{\mathrm{H}}\} \times \{0,1\}^*$. Let $\mathcal{IG}$ be a randomized algorithm called the input generator. The *accepting probability* of $\mathcal{A}$, denoted acc, is defined as the probability that $j \geq 1$ in the experiment

$$x \leftarrow \mathcal{IG} \; ; \; h_1, \ldots, h_{q_{\mathrm{H}}} \leftarrow \{0,1\}^{\ell} \; ; \; (j, \sigma) \leftarrow \mathcal{A}(x, h_1, \ldots, h_{q_{\mathrm{H}}}) \; .$$

The *forking algorithm* $\mathcal{F}_{\mathcal{A}}$ associated to $\mathcal{A}$ is the randomized algorithm that on input x proceeds as follows:

Algorithm $\mathcal{F}_{\mathcal{A}}(x)$
Pick random coins R for $\mathcal{A}$
$h_1, \ldots, h_{q_{\mathrm{H}}} \leftarrow \{0,1\}^*$
$(j, \sigma) \leftarrow \mathcal{A}(x, h_1, \ldots, h_{q_{\mathrm{H}}}; R)$
If $j = 0$ then return $(0, \varepsilon)$
$h'_j, \ldots, h'_{q_{\mathrm{H}}} \leftarrow \{0,1\}^{\ell}$
$(j', \sigma') \leftarrow \mathcal{A}(x, h_1, \ldots, h_{j-1}, h'_j, \ldots, h'_{q_{\mathrm{H}}}; R)$
If ($j = j'$ and $h_j \neq h'_j$) then return $(1, \sigma, \sigma')$
Else return $(0, \varepsilon, \varepsilon)$.

Let

$$frk \;=\; \Pr\left[b = 1 \;:\; x \leftarrow \mathcal{IG}\;;\; (b, \sigma, \sigma') \leftarrow \mathcal{F}_{\mathcal{A}(x)}\right] \;.$$

Then

$$frk \;\geq\; \frac{acc^2}{q_{\mathrm{H}}} - \frac{1}{2^{\ell}} \;.$$

Anonymous Credentials on a Standard Java Card

Publication Data

Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 600–610, Chicago, IL, USA, November 2009. ACM Press.

Contributions

- Principal author.
- Architecture, Java Card analysis, implementation, measurements.

Copyright

Original version: `http://doi.acm.org/10.1145/1653662.1653734`

Anonymous Credentials on a Standard Java Card

P. Bichsel[1], J. Camenisch[1], T. Groß[1], and V. Shoup[2]

[1] IBM Research – Zurich, Switzerland.
{pbi,jca,tgr}@zurich.ibm.com
[2] New York University, USA.
shoup@cs.nyu.edu

Abstract. Secure identity tokens such as *Electronic Identity (eID)* cards are emerging everywhere. At the same time user-centric identity management gains acceptance. *Anonymous credential schemes* are the optimal realization of user-centricity. However, on inexpensive hardware platforms, typically used for eID cards, these schemes could not be made to meet the necessary requirements such as future-proof key lengths and transaction times on the order of 10 seconds. The reasons for this is the need for the hardware platform to be standardized and certified. Therefore an implementation is only possible as a Java Card applet. This results in severe restrictions: little memory (transient and persistent), an 8-bit CPU, and access to hardware acceleration for cryptographic operations only by defined interfaces such as RSA encryption operations.

Still, we present the first practical implementation of an anonymous credential system on a Java Card 2.2.1. We achieve transaction times that are orders of magnitudes faster than those of any prior attempt, while raising the bar in terms of key length and trust model. Our system is the first one to act completely autonomously on card and to maintain its properties in the face of an untrusted terminal. In addition, we provide a formal system specification and share our solution strategies and experiences gained and with the Java Card.

Key words: Anonymous credential systems, Java Card, privacy-enhancing systems, smart card.

1 Introduction

Electronic authentication tokens are spreading rapidly. Applications today already include ticketing, access to buildings, and road tolls. A number of countries have issued electronic ID (eID) cards or are about to do so. All these existing or emerging solutions have in common that the user is fully identifiable in the transactions involving the token. Indeed, many of them offer strong cryptographic

identification or qualified digital signatures. The resulting loss of privacy is subject to discussion as pointed out by Huysmans [17], but it is not a severe problem for e-government applications. However, a government-issued root of trust is very attractive for secondary use by (commercial) service providers. Here, privacy becomes a real issue. Indeed, in many commercial applications, unique identification is inappropriate, attribute-based authentication highly desired, and suitable privacy protection essential to make the services sustainable.

For instance, consider a teenager accessing an online chat room by eID. Here, the aim is to restrict access to teenagers only. It is crucial that no data other than the age range of the teenager is revealed to the chat provider. Indeed, if all the eID card's information is revealed and gets into the wrong hands, more damage is done than protection gained. Furthermore, consider a citizen using the eID throughout her entire lifetime and with various third parties. Without sufficient privacy protection, service providers could trace and profile the citizen across organizations. This would lead to an erosion of the citizens' trust and result in the non-sustainability of the entire system. We believe that sustainable secondary use is a is a make-or-break requirement for eID systems as well as for any identity token that supports authentication with third parties.

The ability to build comprehensive user profiles in the context of attribute-based authentication carries the need for strong privacy protection further than mere trust erosion would. It implies the need for full anonymity, which includes unlinkability. Examining identity tokens, and in particular eID cards, over a long time period, then the monotonous growth of identity information at service providers can only be overcome by full anonymity by default. This requirement entails further goals by implication: firstly, we need privacy-enhanced credential systems, namely, anonymous credential systems. Secondly, no (unnecessary) trusted third parties should be involved in transactions, i.e., the credential system must be autonomous. Optimally, the user shall only trust her own identity token and no other principal. Thirdly, if one considers linkability by timing, the credential system must be able to operate offline, based on long-term certificates.

Let us expand these three thoughts before we analyze the trust model and and hardware setting, in particular, typical smart cards as used to realize eID cards. Luckily, there exist privacy-enhancing technologies addressing our requirement of full anonymity, and allow for attribute-based access control. Anonymous credential systems [8, 20] allow an identity provider to issue an *anonymous credential* to a user. This credential contains attributes such as the user's address or date of birth but also her rights or roles. Using the credential, the user can prove to a third party that she possesses a credential containing a given attribute or role without revealing any other information. For example, in the child-protection example described earlier, the youngster could use a government-issued credential to prove that she fulfills the requirement on the age range. Thus, it seems that what is urgently needed is an implementation of anonymous credentials on tokens, such as smart cards.

The anonymous credential systems proposed by Brands [5] or Camenisch and Lysyanskaya [8] can be implemented on ordinary computers as described in [10] without difficulties. However, it seems they are not suited for implementation on smart cards or USB tokens. Bichsel [3] and Balasch [2] conclude that only systems using joint computation with the terminal can be implemented given the hardware restrictions for the eID scenario. This statement especially holds, if future proof key lengths of at least 1400 bits are considered. But even tremendously reduced systems did not meet the expected transaction times of production eID cards, which are defined to be in the order of 10 seconds.

Beyond the transaction times and key length, there are three more mundane requirements on eID cards imposed by governments and eID technology providers. Firstly, the smart card platform should be standardized, for governments and eID technology providers shy away from proprietary technology lock-ins.[3] Also, we envision the anonymous credential system to be deployed as a complement to existing eID systems and not to replace other authentication mechanisms. Even though one could achieve much more efficient solutions with a native card implementation, this would severely hamper the acceptance for the proposal. We therefore base our work the Java Card 2.2.1 standard [23].

Secondly, the eID card must be certified, for instance in a Common Criteria for Information Technology Security Evaluation [13]. Clearly, we need to aim at making the certification gap as small as possible and, therefore, use an off-the-shelf smart card with comprehensive certification.

Thirdly, the smart card platform must be well-established and cheap. We therefore restrict ourselves to smart cards that are 3–4 years old and in production in current eID systems. We also follow the standard operation procedures of these smart cards, e.g., to avoid write-operations to EEPROM whenever possible.

The main obstacle to implementing anonymous credential systems on such cards seems to be twofold. Firstly, we need to execute fast modular exponentiations, which requires the use of the card's cryptographic co-processor. However, the interfaces offered by the (off-the-shelf) cards' operating system do not give direct access to this, but only offer high-level functionality such as RSA encryption. Consequently, we will use the limited interfaces that standard Java Cards provide. Secondly, typical smart cards are rather limited in the amount of RAM that can be used for computations. This makes it, for instance, hard to store all intermediary results during an authentication transaction.

1.1 Related Work

There have even been several approaches to implement anonymous credential systems on smart cards. Bichsel [3] and Balasch [2] focus on providing the arithmetic functionality required by anonymous credential systems, i.e., fast modular arithmetic. Balasch implements the arithmetic using AVR microcontrollers,

[3] Of course, this requirement also transfers to the standardization of anonymous credential systems.

whereas Bichsel uses the JCOP platform. Danes [14] provides an analysis of different trust models and compares them with respect to security and privacy. He projects computation times using the hardware specification, implicitly assuming a custom operating system, and obtains an execution time of 6 seconds for his preferred protocol. This protocol still assumes trust in the terminal. We provide a comparison of the measurements of the three approaches with ours in Table 1. Note that the table focuses on the computation times of the core anonymous credential system and does not account for additional computations such as revocation equations.

	Danes [14]	**Bichsel** [3]	**Balasch** [2]
Date	2007	2007	2008
Bit Length	—	$72^{344}(72)$	$1024^{1752}(1024)$
Transaction Time	—	450s	133.5s
Trust Model	trust terminal	trust terminal	trust terminal
Implementation	none, prediction of transaction time	on Java Card JCOP v2.2/41	AVR 8-Bit RISC microcontroller

Table 1: Overview of previous approaches to establish anonymous credential systems on a smart card. We compare the implementations in terms of the transaction time, even though systems prior to pur proposal only execute a partial proof and use smaller key length. We denote the system parameters with base bit length ℓ_b, modulus bit length ℓ_n and maximal exponent bit length of ℓ_e by ${\ell_b}^{\ell_e}(\ell_n)$.

	This Paper		
Date	2009		
Bit Length	$1280^{735}(1280)$	$1536^{895}(1536)$	$1984^{1152}(1984)$
Transaction Time	7.4s	10.5s	16.5s
Trust Model	autonomous		
Implementation	on Java Card JCOP v2.2/41		

Table 2: Performance of our implementation of anonymous credentials on a smart card. The computation times can be compared with previously proposed systems listed in Table 2. We compare transaction times using different key lengths denote the system parameters with base bit length ℓ_b, modulus bit length ℓ_n and maximal exponent bit length of ℓ_e by ${\ell_b}^{\ell_e}(\ell_n)$.

Given that the authors use very different systems, we want to analyze the systems on the basis of single exponentiations. The difference to Bichsel [3]

is apparent and does not need further explanation. The implementation by Balasch [2] can be compared by extrapolation of his measurements. An exponentiation of a base/modulus bit length of 1984 with an exponent of 1024 bits accounts to roughly 270 seconds. We are able to compute such an exponentiation in 1.3 seconds.

1.2 Our Contributions

In this paper, we overcome the technical limitations to implement the Camenisch-Lysyanskaya (CL) anonymous credential system on a standard Java Card. We do this by exploiting the RSA encryption interface in a number of ways and by clever management of the available resources (especially RAM). In fact, our implementation can execute a proof of possession of a credential in a few seconds, which is fast enough for a multitude of eID use cases. Thus, we believe to have overcome the possibly final *technical* barrier for privacy-protecting electronic identity tokens.

Our contributions are twofold. Firstly, we discuss the challenges of actually implementing the CL credential system on a Java Card. In particular, we consider the severe platform restrictions, which entails concise analysis of the available interfaces as well as careful treatment of the hardware resources. In addition, we share our experiments, experiences, and strategies to overcome these limitations. Our solutions enabled us to outperform all prior anonymous credential system proposals on smart cards by several orders of magnitude. Moreover, our insights and tricks can be of merit for other implementations of advanced cryptographic primitives on Java Cards.

Secondly, we report the first practical implementation of an anonymous credential system on a standardized, off-the-shelf Java Card, a JCOP v2.2/41. We use a variant of the standardized Direct Anonymous Attestation (DAA) protocol [6], and demonstrate the feasibility of such a system for actual eID cards. In contrast to prior proposals, our smart card credential system is autonomous, that is, it forgoes any joint computation with the terminal. Our system not only guarantees the secrecy of the user's master key during the card's complete life cycle, but also protects user's privacy in face of an untrusted terminal. Our system goes far beyond a pure demonstration as it achieves production-quality parameters for eID cards. This includes strong key length of 1536 bits for the strong RSA modulus, transaction times on the order of seconds, and very modest hardware requirements (see Table 1). In fact, it could be applied to eID Java Cards currently being rolled out in various European countries.

1.3 Outline

The remainder of the paper is structured as follows. We begin with a discussion of the requirements in Section 2, where we start with requirements that are imposed by the eID scenario and continue with functional requirements that

rise in the context of secondary use of eID cards. Section 3 elaborates on the underlying cryptographic system, design decisions and concludes with the protocol specification. In Section 4, we illustrate the main obstacles we encountered when realizing the system we specified, the solutions we developed, the architecture we built, and the measurements we performed. We discuss the results of our implementation with respect to the requirements in Section 5 and conclude with Section 6, where we provide an outlook on future development of anonymous credential systems on smart cards.

2 Requirements

We base our requirements discussion on the scenario of eID cards for three reasons. Firstly, eID technology is likely to pervade entire societies and to affect the life of many citizens. Secondly, its actual hardware platform is particularly challenging for implementing an anonymous credential system. Thirdly, it allows us to intuitively motivate requirements that abstractly hold for any application involving personal tokens with severe resource restrictions.

2.1 Application Requirements

Let us begin with the requirements dictated when using an eID card for applications having non-government organisations as service providers.

Sustainable Secondary Use. The users must be able to use their eID card over their entire lifetime without privacy or trust degradation. A continuous strong privacy protection for all transactions is crucial. This is a key requirement that we are going to meet by using an anonymous credential system.

Autonomous Trust Root. A wide range of trust scenarios must be supported without drawbacks on security or privacy. Particularly, the card must act securely in face of an untrusted or malicious terminal[4]. Therefore, the anonymous credential system must protect the citizen's security and privacy autonomously and cannot (easily) delegate computations to the terminal.

The privacy discussion gains in complexity with the introduction of variable attribute policies, as the terminal may attempt to send the eID card multiple policy requests—without the citizen's knowledge or consent—to infer a profile of the citizen. As the eID card is in principle stateless, it is at the mercy of the terminal. The terminal can easily reset the card and send another policy request. Naive solutions to store the card's state or create an audit log of the terminal's requests are not easily feasible because of the cards limitations in write/erase cycles on persistent memory. Proposals that certify card readers as well as applicable

[4]From a user perspective, sharing data with the own device has different implication compared to sharing with a third-party terminal, e.g., at a bar, or an Internet cafe.

policies are used to confine a potential exposure, be it in terms of obtainable attribute set or of potentially malicious readers.[5]

Long-term Certificates. An eID card must forgo short-term updates, particularly of the keys and certificates, be it because some countries support offline[6] applications (such as vending machines), or because some countries ban card updates outside of a trusted environment. In privacy terms, this allows to prevent a linking by timing.

Performance. An anonymous credential system for eID cards faces stiff performance requirements, notably, the need to complete transactions in mere seconds.

Future Proof Key Length. Currently, lengths of an SRSA modulus size greater than or equal to 1400 bits are considered future proof.

2.2 Functional Requirements

Clearly, unique identification, qualified signatures, and disclosure of the citizen's full address are important functional requirements for eID cards; however, we focus on functional requirements with stronger privacy properties.

Proof of Possession. The card must be able to issue a proof of possession of a credential. Thus, proving the value of an attribute without leaking any information about the attribute value.

Age proof. Nowadays, eID cards are often used as basis for a proof of age, mostly in the area of youth protection. Contrary to the common perception of an age proof as a means to show adulthood and to obtain restricted goods (medias, alcohol, cigarettes), age proofs are also important to establish protection zones for youngsters on the Internet.

Finite-set Attributes. Also, eID cards contain a variety of binary or finite-set attributes that are particularly privacy-sensitive [7]. Consider, for instance, attributes of health and special status: visually or hearing impaired, social benefit recipient, unemployed, or elderly. Undoubtedly, these attributes need to be disclosed only selectively, or even only issue a proof certifying that the citizen is entitled to receive social welfare by holding one out of many attributes.

Revocation. Revocation is of central importance for eID systems. The card needs to be revoked when the owner declares her eID card lost or stolen. As the traditional approach of revocation lists implies privacy hazards for honest citizens, we need to explore privacy-preserving revocation mechanisms.

[5] Our system can easily realize a check of the terminal's attribute policy and restrict the disclosable attributes for uncertified terminals. However, these proposals do not constitute a real solution of the problem at hand, and further research is required in this area.

[6] Offline, here, refers to the terminal being able to serve the request of the card without an online connection to the authorities or to an identity provider.

2.3 Hardware Requirements

Let us summarize our hardware challenge: Our goal is to establish an autonomous credential system on a smart card with the following properties: (i) a standardized Java Card with comprehensive security certification, (ii) used by existing eID systems in production, and (iii) with restricted write/erase-cycles. We use the Java Card 2.2.1 standard [23] interface, which prevents direct access to the cryptographic co-processor, fast multiplication, and exponentiation primitives. It only offers the use of well-defined primitives such as RSA encryption. In addition, transient memory is severely restricted (750 bytes heap, 200 bytes stack), which makes the implementation of multi-base exponentiation and many pre-computation techniques virtually impossible.

These severe limitations explain why prior proposals [2, 3] could only achieve transaction times on the order of minutes, despite the fact that they delegated most computations to the terminal.

3 Protocol Design

Let us consider the protocol design for a standard Java Card in stages. Firstly, we review the cryptographic variants of anonymous credential systems. Secondly, we discuss the options for hardware trust. Thirdly, we present out design decisions that follow these arguments.

3.1 Cryptographic Alternatives

Anonymous credential systems were introduced by Chaum in [11, 12] and subsequently improved, in particular, by Brands [5] as well as Camenisch and Lysyanskaya [8, 9]. Relations based on blind signatures such as those by Brands have a severe drawback when it comes to implementations on a smart card: Proving possession of a credential in an unlinkable, i.e., privacy-maintaining, way requires the issuance of a new credential, which would exhaust the EEPROM write cycles quickly. Identity mixer, developed by Camenisch et al. [19], does not suffer from this limitation, i.e., one credential can be used repeatedly to prove its possession without these proofs becoming linkable.

Therefore, we have chosen the Camenisch-Lysyanskaya (CL) [8] signature scheme as basis for our lightweight credential system on Java Card. Let us first consider the variants of Camenisch-Lysyanskaya itself: The most common one is based on the Strong RSA assumption and specified in the Identity Mixer protocol suite [19]. Subsequently, Camenisch and Lysyanskaya proposed alternatives based on bilinear maps that rely on the LRSW assumption [8], and one that build on the Boneh-Boyen-Shacham group signature scheme [4]. The latter was improved upon by Au, Susilo, and Mu [1]. The bilinear map variants of the CL signature scheme can operate in smaller prime-order groups, whereas the SRSA variant requires a large composite modulus. Thus, the bilinear map variant is advantageous in

general, particularly as the SRSA variant has the client operate with unknown group order. Nevertheless, we dismiss the bilinear maps based variant as the smart cards considered do not offer suitable algebraic support for the required elliptic curves.

3.2 Hardware Resilience

We need to consider an important balance question for eID cards: To what extent can we trust the hardware's resilience and how much do we need to rely on cryptographic protection? We note that typical eID cards are tamper-resistant and equipped to protect their private keys for identification and qualified signatures. Thus, we can assume that it is costly to break/clone a single eID card, and that an attack of the tamper-resistance of one card does not easily transfer to attacks of other cards. Thus, the damage is local as otherwise the system of eID cards would be broken as a whole.

As a means of mitigating the damage of broken or stolen cards, an eID system needs provisions for revocation. Typically the issuing authority would be in charge of revoking cards once a local breach has been detected. The eID scenario holds more potential impact associated with breaking the tamper-resistance of identification and qualified signatures than the attribute-based authentication. Therefore, it is, in principle, sufficient to have the same protection standards as for the other pillars of eID functions and, by extension, good enough to trust the hardware resilience for our use cases.

Finally, we conclude that the resilience of eID cards is an important protection feature that mitigates potential breaches. Under the condition of a sufficient revocation system, it is possible to choose more efficient cryptographic mechanisms while maintaining the same level of protection.

3.3 Design Decisions

Implementing the full-fledged CL anonymous credential system would not be feasible on current cards. That is, features such as an age-proof (i.e., proving that the date of birth contained in the credential issued has a distance of at most n years from the current date) or encoding all the more than 20 typical fields of a standard identity card and allowing selective disclosure for each of them would result in a computation time on the order of 70–100 seconds. As this would not be suited for practice, we rely on the tamper resistance of the hardware for such attribute-related proofs.

Thus, similarly to the model the Trusted Computing Group has taken for their TPM chips with the Direct Anonymous Attestation (DAA) protocol [6], we have a two-stage approach. We use anonymous credentials to have the smart card prove that it is a valid (and intact) card and therefore can be trusted to make statements about its bearer. These statements, e.g., an age proof, are then made by the card itself. Consequently, the correctness of these statements is not

enforced cryptographically but by the tamper resistance of the card. Thus, we will have to protect ourselves against the case when a card is broken open and the cryptographic credentials are extracted. In this case, the extracted credential can be used to back any attribute-related statement. However, breaking a card is costly and will mostly be done for economical reasons. Thus, employing techniques that protect from massive sharing might be the most appropriate action.

Our proposed solution is therefore as follows. We issue a Camenisch-Lysyanskaya credential [8] with a secret key m_0 on an eID card and store all attribute information about the citizen in the card independently of the credential.[7] When the citizen wants to use the cards for some privacy-protecting authentication, we let the eID card compute a valid attribute statement (based on the attribute information stored on the card) and sign it with the Fiat-Shamir heuristic [15] during a proof of possession of the issued credential. On top of that, the card provides a discrete log commitment, i.e., $C = g_r^{m_0}$ on the secret key m_0 with a random base g_r during each transaction (where it is ensured by the proof of possession that this is chosen correctly).[8] This commitment is a pseudo-random value with computational hiding properties. However, it allows for the detection of revoked cards as authorities can check C against $g_r^{\hat{m}_0}$ for each $\hat{m}_0$ retrieved from the revocation list and, if there is a match, decline the transaction (or take legal action).

3.4 Protocol Specification

The system setup starts with the initialization of the smart card, which can only be executed once. It continues with the issuance of at least one credential. From that point on, a proof of possession can be executed. Additional credentials can be issued and bound to the card at any point later on.

Smart Card Setup. The master secret m_0 can only be set once for each card with $m_0 \in \{0,1\}^{\ell_m}$. It will be used to bind all certificates issued to a card together and to the card. It is generated by the card and released only computationally hidden.

After the setup of the smart card, a credential is issued to it. During this process the issuer public key comprising a modulus $n = pq$, where p, q are safe

[7] Our implementation is capable of including more attributes in the credential as well as handling them in zero-knowledge proofs of knowledge and selective disclosure. Each additional attribute exponent comes at a cost of 1684ms transaction time at a modulus bit length of 1536 bits. This accounts for the modular exponentiation, required multiplications, additions and PRNG calls. Of this, 1016ms are pre-computation, 668ms are policy-dependent. We have tested this functionality, yet do not propose it as primary solution.

[8] This discrete log commitment needs to be computed separately from the performance measurements we provide in Table 4. The card needs to generate a random base and compute the commitment as well as prove its representation in zero-knowledge. This costs several calls to the PRNG to generate 1536 bits for a pseudo-random base and two ModExp. The response for the zero-knowledge proof of m_0 can be reused. With a 1536-bit modulus, this makes an additional transaction time of 1474ms, which can be fully handled at pre-computation time.

primes that fulfill $p = 2p' + 1$ and $q = 2q' + 1$, and p', q' are primes, is needed. This key also contains bases $Z, S_2, R_0, \ldots, R_i \in_R \langle S_1 \rangle$ where S_1 is an arbitrarily chosen quadratic residue modulo n and $\langle S_1 \rangle$ denotes the group generated by S_1. A more detailed description of the issuer key generation can be found in [19]. We chose the relevant bit lengths as follows: $\ell_n = 1536$, $\ell_m = 256$, $\ell_e = 592$, $\ell'_e = 120$, $\ell_v = 768$, $\ell_\varphi = 80$ and $\ell_{\mathcal{H}} = 160$. In general, ℓ_k denotes the bit length of parameter k. The bit lengths ℓ'_e, ℓ_φ and $\ell_{\mathcal{H}}$ define the length e', the bit length used to achieve statistical zero knowledge, and the bit length of the hash values, respectively. Note that the parameter ℓ_v is much shorter than suggested in the Identity Mixer specification [19] because two blinding bases are used.

After having run the issuance protocol successfully, the smart card holds a valid CL signature (A, v, e). We want to discuss the proof protocol. The issuance protocol entails similar challenges and benefits from the same solution strategies.

Proof of Possession Protocol. Proving possession of a certificate follows the lines of argumentation of the Identity Mixer protocol. As the card cannot handle exponents that are larger than the modulus, we split the long exponents into two shorter ones at the cost of computing an extra exponentiation. More precisely, instead of computing S^v we compute $S_1^{v_1} S_2^{v_2}$ with $S_1 = S$, $S_2 = S^{2^\ell}$ and $v = v_1 + v_2 2^\ell$ for a suitable ℓ. The verifier starts the protocol by sending a nonce $n_1 \in_R \{0,1\}^{\ell_{\mathcal{H}}}$ to the prover, which guarantees the freshness of the proof. The prover continues by first choosing $v_1^*, v_2^*, r_{g_R} \in_R \{0,1\}^{\ell_v + \ell_\varphi}$ and subsequently computing the following values.

$$A' := A S_1^{v_1 *} S_2^{v_2 *} \pmod n \qquad g_R := S_1^{r_{g_R}} \pmod n$$
$$\bar{v}_i := v_i - v_i^* e,\ i \in \{1,2\} \qquad C := g_R^{m_0} \pmod n$$
$$e' := e - 2^{\ell_e - 1}$$

The card sends $A', (g_R, C)$ to the terminal, which forwards it to the verifier. In addition, the card calculates $\hat{e}, \hat{m}_0, \hat{v}_1, \hat{v}_2$ and the hash c, and sends those values to the terminal. For the calculation mentioned, the card chooses $\tilde{e} \in_R \pm\{0,1\}^{\ell'_e + \ell_{\mathcal{H}} + \ell_\varphi}$, $\tilde{m}_0 \in_R \pm\{0,1\}^{\ell_m + \ell_{\mathcal{H}} + \ell_\varphi + 1}$ and $\tilde{\bar{v}}_1, \tilde{\bar{v}}_2 \in_R \pm\{0,1\}^{\ell_v + \ell_{\mathcal{H}} + \ell_\varphi}$ at random. To continue with the calculation of the proof, the following values are computed:

$$\tilde{T} := A'^{\tilde{e}} R_0^{\tilde{m}_0} S_1^{\tilde{\bar{v}}_1} S_2^{\tilde{\bar{v}}_2} \pmod n \qquad \hat{e} := \tilde{e} + ce'$$
$$\tilde{C} := g_R^{\tilde{m}_0} \pmod n \qquad \hat{m}_0 := \tilde{m}_0 + cm_0$$
$$c := \mathcal{H}\left(sysparam, \tilde{T}, \tilde{C}, n_1\right) \qquad \hat{v}_i := \tilde{\bar{v}}_i + c\bar{v}_i,\ i \in \{1,2\}$$

The verifier can check that the smart card possesses a valid credential by computing the challenge $\hat{c}$ and comparing it with the submitted challenge c.

$$\hat{T} := \left(\frac{Z}{A'^{2^{\ell_e-1}}}\right)^{-c} A'^{\hat{e}} R_0^{\hat{m}_0} S_1^{\hat{v}_1} S_2^{\hat{v}_2} \pmod{n}$$
$$\hat{C} := C^{-c} g_R^{\hat{m}_0} \pmod{n}$$
$$\hat{c} := \mathcal{H}\left(sysparam, \hat{T}, \hat{C}, n_1\right)$$

Subsequently the lengths of $\hat{m}_0$ and $\hat{e}$ have to be verified with $\hat{m}_0 \in \{0,1\}^{\ell_m+\ell_\varphi+\ell_\mathcal{H}+1}$ and $\hat{e} \in \{0,1\}^{\ell'_e+\ell_\varphi+\ell_\mathcal{H}+1}$. It is straightforward to verify that $\tilde{T}$ equals $\hat{T}$.

Finally, checking whether the certificate has been revoked is done as follows. Assume that $(m_{(0,1)}, \ldots, m_{(0,f)})$ for some f is the list of revoked secret keys (i.e., the list of the secret keys that have been extracted from tokens). For each $m_{(0,j)}$ check that $C \neq g_R^{m_{(0,j)}} \pmod{n}$.

This proof protocol does not disclose any attributes and thus implements the DAA case. The extension of adding either disclosed or hidden attributes is straightforward [19].

4 Realization on a Smart Card

Given the cryptographic design decisions and the formal system specification, we now elaborate on the realization on an actual off-the-shelf Java Card in four steps: firstly, an analysis of the JCOP environment, secondly, strategies that can partially overcome the limitations, thirdly, integration of these aspects in a sketch of our high-level system design, and, finally, a report of the performance achieved.

Our system requires modular multi-base exponentiation, multiplication, and addition, all with a large composite modulus and without being privy of the group order. Furthermore, we need random numbers, digests for Fiat-Shamir, and a cache for intermediary results. We analyze the obstacles presented by the card, and derive optimizations methods. We achieve much by tunneling computations to the card's hardware accelerator and reducing other operations to the accelerated ones. Unfortunately, this hardware accelerator is well encapsulated behind the Java Card's high-level crypto interface, so that we resort to disguising credential system computations as RSA encryption operations.

In the following, we discuss the limitations of a Java Card, be it in terms of interfaces or be it in implementation environment (e.g., available RAM). We then show how our lightweight credential system can nevertheless be implemented, highlight key architecture concepts, and conclude with a discussion of the performance of our implementation. This last part shows that privacy-protection tokens are practical today.

4.1 JCOP Environment

The Java Card 2.2.1 standard [23] offers a well-defined set of interfaces to implement custom applications. We used a standard-compliant JCOP smart card [18], which imposes further limitations when it comes to low-level operations. We discuss the interfaces that are most relevant for our implementation. We start with basic restrictions such as RAM and 8-bit arithmetic, and continue with cryptographic primitives.

RAM Restrictions. On top of the restricted access to its crypto acceleration, a smart card has scarce transient memory. Our JCOP v2.2/41 card is equipped with 2304 bytes of RAM. This transient memory is distributed among the Java™stack, APDU buffer (used for communication between card and environment), atomic transaction buffer, and Java heap. For the calculations we can only make use of the transient heap, which is 750 bytes.

Evaluation 1. As we aim at reasonable modulus sizes of at least 1400 bit, each group element already requires at least 175 bytes of transient memory. Moreover, to compute the zero-knowledge proofs as specified in Section 3.4, we need to juggle multiple group elements in RAM at the same time. Therefore, transient memory is a highly limiting factor.

8-bit Arithmetic. The JCOP v2.2/41 Java Card comes with an 8-bit processor/ALU. All arithmetic operations such as addition, subtraction, and multiplication are delegated to it. Whereas the built-in arithmetic operates on byte and short values, we require operations of arbitrary-length integers. This is either supported by BigInteger libraries in newer smart cards or custom-implemented in the application layer. In any case, it is very costly. Our particular hardware contains a FAME-X (Fast Accelerator for Modular Exponentiation - Extended) crypto co-processor that features support for modular exponentiations. It is not directly accessible from the application layer of Java Cards.

Evaluation 2. Any attempt to have the exponentiations or multiplications computed by the 8-bit ALU is bound to fail, and will result in transaction times as highlighted by Bichsel [3] for Java Cards and Balasch [2] for AVR microcontrollers. Our best result for a pure application layer implementation of a 248-byte addition was 76ms. Projecting resulting exponentiation times indicates that we need to do better than that.

Random Number Generation. The JCOP v2.2/41 smart card offers true random number generation (TRNG) and pseudo random number generation (PRNG). Note that the PRNG expects a strong random seed and infuses further randomness sources, i.e., a standard-compliant PRNG does not produce the same outputs deterministically if seeded with the same number.

Evaluation 3. The proofs specified in Section 3.4 require the generation and reuse of multiple random exponents. The severe RAM limitations deny us the option to

store the randomness and the non-deterministic behavior of the PRNG denies its re-computation using the provided functionality.

SHA-1 Interface. The SHA-1 interface allows the hashing of messages of up to $2^{64} - 1$ bit length to a 160-bit string. SHA-1 is implemented in software on the JCOP v.2.2/41 that we use, and therefore, relatively slow. In addition, digest updates have to respect the block size of 64 bytes, which makes certain key lengths, i.e., 1984 bits, less favourable as we would need to hash 2048 bits.

Evaluation 4. The SHA-1 primitive is only a second-rate candidate to generate and recompute pseudo-randomness. We shy away from its slow software implementation and the transient memory impact.

DES/3DES Interface. The symmetric encryption interface offers a variety of modes, be it in terms DES, 2 key 3DES or 3 key 3DES, in terms of cipher block chaining mode (CFB) or electronic codebook mode (ECB), or in terms of the padding scheme.

Evaluation 5. For us, it is of particular importance that the JCOP v2.2/41 card offers hardware acceleration for 3DES operations and that there exist efficient and secure pseudo-random number generators based on 3DES.

DSA Interface. The DSA signing primitive uses various exponentiations that might be leveraged for our purposes. In particular, it executes a multi-base exponentiation with configurable bases.

Evaluation 6. The DSA key interface allows us to specify the public key and private key, but not the value of the exponents. We perceive the involvement of the hash function as obstacle to using the DSA interface for our intended acceleration of arithmetic operations.

RSA Interface. The Java Card 2.2.1 standard [23] offers RSA [22] encryption and decryption, either in normal mode or with Chinese Remainder Theorem (CRT) support for private key operations with known factorization. The RSA public key consists of the modulus n and the public exponent e, whereas the private key contains the secret exponent $d = e^{-1} \pmod{(p-1)(q-1)}$ and modulus factorization $n = p \cdot q$. The parameters p and q are chosen as random prime numbers that are of similar length and not equal. A message m is encrypted to $c = m^e \pmod{n}$ using the public key. The decryption uses the private key and retrieves $m' = c^d \pmod{n}$.

For the Java Card 2.2.1 standard, the public exponent e is usually quite small (4 bytes) and often fixed to common exponents such as 3 or Fermat-4. Whereas the JCOP environment allows us to set exponents and moduli of the RSA keys in a wider value range, it still limits the bit length of exponent (ℓ_e) and base (ℓ_b) to at most equal the bit length of the modulus ℓ_n.

Moreover, the Java Card 2.2.1 standard only allows computations on persistent keys (EEPROM), as normal RSA encryption operates on long-term keys.

Evaluation 7. In principle, the RSA primitive sounds like a good candidate to tunnel computations for the credential system to the hardware acceleration. We face three limitations: firstly, the constraint to small standard exponents for encryption may foil our endeavor altogether. Secondly, the limitation to modulus-size exponents conflicts with the blinding of the credential system: it must be larger than the modulus to stay provably secure. Thirdly, the restriction to persistent keys bars us from exploiting the interface directly: we need to update the keys frequently to obtain exponentiations with random values and would therefore cause many write cycles—slow death—to the EEPROM.

Summary. We have seen that, firstly, a Java Card—and in particular the JCOP v2.2/41 card—imposes severe limitations in terms of transient memory and 8-bit arithmetic on credential system operations, such that a delegation to hardware acceleration is unavoidable. Secondly, the exposed cryptographic interfaces are well encapsulated, either completely unusable for our endeavor or posing further technical obstacles. Thirdly, the RSA interface is promising, but requires a new implementation of transient keys with long public exponents to serve our purpose. Also, it would conflict with compliance with the Java Card 2.2.1 standard.

4.2 Our Solution Strategies

In general, one can solve most of the restrictions indicated using the computation time versus storage trade-off. Balasch shows in [2] some results in this direction. However, we could not allow ourselves this luxury: given the very tight bounds on all relevant metrics with our Java Card, we had to look for other solution strategies.

Multi-base Exponentiations. Undoubtedly, multi-base exponentiations are the most important operation in our protocol. The combination of not having an interface to exploit hardware-accelerated multi-base exponentiation, and computing multi-base exponentiations in the application layer consuming too much transient memory, we dismissed this option altogether. Also, it falls back on a custom implementation on the 8-bit ALU that is too slow. We resort to hardware-accelerated modular exponentiations.

Modular Exponentiation. Modular exponentiation that is implemented on the application layer exhibits a devastating performance, which even holds when using advanced methods such as Montgomery reduction.

Idea 1. We delegate modular exponentiations to the RSA encryption and overcome interface limitations (Section 4.1) as follows:

- By creating a new transient RSA key design that supports public exponents in modulus length and a rapid change of exponents in transient memory. This is made possible by the use of special library and violates the Java Card 2.2.1 standard. Note, that the Java Card 3.0 standard does allow RSA keys to be stored in transient memory.

- By modifying the credential system to execute the blinding over two bases S_1 and S_2 instead of a single base S, thus maintaining provable security with smaller exponent sizes.

Let us consider this in detail: firstly, RSA keys normally reside in EEPROM, or even worse, in a protected EEPROM section. Therefore, executing many exponentiations with changing RSA keys, will deplete the write cycles of this particular EEPROM section very quickly. In addition, writing to EEPROM takes much longer than writing to RAM[9]. We overcome this limitation by creating a new RSA key structure that resides in transient memory. Although Java Card 2.2.1 does not support RSA keys in transient memory, JCOP actually does, and we exploit this in our implementation. Note, however, that the newer standard Java Card 3.0 does support RSA keys in transient memory, and so, looking forward, our solution will be standard-compliant.

Secondly, we overcome the exponent and base length limitations, which prevented us from carrying out the Identity Mixer computations as defined in [19]. Specifically the exponent for blinding the certificate needs to be larger than the modulus. The solution to this problem is the use of two bases with two independently chosen exponents which results in the equation given in Section 3.4.

Modular Multiplication. Modular multiplication is the second most important primitive when it comes to implementing the Identity Mixer anonymous credential system. It is also too heavy-weight for the application layer. Luckily, we succeeded in building an extremely efficient modular squaring primitive that overcomes this issue.[10]

Idea 2. We reduce multiplications to highly efficient squaring operations on the hardware acceleration by employing a *binomial formula*. In particular, we compute the modular multiplication of a and b modulo n by computing $((a+b)^2-a^2-b^2)/2 \pmod n = ab \pmod n$.

Because of the small exponent, the computation is very efficient. The subtraction is implemented naive, which makes it the predominant factor when it comes to computation time. The final division translates to a simple shift operation. Using the optimizations outlined, we can reduce the computation complexity in the application layer from $O(\ell_n^2)$ to $O(\ell_n)$, where ℓ_n is the length of the modulus.

Addition. Given our optimizations of exponentiation and multiplication, the addition and subtraction become predominant when it comes to performance. As production cards can be easily patched to expose a fast byte-array addition primitive, we base our smart card implementation on an application-layer arbitrary position integer addition. However, the following optimization can lead to a considerable protocol speed-up even with a standard card.

[9] Writing a page (1-64 bytes) to EEPROM typically takes 1.6ms according to [21].

[10] A modular squaring of a 1984-bit number with the hardware acceleration takes 9ms.

Idea 3. We could delegate the addition to the hardware acceleration by tunneling it through the RSA-CRT decryption operation. By carefully setting the base and exponent arguments the CRT algorithm produces an addition/subtraction in the decrypted message that can be extracted by inexpensive shifts.

Randomness. The Java Card offers a true random number generator. However, we cannot store the randomness for the proofs because of the severe memory limitations (see Section 4.1). We therefore need to regenerate pseudo-random values on demand. As the Java Card 2.2.1 standard specifies that the pseudo-random generation with the same seed will result in the same random number, we need an alternative mechanism.

Idea 4. We create our own pseudo-random number generator that allows us to regenerate randomness identified by variable names. We generate the seed with the true randomness generation of the JCOP card and use the formal state machine of Section 4.3 to enforce that a each proof is executed with a fresh random seed.

In the current implementation, we use the SHA-1 hash function to generate pseudo-randomness, however, this leaves much room for optimization: using the dedicated 3DES co-processor as PRNG would enable an additional performance gain. Considering the computation times of 3DES, which are specified as $< 35\mu$s [18], and comparing to the measurements in [2] (3ms per SHA-1 Op) as well as our experiments (22ms per 100-byte PRNG data), it is safe to estimate the benefit of this measure to roughly half the computation time of the pseudo randomness.

Transient Memory. We mitigate the scarce resources problem of transient memory by partially using memory dedicated to a fixed component. In our example, we use the card's communication buffer. With a length of 255 bytes, it has a reasonable size to be exploited. This approach carries the major risk that the buffer might be read or changed by other applets. Thus, we need to make sure that no sensitive data resides in this memory.

Idea 5. We use the communication (APDU) buffer of the smart card as additional transient memory. For security reasons we enforce that any data written to buffer is non-sensitive or already cryptographically blinded. This means in particular that a proof's randomness, the user's master key, and the attribute values are never written to the APDU buffer.

Summary. We created a toolbox for efficient computation of various algorithmic components of credential systems. It helped us to make the most of our situation, in view of its high expectations (future-proof key length and short transaction times) and severe limitations (RAM, 8-arithmetic, limited crypto interfaces). In particular, it enabled us to create the credential system on card as specified in Section 3.4 with the architecture and performance as described below.

4.3 Architecture of the Full System

Whereas we dedicated the previous sections to overcoming the low-level obstacles of the JCOP environment, we now take a step back to present the high-level architecture of the overall system. After all, realizing a full-fledged anonymous credential system on a Java Card is not just algorithms and tricks to achieve fast exponentiations, but requires serious consideration at the system level.

Our main requirements on the architecture are twofold. Firstly, it must strongly economize the Java Card's resources, and in particular, use transient memory optimally and in a tightly controlled manner. Secondly, it must feature strong security and robustness properties, i.e., justify its use in high-trust areas such as eID. We briefly discuss these two requirements by mentioning the core points of our architecture and complementing them with a design overview.

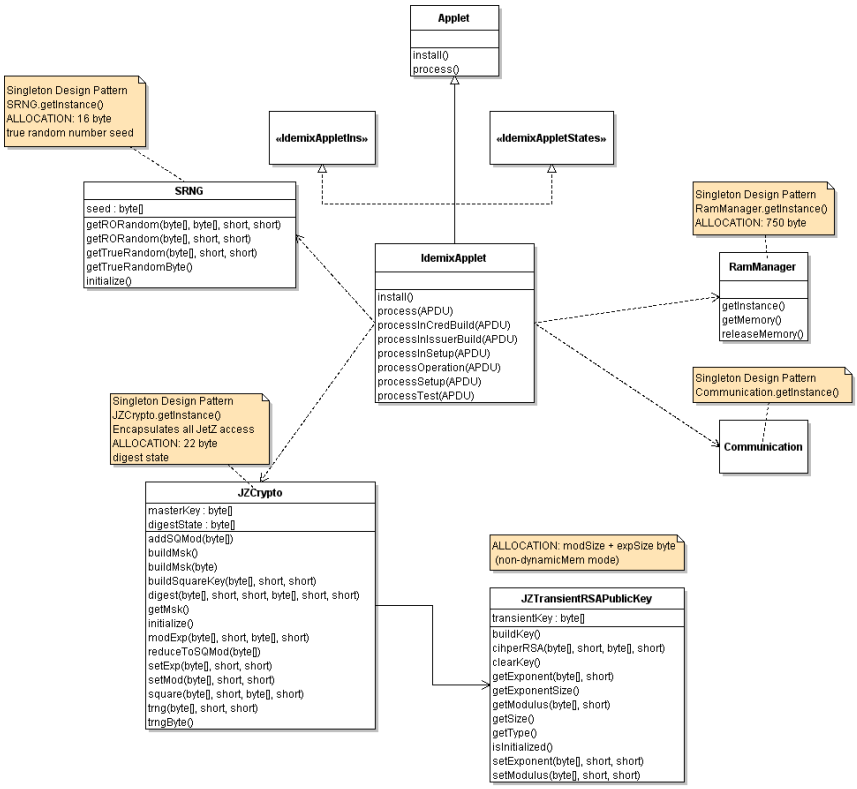

Figure 1: Overview of the class design of the credential system for Java Card.

Let us start with the economy aspects, which we complement with the class design overview of Figure 1. Transient memory clearly is the sparsest resource of the card, particularly because we juggle multiple large byte arrays with group

elements. We therefore first established an explicit RamManager that owns most of the applet's memory. It governs byte arrays for group elements as well as exponents and organizes the request and release of this memory. Secondly, we created most classes either in the *singleton* design pattern [16] or as *static*, such that there exists either only one instance with state or that the class does not have dynamic state at all. This design is not only for economy but also includes security features in terms of information flow/non-interference: the RamManager, for instance, guarantees that byte arrays are zeroed before reuse. Also, security-critical memory, such as digest state and random seed for PRNG, is completely separated from other computations and well encapsulated in the corresponding classes.

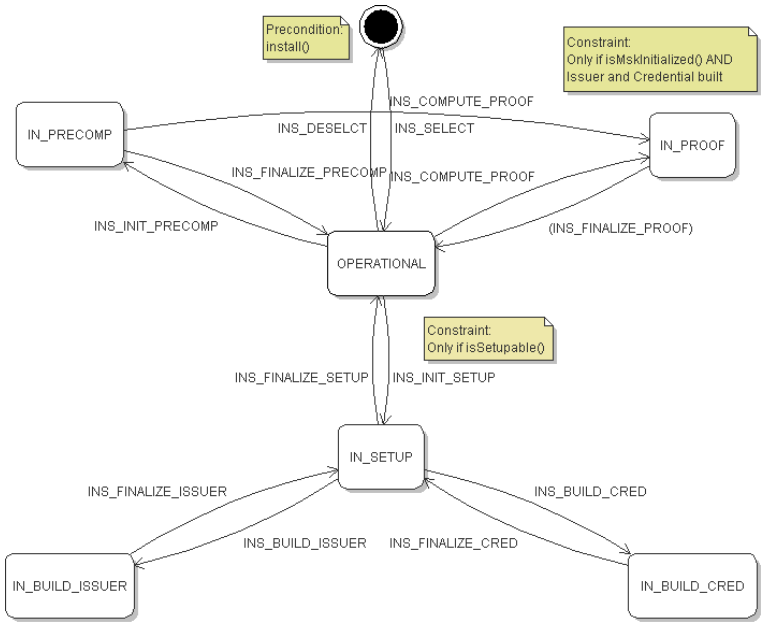

Figure 2: State machine of the anonymous credential system applet.

The security properties of the applet go beyond information flow control and, in particular, ensure consistency of the card's state. This can be on an atomic transaction level or on the system-state level. We solve the first part mostly through *Factory* design patterns [16][11] and prudence for all write operations to the card's EEPROM, which we realize with the Java Card's atomic transaction

[11] Factories, such as our CredentialFactory, are the focus and control point for class instantiation and access. For instance, Credential instances cannot be constructed directly, but need to be created by the corresponding factory in a well-defined robust process.

facility[12] and consistency checks before committing transactions. We solve the latter part with a formal state machine, depicted in Figure 2. It establishes tight control on setup and operational states, acceptable inputs, and potential transitions. Even though we thoroughly tested the applet with white and black box tests, we separated out all test functionality, which are interfaced by the abstract method processTest() and not compiled into the production version (similar to the *Visitor* design pattern [16]).

4.4 Performance

We performed measurements executed on a JCOP v2.2/41 smart card. We first measured the performance of the arithmetic operations. Especially, the modular exponentiation is of interest to us. Running 500 consecutive executions of an exponentiation using a base and a modulus with bit length of 1984 bits and an exponent of length 1024 bit, we measured a computation time of 1.3 seconds for each exponentiation. Cutting the exponent in half reduces the computation time by a factor of 2. Furthermore, the computation time of squaring a 1984-bit base using a modulus of the same length results in a computation time of approximately 15ms.

Our main interest lies in the computation times of the protocol proving holdership of a credential as described in Section 3.4. Note that the exponentiation for the revocation as specified is not included in the measurements. The computations of the credential issuance are less important as there are only a few credentials issued to a card, but possibly a large number of proofs of possession. Also, computations of the credential recipient and an entity proving possession of a credential are very similar, and timings can be well approximated.

We analyzed the performance using different key lengths starting with a 1280-bit modulus up to a modulus length of 1984 bits. We chose the upper limit on the length of the bases to be equal to the length of the modulus[13]. To get a better overview, we split the computation time in a pre-computation and a policy-dependent part. The pre-computation consists of the computation of A' and the computation of $R_0^{\tilde{m}_0} S_1^{\tilde{v}_1} S_2^{\tilde{v}_2}$ as specified in Section 3.4. The timings presented are not only computations, but include communication times, i.e., they represent a real interaction with the card as it proves holdership of a credential. Communication time occurs while sending a number to the card or receiving the result from the card. We outline the result of these measurements in Table 3.

To get an idea which operations need a significant amount of the overall computation time we analyzed the time each individual operation consumes. The result of this experiment is listed in Table 4. We ran 1000 executions of the

[12]The Java Card 2.2.1 standard [23] can make transactions of multiple computations and write operations to persistent memory atomic. Either the entire transaction finishes successfully or the card is reset to the state before the transaction.

[13]In a setting with a large exponent (> 200 bits), the length of the base has a negligible impact on the computation time.

Modulus length	1280 bit	1536 bit	1984 bit
Pre-computation	5203ms	7828ms	13250ms
compute A'	2125ms	2906ms	5000ms
compute T_1	3078ms	4922ms	8250ms
Policy dependent	2234ms	2625ms	3298ms
compute s.	562ms	656ms	828ms
Total	7437ms	10453ms	16548ms

Table 3: Computation times of our implementation comparing different bit lengths of the modulus.

protocol's arithmetic operations to acquire the results. The upper bound on the bit length of the base was 1536 and the upper bound on the bit length of the exponent 895. We used a random 1536-bit number as modulus. For simplicity reasons, we rounded the percentages of computation times.

Operation	**Time**	**# Ops**	**% (time)**
Multiplication	4653ms	9	40%
Addition	2988ms	36	26%
ModSquare	243ms	27	2%
ModExp	4308ms	10	37%
Pseudo RNG	1088ms	16	9%
True RNG	815ms	1	7%
Addition	581ms	7	5%
Digest	220ms	10	2%
Total	11665ms		100%

Table 4: Computation time comparison split up into the different low-level operations.

Table 4 shows that the addition, as a part of the multiplication and in various locations in the protocol, accounts for 31% of the overall computation time. The pseudo randomness generation accounts for roughly 9% of the computation time.

5 Conclusion

We present the first efficient implementation of an anonymous credential system on a standard Java Card. Our system nurtures sustainable secondary use of the user's identity because of the multi-use unlinkability of the credential system. Our system performs the entire computation on card and independently from a

potentially malicious terminal. Therefore, we fulfill our major requirement for an autonomous trust root. In addition, our anonymous credential system offers long-term certificates and, therefore, does not require updates when doing many unlinkable proofs.

Our Java Card implementation is capable of efficient proofs of possession of identity credentials. Even though our implementation is able to include several attributes in a credential, we opt to trust the hardware for attribute statements, because this results in constant and low transaction times. We present this method as means to handle range proofs on a Java Card, as traditional Boudot range proofs are beyond reach of current cards. We propose to combine this with an efficient anonymous card revocation mechanism.

As limitations of our solution, we note that a terminal can attempt to send multiple policy requests to infer the user's data (see the autonomous trust root discussion in Section 2.1). We believe that this is orthogonal to the implementation of an anonymous credential system and requires further research. Even though our anonymous credential system provides multi-use unlinkability, we note the potential risk that a terminal may identify the Java Card by other means. A card could, for instance, contain further applets that disclose traceable information, such as a serial number. The card's hardware may also be traced by low-level information and finger printing.

In conclusion, we are confident that our solution has overcome the final technical barrier to establish privacy-preserving eID cards.

6 Outlook

We present a possible extension of our approach that allows for efficiently proving multiple finite-set attributes. In addition, we throw a glance at the future of smart cards in general, and Java Cards in particular.

6.1 Camenisch-Groß Attribute Encoding

Camenisch and Groß [7] proposed a first approach to achieve higher efficiency with eID cards and CL signatures by encoding binary and finite-set attributes in a single attribute base. This reduces the number of attribute bases and, therefore, of exponentiations by the number of prime-encodable attributes. In addition it allows for efficient *not*, *conjunction* and *disjunction* proofs. To be precise, they encode binary and finite-set attribute values as prime numbers e_j, and condense them in a dedicated attribute base as product exponent $E = \prod_j e_j$, here at base R_1. To disclose a conjunction of prime-encoded values, one discloses the prime representation and proves knowledge of the remainder.

This method is currently realized for the Identity Mixer library on the PC and also suitable for the anonymous credential system on Java Card. In this case, the system requires one additional base for the prime-encoded attributes. AND-

proofs of any number of binary or finite-set attributes will then cost one additional exponentiation.[14]

6.2 Future Smart Cards

Our system has very small hardware requirements. This allows us to implement the anonymous credential system on a smart card that is similar to those currently used in eID production. Our hardware is far from the upper end of the current technology spectrum. This supports our take-home message that todays eID cards are powerful enough to perform advanced privacy-preserving computations.

In addition, it gives us room to plan for the future. Whereas our current implementation copes with 16KB of EEPROM, 2KB of RAM, and a 3.57MHz 8-bit CPU[15], most recent cards are equipped with up to 1MB of EEPROM, 32KB of RAM and a 66MHz, 16-bit CPU[16]. Furthermore, the trends in smart card technology let us predict what smart cards may be chosen for future eID proposals. Those smart cards are clearly able to host further credential system features, such as verifiable encryption and e-cash-based frequency boundaries.

6.3 Java Card 3.0 Standard

The Java Card 3.0 standard as released in April 2008 by Sun Microsystems may also benefit our endeavor, because it allows Java Cards to hold RSA private keys in transient memory. Our current implementation makes use of a JCOP-specific library that allows us to store RSA keys in RAM instead of protected EEPROM, which has major influence on the computation times. With cards implementing the new standard, consequently, we can implement our construction fully standard-compliant on Java Card 3.0 cards. Such cards are to be released in 2009.

7 Acknowledgment

We would like to thank the BlueZ group from the IBM Research Zurich Lab for their continuous support and extremely helpful insight. In particular, we valued the discussions with Michael Baentsch (eID scenarios), Thomas Eirich (fast addition), Thorsten Kramp (fast computations on JCOP), Michael Kuyper (JCOP environment), Michael Osborne (eID scenarios, card support and personalization), Tamas Visegrady (fast computations and crypto acceleration on JCOP), and Thomas Weigold (JCOP environment, fast addition). Without their expertise

[14] Based on the performance measurements of Table 4, we predict that the inclusion of Camenisch-Groß encoding and its AND-proofs will add 1684ms transaction time at a modulus bit length of 1536 bits. This accounts for the modular exponentiation, required multiplications, additions and PRNG calls. Of this, 1016ms are pre-computation, 668ms are policy-dependent. Note that these are upper bounds for the full exponent length.

[15] NXP JCOP v2.2/41

[16] Infineon, SLE88 family

in the domain of smart cards and authentication solution, this work would have not been possible. We also appreciated the support of Doug Dykeman and Peter Buhler for this exploratory research project.

This work has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483. Also, Victor Shoup has done his work at IBM Research and is supported by NSF award number CNS-0716690.

References

[1] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In *Security and Cryptography for Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125, Berlin, 2006. Springer.

[2] Josep M. Balasch. Smart card implementation of anonymous credentials. Master's thesis, K.U.Leuven, Belgium, Leuven, Belgium, 2008.

[3] Patrik Bichsel. Theft and misuse protection for anonymous credentials. Master's thesis, ETH Zürich, Switzerland, November 2007.

[4] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology: CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[5] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.

[6] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 225–234. ACM Press, 2004.

[7] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul Syverson, and Somesh Jha, editors, *Proc. of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 345–356, Alexandria, VA, USA, November 2008. ACM Press.

[8] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, March 2001. Springer.

[9] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology: CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 2004. Springer.

[10] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In Vijay Atluri, editor, *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, Washington, DC, USA, November 2002. ACM Press.

[11] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[12] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In M. Odlyzko, editor, *Advances in Cryptology: CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer, 1987.

[13] Common Criteria Portal. Common criteria for information technology security evaluation. [online; 18 April 2009]. `http://www.commoncriteriaportal.org/`.

[14] Luuk Danes. Smart card integration in the pseudonym system Idemix. Master's thesis, University of Groningen, Mathematics Department, Groningen, Netherlands, 2007.

[15] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology: CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.

[16] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison-Wesley, 1995. Elements of reusable object-oriented software.

[17] Xavier Huysmans. Privacy-friendly identity management in eGovernment. In *The Future of Identity in the Information Society*, volume 262/2008 of *IFIP International Federation for Information Processing*, pages 245–258. IFIP, Springer, June 2008.

[18] IBM. JCOP - the IBM GlobalPlatform JavaCard™implementation. [online; 16 April 2009], February 2002. `ftp://ftp.software.ibm.com/software/pervasive/info/JCOP_Family.pdf`.

[19] IBM. Cryptographic protocols of the Identity Mixer library, v. 1.0. IBM Research Report RZ3730, IBM Research, 2009. `http://domino.research.ibm.com/library/cyberdig.nsf/index.html`.

[20] Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*. Springer, 1999.

[21] Philips. mifare proX P8RF5016. [online; 18 April 2009], May 2003. http://smartdata.usbid.com/datasheets/usbid/2005/2005-q2/sfs051814.pdf.

[22] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[23] Sun Microsystems. Java Card platform specification 2.2.1. http://java.sun.com/javacard/specs.html, October 2003. Specification.

Mixing Identities with Ease

Publication Data

Patrik Bichsel and Jan Camenisch. Mixing identities with ease. In Evelyne De Leeuw, Simone Fischer-Hübner, and Lothar Fritsch, editors, *IFIP Working Conference on Policies & Research in Identity Management (IDMAN)*, volume 343 of *IFIP Advances in Information and Communication Technology*, pages 1–17, Oslo, Norway, November 2010. Springer.

Contributions

- Principal author.
- Credential system overview, architecture, definition of components.

Copyright

Original version: `http://dx.doi.org/10.1007/978-3-642-17303-5_1`

Mixing Identities with Ease

Patrik Bichsel and Jan Camenisch*

IBM Research – Zurich, Switzerland
{pbi,jca}@zurich.ibm.com

Abstract. Anonymous credential systems are a key ingredient for a secure and privacy protecting electronic world. In their full-fledged form, they can realize a broad range of requirements of authentication systems. However, these many features result in a complex system that can be difficult to use. In this paper, we aim to make credential systems easier to employ by providing an architecture and high-level specifications for the different components, transactions and features of the identity mixer anonymous credential system. The specifications abstract away the cryptographic details but they are still sufficiently concrete to enable all features. We demonstrate the use of our framework by applying it to an e-cash scenario.

Key words: Anonymous credential systems, Java Card, privacy-enhancing systems, smart card.

1 Introduction

We all increasingly use electronic services in our daily lives. To do so, we have no choice but to provide plenty of personal information for authorization, billing purposes, or as part of the terms and conditions of service providers. Dispersing all these personal information erodes our privacy and puts us at risk of abuse of this information by criminals. Therefore, these services and their authentication mechanisms should be built in a way that minimizes the disclosed personal information. Indeed, over the past decades, the research community has come up with a large number of privacy-enhancing technologies that can be employed to this end.

A key privacy-enhancing technology are anonymous credential systems [5, 14, 19]. In their basic form, they allow a user to obtain a credential from an issuing authority, attesting to her attributes such as her birth date or access rights. Later, she can use the credential to selectively reveal a subset of the attested attributes, without revealing *any* other information (*selective disclosure*). In particular, even if she uses the same credential repeatedly, the different uses cannot be linked to

*This work has been funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483.

each other. It has been proven that anonymous credentials can be used in practice today (even on Java Cards [2]) and publicly available implementations exist (e.g., `www.primelife.eu/results/opensource/33-idemix`).

The literature provides a number of cryptographic building blocks that allow one to expand this basic functionality; in fact, many of them are needed to meet the practical requirements of a modern public key infrastructure. These include:

Property proofs about attributes allow a credential owner to prove properties about her attributes such as that one attribute is larger than another one (even if they are contained in different credentials). This allows an owner to prove, for example, that her age lies in a certain range [8], or that an attribute is a member of a given set [9].

Usage limitation such as ensuring that an owner can use a credential (i.e., proof ownership of a credential) only a limited number of times (e.g., for e-cash) [11] or a number of times within some context [10, 12] (e.g., twice per hour or once per election). Furthermore, using domain specific pseudonyms enables the implementation of usage restrictions as it makes a user linkable within a given domain.

Revocation of credentials can be implemented using dynamic accumulators [13, 16] or a form of credential revocation lists [3,6,21]. This is necessary for instance to withdraw the right associated with the ownership of the credential or after leakage of the master secret of a user.

Revocation of anonymity in case of abuse of (the rights granted by) a credential can be implemented using techniques from [14].

Verifiable encryption of attributes under some third party's public key [17]. This feature constitutes a generalization of anonymity revocation assuming the user's identity is an attribute encrypted for the party in charge of anonymity revocation. It is a means to control the dispersal of attributes using a trusted entity.

These mechanisms can be combined in various ways. Thereby they allow us to build a multitude of privacy-enhancing applications such as anonymous e-cash, petition systems, pseudonymous reputations systems, or anonymous and oblivious access control systems. It is an enormous challenge to find the balance between offering the whole spectrum of functionality and abstracting away from the cryptographic details when implementing an anonymous credential system. Furthermore, when designing the application programming interface we should require no knowledge of cryptography but only familiarity with the concepts that it realizes. However, reducing complexity bears the risk of tailoring the library towards certain application scenarios which we must avoid. In addition, we require our specifications to be extensible and to go along with current standards.

At IBM Research – Zurich, we have implemented most of the protocols and mechanisms described before. This implementation has been growing over the last couple of years and it has been re-designed and re-implemented several times, the current publicly available version is the forth complete iteration. We were fortunate to receive feed back from a considerable number of universities who have used different versions of our code to build various prototypes. Also, our code has been used in the PRIME and PrimeLife projects to build prototypes, which allowed us to test and discuss our implementation. We believe that the current version provides a good compromise between providing access to the features while ensuring the usability for application developers.

This paper describes the architecture and specification languages for all the interactions of our anonymous credential system called *Identity Mixer*. Due to its generality, the architecture and specification languages also apply to other anonymous credential systems supporting (a subset of) the described features including the one by Brands [5]. In addition, our proposal is extensible, that is, we allow for the specification of low-level features (e.g., commitments, pseudonyms, and verifiable encryption) that can be utilized to implement a high-level functionality (e.g., reputation system). This fosters the usage of the various functionalities described before and simplifies building applications upon them.

We refer to [22] for the complete set of the specification languages for the components of an anonymous credential system. Here we will discuss the most complex ones and depict them in a human readable pseudo code form rather than providing the XML version used by our implementation. We will incorporate our specification language in the next release of *Identity Mixer* (`www.primelife.eu/results/opensource/33-idemix`), where several examples for each component will be available. We will demonstrate our framework by elaborating the example of building an e-cash scenario.

Related Work.

Camenisch and Van Herreweghen [18] describe the basic functions and protocols of an anonymous credential system and define the APIs for them. The system they describe provides only the very basic functionalities (i.e., selective disclosure and anonymity revocation). We provide much more extensive (and less general) specifications at a slightly lower level, that is, we do not directly specify anonymity revocation but provide the more flexible verifiable encryption primitive that can be used for the same purpose (cf. Section 1).

Bangerter et al. [1] provide a cryptographic framework that allows security researchers to design privacy-protecting systems and protocols. In this work we go further: we describe our (Java) implementation of all the building blocks described by Bangerter et al. and describe the architecture and specification languages that enable the design and realization of privacy-protecting systems based on our *Identity Mixer* library.

There are various approaches to specify cryptographic objects such as credentials or authentication information. We provide a specification that is general enough to allow to incorporate, for example, X.509 certificates. On the other hand our proof specification could be extended to comply with the OASIS SAML standard. Consequently, we align very well with current standards while still extending their current functionality to a full-fledged anonymous credential system.

Finally, Microsoft has recently released the protocol specification for U-Prove [7], the credential scheme by Brands [5]. That document specifies the cryptographic protocol for issuing credentials and proving possession of a credential with selective attribute disclosure. We provide a much more extensive specification as *Identity Mixer* allows for more features compared to U-Prove (cf. Section 5).

Organization of this Paper.

In Section 2 we give a high-level description of anonymous credential systems. Next, we describe the architecture in Section 3, which consists of (1) a description of the different components of the *Identity Mixer* (*Idemix*) credential system, (2) a detailed analysis of how those components are used in the *Idemix* protocols, and (3) the specification language for the components. We give an example showing how we make use of the specifications to realize an e-cash scheme in Section 4. Section 5 provides a comparison to the U-Prove specification finally we provide an outlook on the integration with current authentication technology in Section 6.

2 Overview of an Anonymous Credential System

An anonymous credential system involves the roles of *issuers*, *recipients*, *provers* and *verifiers* (or *relying parties*). Parties acting in those roles execute the issuing protocol, where a credential for the recipient is created by the issuer, or the proving protocol, where the owner creates a proof on behalf of the verifier. An entity (for example, user, company, government) can assume any role during each protocol run. For instance, a company can act as verifier and run the proof protocol with a user before assuming the role of the issuer and running the issuance protocol (possibly with the same user). Finally, an extended credential system requires the role of trusted third parties who performs tasks such as anonymity revocation, credential revocation, or decryption of (verifiably) encrypted attributes. Usually organizations or governments assume the roles the issuer, verifier and trusted party, and natural persons the ones of recipient and prover.

Note, all parties in an anonymous credential system agree on general system parameters that define the bit length of all relevant parameters as well as the groups that will be used. In practice, these parameters can be distributed together with the code and they must be authenticated.

To participate a user needs to choose her *master secret key* based on the group parameters of the system. This secret allows her to derive pseudonyms, which she can use similar to a session identifier, that is, it allows the communication partner to link the actions of the user. However, the user can create new pseudonyms at her discretion and all pseudonyms are unlinkable unless the user proves that they are based on the same master secret key. Certain scenarios require one user only having one pseudonym with an organization, where we call such pseudonym a domain pseudonym. In addition to being used for pseudonym generation, the master secret will be encoded into every credential. This constitutes a sharing prevention mechanism as sharing one credential implies sharing all credentials of a user.

The setup procedure for issuers and trusted parties consists of generating public key pairs, create a specification of the services they offer and publish the specification as well as the public key. As an example, an issuer runs the issuer key generation and publishes the structure(s) of the credential(s) it is willing to issue together with its public key.

Let us now elaborate on the issuing and the proving protocol. The credential *issuance protocol* is carried out between an issuer and a recipient with the result of the recipient having a credential. The credential consists of a set of attribute values as well as cryptographic information that allows the owner of the credential (i.e., the recipient) to create a *proof of possession* (also called 'proof of ownership' or 'proof'). When encoding the values into a credential, the issuer and recipient agree on which values the issuer learns and which will remain unknown to it, that is, they agree on a credential structure. In addition, they agree on the values that will be encoded.

The *proving protocol* requires a prover and a verifier to interact, that is, the owner of one or several credentials acts as prover in the communication with a verifier. Firstly, the entities define (interactively) what statement will be proved about which attribute value. Secondly, the prover compiles a cryptographic proof that complies with the statements negotiated before. Thirdly, the verifier checks if the given proof is compiled correctly. The first step is a very elaborate process that is outside of the scope of this paper. To indicate the complexity remember that a proof can range from merely proving possession of a credential issued by some issuer to proving detailed statements about the individual attributes. Our specification focuses on the language that expresses the results from the negotiation phase as well as the second and third step from before. The difficulties here lie in the fact that a proof may be linked to a pseudonym of the user's choice or it may release a verifiable encryption of some attribute value under a third party's public key. In addition, we need to be able to express statements about attributes that will be proved. Finally, the protocols for proving possession of credentials and issuing credentials may be combined. In particular, before issuing a new credential, the issuer may require the recipient to release certified attribute values, that is, prove that she holds a credential issued by another party.

3 Architecture & Specifications

In this section we first discuss the components of *Idemix*, then we show how the components are used in the protocols, and finally we provide the specification of the objects used in those protocols. In particular, we introduce the specification languages for the information that needs to be passed between participants.

3.1 Components of *Idemix*

An extended anonymous credential system consists of many components. We will introduce them starting with the attributes that are contained in credentials. Continuing with the credentials we will finish the discussion with the optional components such as commitments and pseudonyms, which are used to implement extensions.

Attributes.

We denote an attribute a_i as the tuple consisting of *name*, *value* and *type*, that is, $a_i = \{n_i, v_i, t_i\}$. The name must be unique within its scope (e.g., a credential structure or a commitment), which will allow us to refer to the attribute using that name and the scope. The value refers to the content of the attribute, which is encoded as defined by the type. For each type we define a mapping from the content value to a value that can be used in the cryptographic constructions. Currently, *Idemix* supports the attribute types *string*, *int*, *date1900s*, and *enum*. Encoding a string to be used in a group $\mathbb{G}$ with generator g can be achieved by use of a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{G}$. Integers do not require such mapping unless they are larger than the order of the group used by *Idemix*. In such case, the value will be encoded into several attributes. We chose the granularity of the currently implemented date type as a second and set the origin to 1.1.1900. Enumerated attributes are mapped using a distinct prime according to the description in [9].

Credentials.

We denote the set of attributes together with the corresponding cryptographic information as credentials. We classify attributes contained in credentials depending on which party knows the value of an attribute. More concretely, the owner of a credential always knows all attribute values but the issuer or the verifier might not be aware of certain values. During the issuance of a credential we distinguish three sets of attributes as the issuer might *know* a value, have a *commitment* of the value, or the value might be completely *hidden* to him. Let us denote these sets of attributes by A_k, A_c, and A_h, respectively. Note that the user's master secret, as introduced in Section 2, is always contained in A_h.

When creating a proof of possession of credentials, the user has the possibility to reveal only a selected set of attributes. Therefore, we distinguish the *revealed*

attributes, which will be learned by the verifier, from the *unrevealed* attributes. We call the two sets of attributes during the proving protocol A_r and $A_{\overline{r}}$. Note, that each attribute can be assigned to either A_r or $A_{\overline{r}}$ independently of all previous protocols and, in particular, independently of the issuing protocol.

Commitments and Representations of Group Elements.

With commitments [20] a user can commit to a value v, which we denote as $C \leftarrow \mathrm{Comm}(v)$. The commitment has a hiding and a binding property, where hiding refers to the recipient not being able to infer information about v given C and binding refers to the committer not being able to convince a recipient that $C = \mathrm{Comm}(v')$ for a $v' \neq v$. Either of the two properties can be information theoretically achieved where the other will hold computationally.

In our context the bases of a commitment are selected from the bases of the group parameters. When we need the more general version of arbitrarily chosen bases, we call the corresponding object a representation. Where the name is chosen because such objects are representations of group elements w.r.t. other group elements. Representations enable the integration of almost arbitrary proof statements, for example, they are building blocks for building e-cash schemes or (more generally) cloning prevention for credentials.

Pseudonyms and Domain Pseudonyms.

We denote randomized commitments to the master secret as pseudonyms. Thus, a pseudonym is similar to a public key in a traditional PKI and can be used to establish a relation with an organization, for example, in case a user wants an organization to recognize her as a returning user. In contrast to an ordinary public-secret key pair, however, the user can generate an unlimited number of pseudonyms based on the same master secret without the link between those pseudonyms (i.e., the master secret key) becoming apparent.

A domain pseudonym is a special kind of pseudonym in the sense that a user can create exactly one pseudonym w.r.t. one domain. The domain is specified by a verifier, which allows it to enforce usage control for its domain. Note that no two pseudonyms (be it domain or ordinary) are linkable unless a user proves that the underlying master secret key is the same.

3.2 Protocols

The basic building block of *Idemix* is the Camenisch-Lysyanskaya (CL) signature scheme [14, 15] which largely determines the protocols. The signature scheme supports blocks of messages, that is, with a single signature many messages can be signed. In a simple credential, thus, each attribute value is handled as a separate message. A more elaborate idea is to use a compact encoding as in [9] to combine several attribute values into one message. The signature scheme also supports

"blind" signing, where the recipient provides the issuer only with a commitment of the attribute value that will be included in the credential. This is used for attributes of the set A_c. Credentials are always issued to a recipient authenticated with a pseudonym, which ensures that the user's master secret gets "blindly" embedded into the credential.

The distinguishing feature of a CL signature is that it allows a user to prove possession of a signature without revealing the underlying messages or even the signature itself using efficient zero-knowledge proofs of knowledge. Thus, when a prover wants to convince a verifier that she has obtained a credential from an issuer and selectively reveal some of the messages of the credential, she employs a zero-knowledge proof stating that she "knows" a signature by the issuing organization and messages such that signature is valid. As the proof is "zero-knowledge", the user can repeat such a proof as many times as she wants and still it is not possible to link the individual proofs. This statement even holds if the verifier and the issuer pool their information. Of course, a user can also prove possession of several credentials (acquired from different issuers) at once to a verifier and then prove that these credentials share some messages (without revealing the messages).

Let us specify the inputs of the protocols. The issuance protocol requires two inputs for either participant, namely an issuance specification and a set of values. The former is the same for both participants as it defines the issuance process, that is, it links to the definition of the structure of the credential to be issued or the system parameters. The latter are the values assigned to the attributes of the newly created credential. As we pointed out already, the issuer may operate on a set of the values that differs from the one used by the receiver as A_h are not know to it and for values in A_c the issuer only knows a commitment. Note, the issuer may additionally input cryptographic components into the protocol. This is useful when combining the issuance and the proving protocol, for example, the issuer can input a commitment received during a previous run of the proving protocol. It can use the value "sealed" in the commitment as the value of an attribute from the set A_c.

The proving protocol most notably makes use of the proof specification, which the prover and the verifier both must provide as input to the protocol. This specification defines all details of the proof. In addition, it links to the necessary elements for compiling and verifying such proof. The prover provides all credentials referenced in the proof specification as input and the verifier uses the credential structures (cf. Section 3.3) to verify the proof. The cryptographic proof object will be provided to the verifier during the protocol run.

Extensions to the Issuing Protocol.

The issuing protocol has fewer degrees of freedom compared to the proving protocol. This results from the credential structure setting many limitations on the protocol. For instance, the structure defines which attributes belong to which set (i.e., A_k, A_c, or A_h). Still we provide a mechanism for extending the issuing

protocol and use it for implementing a feature that enables efficient updates of the attribute values (A_k) contained in a credential.

Credential Updates. As the issuing protocol is interactive (and for security reasons might need to be executed in a particularly protected environment) re-running it would be impractical in many cases. Rather, *Idemix* offers an non-interactive method to update credentials where the issuer publishes update information for credentials such that attribute values are updated if necessary.

This feature can, for example, be used to implement credential revocation. The mechanism that we have implemented employs epochs for specifying the life time. A credential thus expires and can be re-validated when updating the expiration date (given that the issuer provides such) [13].

Extensions to the Proving Protocol.

The proving protocol requires the prover and the verifier to agree on the attribute values that will be revealed during the proof, that is, all attributes a_i are contained in either A_r or $A_{\overline{r}}$ such that $A_r \cap A_{\overline{r}} = \emptyset$. In addition, the verifier may define what partial information about the attributes $a_i \in A_{\overline{r}}$ has to be proved, where partial information denotes:

Equality. A user can prove equality of attribute values, where the values may be contained in different credentials. In particular, equality proofs can be created among values that are contained in any cryptographic object such as credentials or commitments. As an example, a user can compute a commitment to a value v, with $C \leftarrow \mathrm{Comm}(v)$. Assuming a value v' is contained in a credential, the user can prove that $v = v'$.

Inequality. Allows a user to prove that an attribute value is larger or smaller than a specified constant or another attribute value.

Set Membership. Each attribute that is contained as a compact encoding as described in [9] enables the user to prove that the attribute value does or does not lie in a given set of values.

Pseudonym. A pseudonym allows a user to establish a linkable connection with a verifier. Furthermore, domain pseudonyms allow a verifier to guarantee that each user only registers one pseudonyms w.r.t. his domain.

Verifiable Encryption. A user can specify an encryption public key under which an attribute value contained in a credential shall be (verifiably) encrypted.

3.3 Specification Languages

As pointed out in Section 1, one challenge when designing the specification languages is to abstract from the underlying cryptography while allowing access to flexible primitives that enable developers to build a broad range of systems. The necessity of both parties having certain information (e.g., the credential structure) in order to extract the semantic of a proof presents another difficulty. For instance, a verifier needs to know the issuer of a credential, the attributes names, their order or their encoding within a credential used in a proof. Thus, it is essential to separate the structural information from the data, where the latter may remain unknown to one communication partner. We will not introduce such separation for objects that do not require it (e.g., public keys). Our specifications are in XML and each component uses and XML schema to define its general structure. Note that the information acquired through unsecured channels needs to be authenticated, which can be attained using a traditional PKI.

System and Group Parameters. The system and group parameters are specified as a list of their elements. In addition, the group parameters contain a link to the system parameters. Both system and group parameters need to be authenticated.

Issuer Key Pair. The issuer key pair consists of a public key and a private key, where mostly the specification of the public key is of interest as the private key as it is never communicated. The public key links to the group parameters with respect to which it has been created. Note that apart from the public key, an issuer needs to publish the structures of the credentials it issues. Even though this information might be included in the public key, we suggest to create a designated file.

Credentials. As mentioned earlier, we decompose credentials into a *credential structure*, which is the public part, and the *credential data*, which is private to the owner of the credential. In addition a credential data object is partially populated and sent to the verifier during the proving protocol. This decomposition is needed in the issuing process, when the credential data has not been created, as well as in the verification protocol, where the verifier does only get to know a selected subset of the credential data.

In Fig. 1 we describe the credential structure. It contains (1) references to the XML schema and the issuer public key and (2) information about the structure of a credential, which is needed to extract the semantics of a proof. We partition the latter into the attribute, feature, and implementation specific information.

The attribute information defines name, issuance mode (cf. Section 3.1), and type (e.g., string, enumeration) of each attribute. The feature section contains all relevant information about extensions such as domain pseudonyms. Finally, the implementation specific information is mapping general concepts to the actual implementation. As an example, enumerated attributes are implemented using

```
References{
   Schema = http://www.zurich.ibm.com/security/idemix/credStruct.xsd
   IssuerPublicKey = http://www.ch.ch/passport/ipk/chPassport10.xml
}
Attributes{
   Attribute { FirstName, known, type:string }
   Attribute { LastName, known, type:string }
   Attribute { CivilStatus, known, type:enum }
      { Marriage, Widowed, Divorced  }
   Attribute { Epoch, known, type:int }
}
Features{
   Domain { http://www.ch.ch/passport/v2010 }
   Update  { http://www.ch.ch/passport/v2010/update.xml }
}
Implementation{
   PrimeFactor { CivilStatus:Marriage = 3 }
   PrimeFactor { CivilStatus:Widowed = 7 }
   PrimeFactor { CivilStatus:Divorced = 17 }
   AttributeOrder { FirstName, LastName, CivilStatus, Epoch }
}
```

Figure 1: Example credential structure where we assume this structure being located at `http://www.ch.ch/passport/v2010/chPassport10.xml` and corresponding to a Swiss passport.

prime encoded attributes [9], which requires the assignment of a distinct prime to each possible attribute value.

The credential data most importantly refers to the credential structure that it is based on. In addition, it contains the (randomized) signature and the values of the attributes. Figure 2 shows a credential created according to the structure provided in Fig. 1 and corresponding to the proof specification given in Fig. 3.

Credential Updates. Credential update information is twofold: it consists of (1) general information detailing, e.g., which attributes will be updated, and (2) the information specific to each credential. The former is linked from the credential structure (see Fig. 1), the latter is referenced from the credential (see Fig. 2). Only attributes from the set A_k can be updated.

Commitment and Representation. A commitment and a representation, similar to a credential, consist of a set of values. We assume that the bases for the commitments are listed in the same file as the group parameters. Thus, they use a

```
References{
   Schema = http://www.zurich.ibm.com/security/idemix/cred.xsd
   Structure = http://www.ch.ch/passport/v2010/chPassport10.xml
}
Elements{
   Signature { A:4923...8422, v:3892...3718, e:8439...9239 }
   Features { Update:http://www.ch.ch/passport/v2010/7a3i449.xml }
   Values { FirstName:Patrik; LastName:Bichsel; ... }
}
```

Figure 2: This example shows a Swiss passport credential. Note that the owner who will act as prover knows all the attribute values.

reference to link to the corresponding parameters. The representations, however, list their bases in addition to the list of exponents.

Pseudonym and Domain Pseudonym. As pseudonyms are a special case of commitments, they also contain a reference to the group parameters they make use of. In addition, at the user's side pseudonyms contain the randomization exponent value. Domain pseudonyms additionally link to their domain.

Verifiable Encryption. A verifiable encryption is transferred to a verifier and (if necessary) to the trusted party for decryption. It contains the public key used for the encryption as well as the name used in the proof specification, the label and the ciphertext of the encryption.

Protocol Messages. When running the protocols, there are several messages that are passed between the communication partners. The specification of those objects contains the reference to the schema and the cryptographic values. Each cryptographic value is assigned a name such that the communication partner can retrieve the values easily.

Issuance Specification. Issuing a credential most importantly requires a credential structure and a set of attribute values. As introduced in Section 3.1, the set of values from the issuer may differ from the set of the recipient. More specifically, values of attributes in A_k are known to both recipient and issuer and values of attributes $a_i \in A_h$ are only known to the recipient. For each attribute $a_i \in A_c$ the recipient knows the corresponding value v_i and the issuer only knows a commitment $C \leftarrow \mathrm{Comm}(v_i)$. We define the issuance modes *known*, *hidden*, and *committed* in the credential structure to denote the set an attribute belongs to. The reason for defining the issuance mode in the credential structure is to unify the issuance modes between different recipients.

As the majority of the information used in the issuance protocol is defined by the credential structure, the issuance specification is only needed to implement advanced features (e.g., binding a proving and an issuing protocol).

Proof Specification. The proof specification is more elaborate than the issuing specification as the *Idemix* anonymous credential system supports many features that require specification. Thus, even when using a specific credential we can imagine a broad range of different proofs to be compiled. We start by specifying an identifier for each distinct value that will be included in a proof. Also, we specify the attribute type of each identifier, where the protocol aborts if the type of the identifier and the type of an attribute that it identifies do not match. In addition to identifiers, we allow for constants in the proof specification.

```
Declaration{ id1:unrevealed:string; id2:unrevealed:string;
             id3:unrevealed:int; id4:unrevealed:enum;
             id5:revealed:string}
ProvenStatements{
   Credentials{
      randName1:http://www.ch.ch/passport/v2010/chPassport10.xml =
          { FirstName:id1, LastName:id2, CivilStatus:id4 }
      randName2:http://www.ibm.com/employee/employeeCred.xml =
          { LastName:id2, Position:id5, Band:5, YearsOfEmployment:id3 }
   Enums{           randName1:CivilStatus = or[Marriage, Widowed] }
   Inequalities{    {http://www.ibm.com/employee/ipk.xml, geq[id3,4]} }
   Commitments{     randCommName1 = {id1,id2} }
   Representations{ randRepName = {id5,id2; base1,base2} }
   Pseudonyms{      randNymName; http://www.ibm.com/employee/ }
   VerifiableEncryptions{ {PublicKey1, Label, id2} }
   Message { randMsgName = "Term 1:We will use this data only for ..." }
}
```

Figure 3: Example proof specification using a Swiss passport and an IBM employee credential.

We start the definition of the statements to be proved with a list of credentials that the user proves ownership of (i.e., the user proves knowledge of the underlying master secret key). Next, we assign attribute identifiers or constants to the attributes, where the constants will cause an equality proof w.r.t. the constant. Using the same identifier several times creates an equality proof among those attributes (e.g., `id2` is used within two credentials). Note that we only need to assign an identifier to attributes that are either revealed or partial information is proved.

Apart from the equality proofs all proofs are specified explicitly. Let us begin with the proofs of set membership for enumerated attributes, where the *Idemix* library supports the *and*, *or*, and *not* operators. Those operators can be used on

the set of values specified in the credential structure corresponding to the given credential. Similar to set membership proofs, we allow for inequality proofs, that is, proofs for statements of the form $v_i \circ \hat{v}$, where v_i is an attribute value, $\circ$ is the operator, and $\hat{v}$ can be a constant or another attribute value. Currently, the following operators are implemented: $<$, $>$, $\leq$, and $\geq$. Note that inequality proofs require a reference to group parameters that are to be used, which we provide by linking to an issuer public key.

Relating to the components that we describe in Section 3.1, we specify how commitments, representations, pseudonyms and domain pseudonyms relate to the identifiers. More concretely, the proof specification defines for each exponent of any of those components a corresponding identifier or constant. In addition, all the components of a proof specification are assigned random names, which is mandatory for the identification of the corresponding object in the context of a proof but prevents different proofs from becoming trivially linkable.

4 Example Use Case

In this section we describe how to implement a simple anonymous e-cash scheme with our library to give the reader an idea of how our specifications can be used. We recall the basic idea of anonymous e-cash [4]: The user has an account with the bank under some identity u. To withdraw a coin from the bank, the bank issues the user a credential with the following three attributes ($user_{id}$, $serial_{num}$, $randomizer$) (see Fig. 4). The first one is known to the issuer and is set to U, the other two are not known to the issuer ($serial_{num}, randomizer \in A_h$) and are random values chosen from $\mathbb{Z}_q$ by the user as s and r, where q is the order of the groups used for the pseudonyms (and is part of the system parameters). Let g denote the generator of that group (which is part of the group parameters). The form of the credential can be deducted from Fig. 2.

When the user wants to spend a coin anonymously with a merchant, the user obtains from the merchant a random value $v \in \mathbb{Z}_q$, computes $a = u + rv \pmod q$, generates a representation with g^a being the group element, and g and g^v being the bases. Then she generates a proof to show that she owns a credential from the bank where she reveals the attribute $serial_{num}$ and proves that the attributes $user_{id}$ and $randomizer$ are also appearing in the representation. Figure 5 shows the representation object that contains the representation g^a and the bases g and g^v. We provide the proof specification in Fig. 6.

The user then sends (a, s) along with the proof to the merchant who accepts the coin if the proof verifies and if the representation object was indeed computed correctly. The merchant verifies the latter by re-computing the representation. Later, the merchant will deposit the coin with the bank who debits the merchant if the proof verifies. Also, the bank will check whether s has appeared before. If this is the case it will compute u from the two a and v values present in the two deposits (i.e., solve the two linear equations $a_1 = u + rv_1 \pmod q$ and $a_2 = u + rv_2$

```
References{
   Schema = http://www.zurich.ibm.com/security/idemix/credStruct.xsd
   IssuerPublicKey = http://www.bank.ch/ecash/ipk/credPK.xml
}
Attributes{
   Attribute { UserId, known, type:int }
   Attribute { SerialNum, hidden, type:int }
   Attribute { Randomizer, hidden, type:int }
}
Implementation{
   AttributeOrder { UserId, SerialNum, Randomizer }
}
```

Figure 4: Credential structure of the bank-issued e-coin, where we assume this structure to be located at `http://www.bank.ch/ecash/coin.xml`.

```
References{
   Schema = http://www.zurich.ibm.com/security/idemix/rep.xsd
   Params = http://www.zurich.ibm.com/security/idemix/gp.xml
}
Elements{
   Name = ksdfdsel
   Value = 8483...2939
   Bases { 3342...2059, 4953...3049 }
}
```

Figure 5: This example shows the representation that the user created.

(mod q) for u) and then punish the user u accordingly (e.g., by charging the user for the extra spending).

5 Comparison with the U-Prove Specification

Microsoft has recently released the specification of the U-Prove protocols by Brands and Paquin. The specification describes the interactive issue protocol between the receiver and the issuer and the mechanisms to present and verify tokens to a verifier. The issue specification defines a number of attributes that will be contained in the token. These attributes are known by both the receiver and the issuer. At the end of the protocol, the receiver possesses a signature by the issuer on the attributes. While they call this signature a U-Prove token we would call it a credential. This issuing process is called a blind signature scheme in the literature, that is, the issuer does not learn the token that the receiver obtains but

```
Declaration{ u1:unrevealed:int; u2:unrevealed:int;
             r1:revealed:int }
ProvenStatements{
   Credentials{
      sfeoilsd:http:www.bank.ch/ecash/coin.xml =
          {UserId:u1, SerialNum:r1, Randomizer:u2};
   Representations{ ksdfdsel = {u1,u2; base1,base2} }
}
```

Figure 6: The proof specification for the user when spending the e-coin at a merchant. Note that `base1` $= g$ and `base2` $= g^v$ holds.

only learns the attributes. Brands and Paquin then specify a token presentation algorithm (subset presentation proof). The input to the algorithm is the U-Prove token and the subset of the attributes that shall be disclosed (the other attributes remain hidden to the verifier). The output is an augmented U-Prove token that can then be sent to a verifier who runs the verification procedure to assert the validity of the token w.r.t. to the issuer's public key and the disclosed attributes.

Let us compare the U-Prove specifications to the ones presented in this paper. We do not attempt a cryptographic comparison here. The U-Prove issuing specification realizes a subset of our issuance specification, that is, U-Prove requires that all attributes have to be known by the issuer, whereas in our specification, some attributes can be hidden from the issuer or only be given by commitments. Thus, it is for instance not possible with U-Prove to issue several credentials (tokens) to the same (pseudonymous) user as this requires all credentials containing a (secret) attribute that is essentially the user's secret key and plays the role of a secret identity.

Similarly to the issuing specification, the proof specification (or token presentation specification) of U-Prove realizes a subset of ours. U-Prove only supports that a subset of the attributes can be disclosed, but does not feature proofs of statements about attributes nor does it provide the possibility to release attributes as commitments or verifiable encryption. Furthermore, U-Prove does not support proving possession of several credentials at the same time and proofs among attributes (be they disclosed or not) contained in different credentials. Furthermore, U-Prove has no support for pseudonyms. Let us finally remark that for cryptographic reasons U-Prove tokens can be presented only once (afterwards the different presentations would become linkable to each other). The *Idemix* credentials can be used for an unlimited number of proving protocols without transactions becoming linkable.

Despite the differences in the specifications, it is possible to use U-Prove tokens as part of the framework described in this paper. After all, the U-Prove issuing specification is a means to issue a signature on attributes and it is not hard to extend their specification to cover all the features of our specification.

The resulting U-Prove Tokens would still be valid U-Prove Tokens. The same holds for the U-Prove subset presentation proof specification, but of course such extended U-Prove tokens could no longer be verified according to the (unmodified) U-Prove subset presentation proof as the extended proof will necessarily contain new elements.

6 Conclusion

We have provided an architecture and specifications of the components, protocols, and data formats of the *Idemix* anonymous credential system. The architecture and specification builds the basis to build a large range of applications that require some form of anonymous authentication. We believe that especially our specification language for the various features of the proving protocol is well-suited for making easy use of the different components such as commitment schemes, verifiable encryption, and representations of group elements. That is, with our specification we enable implementation of systems without having an understanding of the cryptography realizing a feature, in fact, we only require knowledge of the very principle. However, this is the minimal understanding that we can require.

We compared our languages to the U-Prove specification and noticed that the more extensive set of features requires a more powerful language. Our language does not manage to hide all this complexity. Still, we hide all the cryptographic complexity (e.g., which groups need to be used or which exponentiation should be computed) while offering access to primitives that proved helpful when designing various privacy friendly systems.

When it comes to established standards we note that the proof specification together with the corresponding (cryptographic) proof values can be seen as the privacy-enhanced equivalent of an X.509 attribute certificate or SAML token: The proof specification defines the attributes that are stated and the proof values correspond to the digital signature on the certificate/token. We could also integrate with X.509 and SAML by using their formats for the specification of the attribute statement and then derive the proof protocol specification from that. The proof specification and the proof values would in this case be the digital signature. This approach would, however, require some changes in the X.509 and SAML specifications. We leave this as future work.

Acknowledgements

We enjoyed numerous discussions about Identity Mixer and its implementation with far too many people to mention all of them here. Thanks to all of you! We are especially grateful to our collaborators at IBM who contributed in numerous ways: Endre Bangerter, Abhilasha Bhargav-Spantzel, Carl Binding, Anthony

Bussani, Thomas Gross, Els van Herreweghen, Thomas S. Heydt-Benjamin, Susan Hohenberger, Phil Janson, Markulf Kohlweiss, Anna Lysyanskaya, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, abhi shelat, Victor Shoup, Dieter Sommer, Claudio Soriente, Michael Waidner, Andreas Wespi, Greg Zaverucha, and Roger Zimmermann.

References

[1] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Proc. of the 12th International Workshop on Security Protocols (SPW)*, Lecture Notes in Computer Science. Springer, 2004.

[2] Patrik Bichsel, Jan Camenisch, Thomas Groß, and Victor Shoup. Anonymous credentials on a standard Java Card. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, pages 600–610, Chicago, IL, USA, November 2009. ACM Press.

[3] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *Proc. of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 168–177, Washington, DC, USA, October 2004. ACM Press.

[4] Stefan Brands. Electronic cash systems based on the representation problem in groups of prime order. In *Advances in Cryptology: CRYPTO 1993*, pages 26.1–26.15, 1993.

[5] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates— Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[6] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Proc. of the 12th Australasian Conference on Information Security and Privacy (ACISP)*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415, Townsville, Australia, July 2007. Springer.

[7] Stefan Brands and Christian Paquin. U-Prove cryptographic specification v1.0. Technical report, Microsoft Research, March 2010.

[8] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology: ASIACRYPT '08*, pages 234–252, 2008.

[9] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul Syverson, and Somesh Jha, editors, *Proc. of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 345–356, Alexandria, VA, USA, November 2008. ACM Press.

[10] 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, *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 201–210, Alexandria, VA, USA, October 2006. ACM Press.

[11] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-cash. In Ronald Cramer, editor, *Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer, 2005.

[12] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In *Proc. of the 5th Conference on Security and Cryptography for Networks (SCN)*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155, 2006.

[13] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In Stanislaw Jarecki and Gene Tsudik, editors, *Proc. of the 12th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500, Irvine, CA, USA, March 2009. Springer.

[14] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, March 2001. Springer.

[15] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Proc. of the 3th Conference on Security and Cryptography for Networks (SCN)*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, September 2003. Springer.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology: CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 2004. Springer.

[17] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. `http://eprint.iacr.org/2002/161`, 2002.

[18] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In Vijay Atluri, editor, *Proc. of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, Washington, DC, USA, November 2002. ACM Press.

[19] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[20] Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology: ASIACRYPT '02*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, December 2002. Springer.

[21] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In Stanislaw Jarecki and Gene Tsudik, editors, *Proc. of the 12th International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480, Irvine, CA, USA, March 2009. Springer.

[22] Security Team, IBM Research – Zurich. Specification of the Identity Mixer cryptographic library (a.k.a. cryptographic protocols of the Identity Mixer library). *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, April 2010.

A Comprehensive Framework Enabling Data-Minimizing Authentication

Publication Data

Patrik Bichsel, Jan Camenisch, and Franz-Stefan Preiss. A comprehensive framework enabling data-minimizing authentication. In Thomas Groß and Kenji Takahashi, editors, *Proc. of the 7th ACM Workshop on Digital Identity Management (DIM)*, pages 13–22, Chicago, IL, USA, November 2011. ACM Press.

Contributions

- Principal author.
- Identity mixer components, comparison of privacy protection.

Copyright

Original version: `http://doi.acm.org/10.1145/2046642.2046647`

A Comprehensive Framework for Data-Minimizing Authentication

P. Bichsel[1], J. Camenisch[1], and F.-S. Preiss[1]

IBM Research – Zurich,
Switzerland.
{pbi,jca,frp}@zurich.ibm.com

Abstract. Classical authentication mechanisms have various drawbacks such as the weak security properties they achieve, users' privacy, service providers' data quality, and the necessary protection of the collected data. Credential-based authentication is a first step towards overcoming these drawbacks. When used with *anonymous credentials*, the personal data disclosed can be reduced to the minimum with respect to a business purpose while improving the assurance of the communicated data. However, this privacy-preserving combination of technologies is not used today. One reason for this lack of adoption is that a comprehensive framework for privacy-enhancing credential-based authentication is not available. In this paper we review the different components of such an authentication framework and show that one remaining missing piece is a translation between high-level authentication policies and the cryptographic token specification level. We close this gap by (1) proposing an adequate claim language for specifying which certified data a user wants to reveal to satisfy a policy and by (2) providing translation algorithms for generating cryptographic evidence from a given claim. For the latter we consider the Identity Mixer and the U-Prove technologies, where we provide detailed translation instructions for the former.

Key words: Access Control, Policy Languages, Privacy, Anonymous Credentials, Digital Credentials.

1 Introduction

Authentication has become an all-embracing requirement in electronic communication. Virtually any online service, from music streaming platforms to online bookstores, requires its users to log in or at least provides added value for registered users. The prevailing method to authenticate is by username and password, a simple and cheap solution most users are familiar with. At account establishment

time, users often have to provide an extensive set of personal information. This erodes the users' privacy and opens the door for criminals misusing the data, for example, for identity theft.

Authentication based on anonymous credentials (proposed by Chaum [11] and implemented by Brands [5] or Camenisch and Lysyanskaya [8]) can overcome these security issues by providing strong authentication, minimizing the personal data required for a transaction, and ensuring correctness of data revealed, all at the same time. Unfortunately, this technology is not deployed today as it is hard to understand and complex to use.

In this paper we aim at removing the barriers for using privacy-friendly authentication. To this end, we review the different pieces of a framework for credential-based authentication. This framework consists of (1) a policy language that allows service providers to express which data about a user they require and which authorities they trust in vouching for such data, (2) mechanisms to generate a specification about how a user wants to satisfy the policy, (3) means to generate evidence supporting this specification, (4) mechanisms for the service provider to verify whether the evidence corresponds to the original policy, and (5) means to verify the evidence itself.

As a concrete instance of the first component, that is, a policy language, we use the credential-based authentication requirements language (CARL) as proposed in [9]. It abstracts the cryptographic aspects of anonymous credentials into well-known authentication concepts. We selected this language as it provides the most comprehensive support for privacy-preserving features. The third and fifth components are provided by several credential technologies, for example, Identity Mixer (*Idemix*), U-Prove, or X.509 that provide evidence generation and verification. However, so far no efforts have been made to provide the components (2) and (4) and thereby to close the gap between high-level authentication languages and low-level technology-dependent specification languages used for the evidence generation and verification algorithms.

We solve this problem by (a) proposing a high-level claim language, (b) showing how this language can be translated into two specific specification languages of anonymous credentials, namely, the *Idemix* proof specification and the U-Prove token specification, and (c) showing how a service provider can verify the evidence it received against the policy it had sent. For translating the claim specification into the *Idemix* proof specification, we show how the different cryptographic building blocks need to be orchestrated to generate evidence realizing the different claim elements. Finally, we discuss how to extend *Idemix* and U-Prove so that all concepts in the claim language (except disjunction) can be realized. We believe that connecting the high-level languages to the specific technologies is a major step towards enabling data-minimizing credential-based authentication and will foster deployment of the technology.

Related Work While currently no other authentication solution provides the comprehensive set of privacy features offered by our framework, the Security Assertion Markup Language (SAML) and WS-Trust as standards for exchanging certified information must be mentioned.

SAML enables a party to send certified attribute information to a recipient. Such attribute information is accumulated within so-called *assertions*, which are similar to what we call credentials. Depending on the underlying certification technology, attributes may be disclosed selectively, however, there is neither support for attribute predicates nor for concepts such as attribute disclosure to third parties. Therefore, without extensions, SAML is not suitable for being used as claim language in data-minimizing authentication scenarios. Ardagna et al. [1] give a brief intuition on how SAML may be extended with those missing features. This extended version of SAML is an alternative to our proposed claim language, however, our language makes deriving claims from CARL policies much easier as it is based on CARL. Further, our language provides a clear grammar that can directly be used for implementing the language. WS-Trust defines protocols to issue, renew and cancel WS-Security tokens. However, WS-Trust is agnostic with respect to the type of token. Users obtain tokens from so called security token services (STS) and present those to web services. Web services publish their security policy by means of the WS-Policy standard. WS-Policy, however, merely standardizes an empty container that needs to be filled by concrete policy languages such as CARL. Thus, while we may use WS-Trust or WS-Policy in a data-minimizing authentication framework, they do not close the existing gap between the different levels of languages we currently have.

2 Preliminaries

Anonymous credentials are an important ingredient of privacy-friendly authentication. Therefore we provide a brief overview about the concepts and technologies that implement such systems, borrowing notation from Camenisch et al. [9].

We consider a *credential* to be a set of attributes together with the values of a specific entity, which we call the *owner* of the credential. Each credential has a type that defines the set of attributes the credential contains. As an example, a credential of type 'passport' would contain the attributes *name*, *address*, and *date of birth*. Further, each credential has an entity, the so-called *issuer*, that vouches for the attribute values. As the certified values identify the owner, we also denote the issuer as *identity provider* (IdP). The reputation of an issuer as well as the issuance process (e.g., with physical presence or not) influence the trustworthiness of a credential.

Credentials can be issued using various technologies such as anonymous credential systems [5, 8], X.509 [13], OpenID [12], SAML [15], or LDAP [19]. Identity providers can vouch for users directly or by means of certification. That is, the issuer either communicates the credential directly to the relying party

(i.e., service provider) or it provides the user with a certified credential she can then show to the relying party. We call these two approaches *online* and *certified* credentials, respectively, and discuss them in the remainder of this section.

2.1 On-Line Credentials

In the case of online credentials, the issuer retains the user's attribute values and when a user wants to use a credential, the relying party and the issuer interact directly. We call such credentials 'online' as the issuer needs to be online for each transaction of a user.

Let us illustrate how online credentials work on the example of OpenID. An OpenID provider, which may be seen as identity provider, stores the user's attribute values, for example, in an database. If the user wants to release any of her attribute values, she relates the relying party to her issuer, that is, her OpenID provider. The latter sends the attributes to the relying party by using a secure channel to transfer the information. Based on the trust of the relying party in the OpenID provider as well as the security provided by the communication channel, the relying party derives the assurance about the communicated attribute values. Note that this information flow does not require certified information to be transferred.

2.2 Certified Credentials

Credential technologies such as X.509 or anonymous credentials use a different approach. They add a certification value to the credential, that is, some form of a digital *signature*. This value allows a user to prove that the issuer vouches for her credential without involving the issuer into the communication with the relying party. From a privacy perspective, this is an important advantage over online credentials as the issuer does not get involved into any transaction of a user. As mentioned before, our main interest lies in credential technologies that support even more privacy-preserving features compared to standard certification technology such as X.509.

Anonymous credential system implementations, more specifically, *Idemix* [18] or U-Prove [16], do offer such additional features. In essence, they allow a user to obtain a signature from an issuer on a number of attributes similar to standard certification technology. One important difference being that the issuer does not necessarily learn the attributes that it certifies.

After a user has obtained a credential she can release the certified attributes to a relying party. In contrast to other certified credentials the signature released to the relying party is *unlinkable* to the signature generated by the issuer. In addition, not all attributes need to be shown for verifying the signature. Anonymous credentials enable a user to only release a subset of the attributes where the signature of the issuer can still be verified by the relying party. This feature is called *selective attribute disclosure*. Another important advantage of anonymous credentials lies

in the fact that properties about attributes can be proven without revealing the attributes themselves. For example, using an anonymous credential containing a user's date of birth allows her to prove the certified statement that she is older than 21 years (provided this is indeed the case) without revealing the exact date itself.

3 Data-Minimizing Authentication

In this section we discuss the different components of credential-based authentication systems [9] and classify them into already existing and missing components. Thereafter we discuss the existing ones in this section and provide the missing components in the remainder of this paper.

Figure 1 depicts the components of a data-minimizing authentication system and the sequence of an authentication transaction. Users own certified credentials (in [9] called "cards") that were previously issued to them from identity providers. The figure depicts how a user wants to use a service (e.g., a teenage chat room) hosted by some server. For using their service, the server requires the user to authenticate with respect to service-specific authentication policy. An important aspect of data-minimizing authentication is that the policy is formulated in terms of properties of the user's credentials. For example, a policy could specify that only users who are teenagers according to an ID card may use the service.

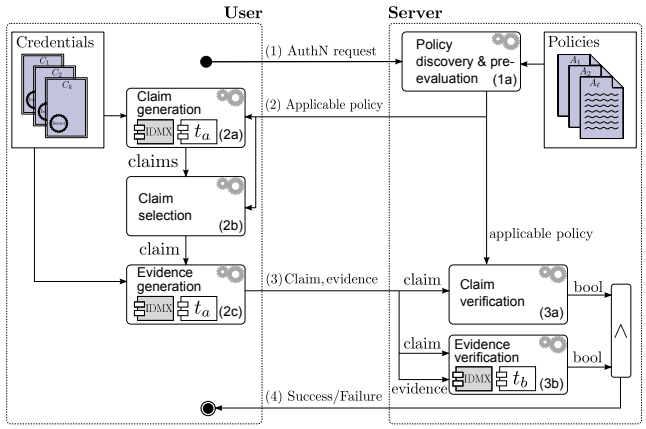

Figure 1: Data-Minimizing Authentication.

Upon receiving an authentication request (1) for a service, a server determines and pre-evaluates the applicable policy (1a) and sends it to the user (2). During this pre-evaluation, references to static content such as the current date are resolved to generate the policy sent. Having received the policy, the user's system determines which *claims*, that is, statements about a subset of attributes of one or

more of the available credentials, can be made that fulfill the given policy (2a). For example, a policy requiring the user to be a teenager according to an ID card may be fulfilled by means of a user's national ID card or her student ID. In doing so, the statement of being a teenager can be made by disclosing the exact date of birth or by a (cryptographic) proof that the birth date lies within the required range, i.e., the claims that a user can make depend on the capabilities of the underlying credential technology. The favored claims are then selected (2b) interactively by the user [3] or automatically by a heuristics capable of finding the most privacy-preserving one. Once the claim is defined, the specific credential technologies are instructed to generate the necessary evidence (also called token) that satisfies this claim. To this end, a technology-specific *proof specification* (e.g., an *Idemix* proof specification or a U-Prove token specification) is generated. Based on this specification the technology-specific *evidence* can be generated (2c). The claim is then sent together with the accompanying evidence to the server (3) who verifies that the claim implies the policy (3a) and checks whether the claim's evidence is valid (3b). Depending on the credential technology, the evidence may be generated and verified with or without the credential issuer being involved. After successful verification, the user is authenticated (4) as someone fulfilling the authentication requirements dictated in the policy. The strength of anonymous credential systems lies in the fact that the server does not learn more than what it strictly requested. For example, the only information the server learns about the user is the fact the she or he is indeed a teenager according to an ID card issued by a trusted identity provider. Thus, the user has minimized the information revealed about herself with respect to the given authentication policy. Ideally, the policy also reflects the minimal information necessary for conducting the scenario at hand.

For implementing such authentication scenario, at least three types of languages are required. First, a policy language to express the server's authentication requirements. Second, a claim language to describe statements about (attributes of) a user's credentials, and third, a technology-specific language that directs the evidence generation. Camenisch et al. [9] provide a suitable policy language with their *credential-based authentication requirements language* (CARL, cf. Section 3.1). The currently existing *Idemix* proof specification [2] language is a technology-specific language for generating evidence (cf. Section 3.2). The language expressing a user's claim, however, is missing. In this paper, we provide this missing piece by defining such language on the basis of CARL. In addition, we extend the *Idemix* proof specification language to make it as expressive as our claim language.

Given this new claim language, in Section 4 we further describe how technology-specific claims can be generated and verified for a given policy. In Section 5 we explain how evidence can be generated and verified for a claim formulated in the specified claim language with a focus on anonymous credential technologies.

3.1 CARL Policy Language

We briefly introduce the CARL policy language [9] as it is the basis for the claim language that we define in Section 4.2. The language allows for expressing authentication requirements in terms of credentials in a technology-agnostic way. One kind of requirement states a predicate over the credential's attributes. The predicate is a Boolean expression that allows for applying logic, comparison and arithmetic operators to attributes, constants or further expressions. Consider the following example policy for a car rental service that illustrates the language features relevant for us:

```
01: own dl::DriversLicense issued-by DeptMotorVehicles
02: own mc::MemberShipCard issued-by CarRentalCo
03: own cc::CreditCard issued-by Amex, Visa
04: own li::LiabilityInsurance issued-by InsuranceCo
05: reveal cc.number
06: reveal li.pNo to EscrowAgent under 'in case of damage'
07: where dl.issueDate ≤ dateSubtrDuration(today(), P3Y) ∧
08:       li.guaranteedAmoutUSD ≥ 30.000 ∧
09:       (mc.status == 'gold' ∨ mc.status == 'silver') ∧
10:       dl.name == li.name
11: sgn 'I agree with the general terms and conditions.'
```

According to this policy, authentic users (a) have their driver's license for at least three years, (b) have *gold* or *silver* membership status with the car rental company, (c) reveal their American Express or Visa credit card number, (d) reveal the policy number of the driver's liability insurance – with a coverage of at least thirty thousand dollars – to a trusted escrow agent who may disclose this number only in case of damage to the car, (e) consent to the general terms and conditions, and (f) have the driver's license and the insurance issued to the same name. Users who fulfill this policy with *Idemix* evidence reveal – aside from their credit card number – merely the *facts* that (1) they do own credentials of the stated types from the stated issuers, (2) those credentials do fulfill the stated predicate and (2) they indeed disclosed the proper values to the escrow agent. In particular, the server neither learns the exact values of the attributes occurring in the policy's where clause nor the number of the insurance policy. However, despite the users' preserved privacy, they are accountable in case of damage due to the information the escrow agent learned. Note that identities such as Visa represent aliases (e.g., to cryptographic keys) that are resolved by the credential technology used to fulfill the policy. We refer to Sections 4 and 5 of [9] for more detailed information on syntax and semantics of CARL, respectively.

3.2 Idemix Proof Specification

The *Idemix* anonymous credential system consists of a number of cryptographic building blocks including signature scheme, encryption, and commitment schemes.

Combined in the right way, those components allow for data-minimizing authentication to be realized. The components as well as their combination is driven by specification languages that abstract from cryptographic details [2]. Thus, to generate *Idemix* evidence that fulfills a given claim, we have to translate claims into the *Idemix* proof specification language. We summarize the main features that *Idemix* provides and that can be realized using the *Idemix* proof specification. One major advantage of anonymous credentials over other credential technology is their ability to *disclose attributes selectively*. The language supports this feature by specifying for each attribute of each credential whether it should be revealed or not. A further advantage is that they allow a user to prove that attributes that are certified in different credentials fulfill a specified relation using so called *cross-credential predicates*.

Note that there is a gap between what has been proposed as features for *Idemix* in the scientific community (e.g., limited spending [7]), what is specified and implemented [18], and what can be expressed with the language proposed in [2]. We start by highlighting the *predicates* that can be expressed by the language as this is the most limited set. Those predicates are (1) equality among attributes, (2) inequality between attributes and constants, and (3) set membership of attributes. First, equality among attributes can cryptographically be proven by using the same values for both attributes within a zero-knowledge proof. The proof specification achieves this feature by using the same so-called *identifier* for several attributes. Second, inequalities allow the user to specify that an attribute is smaller or larger than some constant. In fact, the language supports the operators $<, \leq, \geq$ and $>$ and provides a distinct construct for the specification of the attribute, the constant, and the operator. Third, the language specifies a construct to define attributes that should be used in a set membership proof. Set membership proofs are only available for specially encoded attributes, where the *Idemix* implementation uses prime encoded attributes as proposed by Camenisch and Gross [6].

In addition to the given predicates the library implements concepts such as *disclosure of attributes to a third party*, or *signature on messages*. The former is realized using verifiable encryption as proposed in [10] and the latter amounts to signing the message using the Fiat-Shamir heuristic [14]. As those features are not credential-specific, they are addressed by dedicated statements in the proof specification language. Furthermore, the library implements that *Idemix* proofs can be tied to pseudonyms. However, this concept is currently not reflected in our claim language.

Finally, the proof specification language offers the creation of pseudonyms, commitments and representations. All of which are cryptographic objects that can be employed to implement high level functionality. In this paper we show how those constructs can be used to implement arbitrary arithmetic statements about certified attributes. For example, in Figure 2 we provide the *Idemix* proof specification corresponding to the claim described in Section 4.2. Note that we only indicate the attributes required in the claim.

```
<ProofSpecification [...]>
  <Declaration>
    <AttributeId name="id1" proofMode="unrevealed" type="string"/>
    <AttributeId name="id2" proofMode="unrevealed" type="date"  />
    <AttributeId name="id3" proofMode="revealed"   type="string"/>
    <AttributeId name="id4" proofMode="revealed"   type="int"   />
    <AttributeId name="id5" proofMode="unrevealed" type="int"   />
    <AttributeId name="id6" proofMode="unrevealed" type="int"   />
  </Declaration>

  <Specification>
    <Credentials>
      <Credential name="kdsfjk230fsefj32" ipk="http://www.DeptMotorV.com/ipk.xml"
          credStruct="http://www.un.org/license/driver.xml">
        <Attribute name="name">id1</Attribute>
        <Attribute name="issueDate">id2</Attribute>
        [...]
      </Credential>
      <Credential name="oiwd26ia3m232ewo" ipk="http://www.CarRentalCo.com/ipk.xml"
          credStruct="http://www.CarRentalCo.com/memCard.xml">
        <Attribute name="status">id3</Attribute>
        [...]
      </Credential>
      <Credential name="kdf92fjiu01fj028" ipk="http://www.visa.com/ipk.xml"
          credStruct="http://www.imf.org/creditcard.xml">
        <Attribute name="number">id4</Attribute>
        [...]
      </Credential>
      <Credential name="028dkd93rdlra039" ipk="http://www.InsuranceCo.com/ipk.xml"
          credStruct="http://www.InsucanceCo.com/policy.xml">
        <Attribute name="name">id1</Attribute>
        <Attribute name="pNo">id5</Attribute>
        <Attribute name="guaranteedAmountUSD">id6</Attribute>
        [...]
      </Credential>
    </Credentials>
    <Inequalities>
      <Inequality pk="http://www.DeptMotorV.com/ipk.xml" operator="leq"
          secondArgument="76168">id2 </Inequality>
      <Inequality pk="http://www.InsuranceCo.com/ipk.xml" operator="geq"
          secondArgument="30000">id6</Inequality>
    </Inequalities>
    <VerifiableEncryptions>
      <VerifiableEncryption name="jd2e0asfdkkj3rqq1" label="in case of damage"
          pk="http://www.EscrowAgent.com/vepk.xml">id5</VerifiableEncryption>
    </VerifiableEncryptions>
    <Messages>
      <Message name="d0fsdfkii2fucxzkl">I agree with the general terms and
                conditions.</Message>
    </Messages>
  </Specification>
</ProofSpecification>
```

Figure 2: Idemix proof specification that realizes the example claim specified in Section 4.2.

3.3 Privacy Benefits

The choice of anonymous credential systems as credential technology lies in their privacy benefits over competing technologies. We provide an overview of the privacy features and distinguish again between online and certified credentials. Depending on the definition of privacy, online credentials may have many advantages. That is, if we only care about a user's privacy with respect to to the relying parties, they are a feature-rich and privacy-friendly variant. Their main drawback is that the issuer of the credential (i.e., the IdP) is involved in each transaction, that is, it provides unlinkability when using a credential multiple times *only* with respect to the relying parties. In addition, features such as proving predicates that involve credentials issued by different IdPs can only be achieved using special protocols between those IdPs. Table 1 shows that, except for the generation of the evidence independently from the IdP, all privacy features are provided by online credentials. Note that the restrictions on unlinkability and cross-credential proofs are due to the reasons mentioned before.

Certified credentials such as X.509, *Idemix*, or U-Prove credentials provide a significant advantage over online credentials. They allow a user to proof possession of the credential without involving the IdP, which provides privacy with respect to to the issuer that an online credential can never provide. However, when it comes to the privacy with respect to to the relying party, certified credentials cause some difficulty. On one hand, protection of the user's privacy demands that the latter can change the credential to make it fit for a given purpose. On the other hand the issuer needs to make sure that only specific changes to the credential can be made, that is, the semantics of the credential must remain unchanged.

Feature	**On-line**	**X.509**	**U-Prove**	**Idemix**
Message Signatures	✓	✓	✓	✓
Proof of Ownership	✓	✓	✓	✓
Evidence w/o IdP		✓	✓	✓
Selective Disclosure	✓		✓	✓
Predicate Proofs	✓		✓(+)	✓
Disclosure to Third-Parties	✓		✓(+)	✓
Limited Spending	✓		✓(+)	✓(+)
Cross-Credential Proofs	(✓)		✓(+)	✓
Unlinkable Multi-Use	(✓)			✓

Table 1: Certification technology feature comparison. ✓: Supported. (✓): Limited support. ✓(+): Possible and described in the literature, but currently not implemented.

Table 1 shows that X.509 credentials only support basic signatures and ownership proofs. As mentioned before they do allow to generate evidence without involving the IdP. The reason for providing such limited set of functionality lies in the fact that a user cannot adapt the cryptographic signature.

Implementations of anonymous credentials, such as *Idemix* or U-Prove, provide more flexibility in terms of privacy protection. Both allow a user to selectively reveal attributes that have been certified in a credential (cf. *Selective Disclosure* in Table 1). U-Prove strives for simplicity, thus, it does not provide further privacy protecting features even though the underlying signature scheme would support them. The *Idemix* library[1] is currently the most advanced implementation of a privacy-preserving authentication system. In addition to proofs of ownership and selective release of attributes it supports proofs of predicates over attributes, for example, proving equality among attributes (cf. Section 3.2). Using verifiable encryption, the implementation allows for conditionally disclosing attributes to a (trusted) third party. Limiting the possibilities of spending credentials such as allowing a credential to be used only k times within a certain time interval as proposed in [7] is currently not implemented but it could be added to the current implementation. The distinction between *Idemix* and U-Prove boils down to the fact that an individual *Idemix* credential can be used multiple times without the resulting evidence becoming linkable. We refer to this property as *Unlinkable Multi-Use* in Table 1.

4 Claim Handling

We compare in Section 4.1 different ways on how an authentication policy can be fulfilled. The decision which of the different possibilities is chosen, is mainly driven by the capabilities of the underlying credential technology. Therefore, those capabilities have to be known and considered at the time of claim generation. No matter how the policy is fulfilled, however, an adequate claim language is needed to express the statements made about the credential's attributes.

In general, claims may be accompanied with evidence from different credential technologies. However, requirements across different credentials, so called *cross-credential predicates* cannot be proven using different credential technologies. Although the claim language itself is technology-independent, the expressed statement must be in accordance with the capabilities of the underlying credential technology. In this section we describe how to generate claims for the anonymous credential technologies *Idemix* and U-Prove. Naturally, we are only interested in claims that logically imply the policy, therefore we define claim semantics in Section 4.2.

4.1 Methods To Fulfill A Policy

An authentication policy can be fulfilled in several ways. Intuitively, in case a policy requires the user to show that she owns a driver's license, we can see that the user can comply by providing a proof of such statement or by simply revealing the driver's license information as we do today. On a more conceptual

[1] `http://www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/`

level we can distinguish three methods for complying with a given authentication policy. First, using online credentials, a user can request a claim that closely matches the given policy. Second, the use of standard certified credentials as introduced in Section 2 allows a user to generate a claim without involvement of the IdP. However, standard technology lacks the ability to adapt claims to a given policy, that is, to limit the released information. Third, privacy-preserving certified credentials such as anonymous credentials enable a user to generate a claim specific to a given policy. The privacy implications on each of those options are discussed in more detail in Section 3.3. For all three methods a claim language is needed to describe the generated evidence.

4.2 Claim Language

Just like servers who express their authentication requirements in a policy language, users make authentication statements in a claim language. A claim precisely describes the statements that a user proves by means of technology-dependent evidence. In particular, claims serve as technology-independent input to technology-specific evidence generation modules. Although such claim language is a crucial building block for data-minimizing authentication systems, no adequate claim language has been proposed yet.

The claim language we propose allows a user to state which credentials she owns and what properties those credentials have. Such properties are expressed in terms of a logical predicate over the credential's attributes. Additionally, the language allows users to consent to a certain message or to disclose attributes to a (trusted) third party. The language we propose is intended as counterpart to the CARL policy language (cf. Section 3.1). In fact, most language constructs can be reused, however, three concepts need special attention.

First, for credential ownership (i.e., 'own' lines) CARL policies allow for specifying a list of issuers with disjunctive semantics, that is, ownership can be proven for any of those issuers. As it must be unambiguous for the underlying credential technology which cryptographic key to use for generating the claim's evidence, the claim language just states one issuer. Second, disclosure requests for attributes that are to reveal to the server (i.e., 'reveal' lines without 'to') are only meaningful in policies. A claim must rather fulfill those requests by disclosing the corresponding attribute values. In our language, such attribute disclosure is addressed by stating equality between the respective attribute and its value. Third, CARL supports basic variables that may act as substitute for a number of syntax elements. While being useful in policies, such variables provide no benefit to a claim language. Therefore, the only kind of variable we consider are attribute variables which reference credential attributes.

Figure 3 shows the (left factored) grammar of our claim language. Apart from above mentioned restriction, credential ownership is expressed with 'i own' lines in the same way as 'own' lines in CARL. We prefix the main keywords with 'i' to stress the claim's active statement character. The attribute predicate is

expressed after the 'where' keyword in terms of a Boolean expression. Beside the standard operators of *logics* ($\wedge$, $\vee$ and $\neg$), also *equality*, *inequality* ($\neq$, $>$, etc.) and *arithmetic* operators may be applied to expressions. Expressions may further be (1) attributes qualified with the ID of a previously declared credential (e.g., $dl.issueDate$), (2) constants of data type String, Boolean, Date (e.g., 1984/01/01), Float and Duration (e.g., P3Y represents a period of three years), as well as (3) function calls with expressions as arguments. A type system equivalent to the one of CARL [9, Appendix C] ensures that the predicates are type correct with respect to the data types defined in the credentials' types and the function definitions. The message to sign is given after the 'i sign' keyword. It must be a constant expression that evaluates to data type string. To disclose a list of terms to a third party, the 'i reveal' keyword is used. Although CARL also provides syntax to address limited credential spending, we do not consider this concept here. Consider the following example claim:

```
01: i own dl::DriversLicense issued-by DeptMotorVehicles
02: i own mc::MemberShipCard issued-by CarRentalCo
03: i own cc::CreditCard issued-by Visa
04: i own li::LiabilityInsurance issued-by InsuranceCo
05: where dl.issueDate ≤ dateSubtrDuration(today(), P3Y) ∧
06:       li.guaranteedAmoutUSD ≥ 30.000 ∧
07:       mc.status == 'silver' ∧ dl.name == li.name ∧
08:       cc.number == '1234 5678 9012 3456'
09: i reveal li.pNo to EscrowAgent under 'in case of damage'
10: i sign 'I agree with the general terms and conditions.'
```

This claim is one possible counterpart to the policy given in Section 3.1. Its intent is to fulfill the choices given in the policy with a visa credit card and a membership card of silver status. Additionally, a concrete credit card number is revealed. The functions and their interpretations are specified in an ontology that is commonly agreed upon by the user and the server.

Note that the example claim reveals that the membership status is silver, rather than just saying that it is silver or gold. In general, the latter would be preferable, however, current implementations do not yet allow to prove disjunctive statements (cf. Section 5). Further note that the user may be involved in the selection of the most desirable claim in case (1) multiple credentials are possible to use according to policy (e.g., multiple credit cards), or (2) multiple claims result from one assertion, for example, because a credential technology does not support disjunctive statements.

4.3 Claim Generation

When a user u receives the policy applicable to her authentication request (cf. Step 2 in Figure 1), is has to be determined which claims can be made with respect to her credential portfolio $\mathfrak{P}_u$. To do so, the possible options of assigning credentials

```
Claim     = PrfOfOs+ ['where' Exp] Discl* ['i sign' Exp] ;
PrfOfOs   = 'i own' CredVar '::' URI 'issued-by' URI ;
Discl     = 'i reveal' Term (',' Term)* 'to' URI ;
CredVar   = ID ;
AttrVar   = CredVar '.' ID ;
Term      = AttrVar
          | String | Float | Date | Bool | Duration ;
Exp       = DisjExp ;
DisjExp   = ConjExp ('∨' ConjExp)* ;
ConjExp   = EquExp ('∧' EquExp)* ;
EquExp    = InEquExp ('==' InEquExp)* ;
InEquExp  = AddExp (('≠' | '<' | '>' | '≤' | '≥') AddExp)*;
AddExp    = MultExp (('+' | '−') MultExp)* ;
MultExp   = NegExp (('·' | '÷') NegExp)* ;
NegExp    = ['¬'] SigExp ;
SigExp    = ['+' | '−'] PrimExp ;
PrimExp   = '(' Exp ')'
          | Term
          | ID '(' [Exp (',' Exp)*] ')' ;
ID        = Alpha Alphanum* ;
```

Alpha and Alphanum are alphabetic and alphanumeric characters. The URI after the 'issued-by' keyword must map to a identifier that the underlying credential technology can resolve. The URI after the '::' keyword must map to a credential type. IDs must be different from the used keywords.

Figure 3: Claim Language Grammar.

from $\mathfrak{P}_u$ to all of the credential variables occurring in the policy are calculated. We call one such option a *credential assignment*. An important aspect of the claim generation component is to determine *all* possible credential assignments of a user for a given policy. This is to enable the user to select the most privacy-preserving assignment. In case no credential assignments are found, the user's credentials are not sufficient to fulfill the policy.

Every credential assignment found by above-mentioned algorithm is then transformed into a claim. The transformation, however, is dependent on the technologies of the assignment's credentials. This is due to the varying capabilities the different credential technologies (cf. Section 3.3). In case all credentials in the assignment are of the same technology, the assignment is transformed to a technology-specific claim according to the *Idemix* and U-Prove restrictions given

in the following subsections, respectively. In general it is possible to support the case where the assignment's credentials are of different technologies. However, in this work this is out of scope.

Idemix Claim Restrictions

To generate claims from an assignment that is based on *Idemix* credentials, the capabilities of the *Idemix* technology must be taken into account. The current implementation of *Idemix* supports most of the possible claim statements. In particular, with the extensions to the proof specification that we discuss in this paper, *Idemix* supports all statements with the exception of disjunctive expressions (cf. Tables 1 and 2). Thus, generating claims from an *Idemix*-based assignment is done in the following way. The policy's predicate is first transformed to disjunctive normal form (DNF) and separate claims are generated for all monomials (also called conjunctive clauses, or conjunctions of literals) of the DNF that (a) hold with respect to the given credential assignment, and (b) resemble the given policy (apart from the predicate being only the monomial, not the full predicate). Note that at least one of those monomials does hold, otherwise the assignment finder component would not have produced the assignment. Further note that using a monomial of the predicate's DNF as predicate in the claim is less privacy-preserving than stating the full predicate as it is disclosed which disjunct is proven. To create a claim that resembles the policy, we use a 'copy' of the policy and modify it according to the constraints on the claim language defined in Section 4.2. According to the definition, every credential declaration must have exactly one issuer. This issuer is unambiguously defined as the issuer of the credential that is assigned to the corresponding credential variable via the credential assignment. Further, all attributes, for which a disclosure request exists in the policy, are disclosed by adding an equality expression between the corresponding attribute and its value as additional conjunct to the monomial.

U-Prove Claim Restrictions

The latest specification of U-Prove [16] allows for selective disclosure of attributes, signing messages, and proof of ownership, but does not support features such as predicate proofs and disclosure to third parties (cf. Tables 1 and 2). A policy that includes predicates over attributes or disclosure to third parties, can be fulfilled if all these attributes are fully disclosed to the relying party. To this end, one needs to process the policy's predicate and transform it into a claim predicate in which all attributes occurring in the predicate are selectively disclosed. Claims including cross-credential statements and limited spending cannot be fulfilled with the current specification of U-Prove.

4.4 Claim Verification

A server receiving a claim with accompanying evidence from a user verifies whether this claim indeed implies the initially provided policy (cf. Step 3a in Figure 1). A claim implies a policy if all of the following five conditions hold.

(1) For each disclosure request in the policy there is a corresponding attribute – with the same credential variable and the same attribute variable – disclosed in the claim. (2) The predicate of the policy implies the predicate of the claim's statement. To account for technologies that fulfill the policy's predicate by disclosing all attributes occurring in it (e.g., U-Prove), all the attributes of the policy's predicate that are disclosed in the claim are substituted with the revealed values. Then, if the resulting predicate is constant, it is verified whether it evaluates to true. If so, the predicate is fulfilled. Otherwise the claim does not imply the policy. (3) The credential declarations of the claim's statement imply those of the policy. A claim's credential declaration implies a policy's declaration if (a) their credential variables are equal, and (b) their credential types are equal (for hierarchical credential types, this might be extended to checking whether the claim's credential type is a subtype of the policy's credential type), and (c) the issuer of the claim's declaration is contained in the list of issuers of the policy's declaration. (4) In case the policy requires the signature of a message m, the claim must also contain an 'i sign' statement for m. (5) The set of terms disclosed to third party S_1 in the claim must be a superset of the set of terms that is required to be disclosed to S_1 in the policy.

5 Evidence Handling

In this section we show how *Idemix* and U-Prove evidence is generated and verified for a given claim. In particular, this section elaborates on the components (2c) and (3b) of Figure 1. Note that in the evidence verification we only handle claims that have previously been generated (cf. Section 4.3) and we assume that claims adhere to the restrictions of the respective technology.

For transforming the claim to semantically equivalent evidence, we break our claim language syntax down to a set of building blocks. We therefore only need to show how evidence is generated for those building blocks.

5.1 Claim Building Blocks

In Table 2 we show the building blocks of our claim language and detail which ones are supported by the current implementations of *Idemix* and U-Prove. For *Idemix*, we distinguish between existing support and support introduced through extensions we propose in Section 6. Of particular interest are the building blocks for attribute predicates. We show how every attribute predicate can be rewritten in terms of building blocks. In particular, any attribute predicate with arbitrary logical nesting and negations can be transformed to disjunctive normal form

(DNF), that is, $\bigvee_i \bigwedge_j \ell_{ij}$, where ℓ_{ij} is an atomic expression (AtomicExp in Table 2) with no further logical structure. In DNF negations occur only immediately before atomic expressions. Such negations are eliminated by inverting the respective operators (e.g, $\neg(a < b)$ maps to $a \geq b$). Further, expressions with no further logical structure that do not match any of the building blocks 6 – 15 can be rewritten to do so. For example, the expression $(a.b + c.d) \leq e.f$ can be rewritten to $(a.b + c.d) - e.f \leq 0$, which is building block 14. Atomic expressions that are constant cannot be proven using any credential technology but they can be trivially evaluated.

Note that we distinguish three cases for equality, not equal to, and inequality although they are overlapping. For example, blocks 6 and 7 are instances of 8. The reason being that simpler cases can typically be implemented more efficiently and are already supported, while the more general case is not implemented yet. For instance, block 6 and 7 are currently implemented in *Idemix*, but block 8 is not.

5.2 Idemix Evidence

To generate evidence for a claim with the *Idemix* technology, we assume it only contains expressions that are supported in the implementation extended with our proposals as described in Section 6. In particular, for *Idemix* this means that a claim must not contain disjunctive expressions, while all other constructs are supported. As the current *Idemix* implementation does not support proving of disjunctive expressions, the corresponding *Idemix* claim generator (cf. Section 4.3) only produces claims that have monomials (i.e., conjunctions of Boolean literals) as their statement's formula (or the formula is null). Thus, we assume that the statement of the given claim is a monomial and determine the individual Boolean literals of the statement's formula. In case the formula is null, the list of Boolean literals is empty. Every literal of the monomial is an instance of one of the expression patterns described in the following paragraphs, which describe how a semantically equivalent proof specification can be created.

Generating Idemix Evidence

This section describes how to generate *Idemix*-specific evidence. We show an overview of the supported features (which includes those that require extension to the *Idemix* proof specification) in Table 2.

Proof of Ownership (incl. Selective Disclosure) A proof of ownership is the basic feature of certified credentials. The *Idemix* proof specification provides this functionality by including for each credential declaration a `Credential` tag in the proof specification. This tag contains `Attribute` tags for all attributes that are declared in the credential's type definition, and also allows for referring to the credential in the rest of a proof specification. An attribute tag refers to a

#	Type		Claim Syntax	Examples	Idemix	U-Prove
1	Proof of Ownership		i own $c::\tau$ issued-by I	i own $c::CreditCard$ issued-by VISA	✓	✓
2	Message Signatures		i sign m	i sign 'terms and conditions'	✓	✓
3	Disclosure To Third Parties		i reveal $c.a$ to TTP under ℓ	i reveal $li.pNo$ to ECRAGT under 'damage'	✓	
	Attribute Predicate		where ϕ	*See rows 4 - 15*		
4		Logics	AtomicExp $\wedge$ AtomicExp	$(a.b == c.d) \wedge (e.f < 1984/01/01)$	✓	
5			AtomicExp $\vee$ AtomicExp	$(a.b == c.d) \vee (e.f < 1984/01/01)$		
6		Equality (incl. Selective Disclosure)	$a.b == c.d$		✓	
7			$a.b ==$ ConstExp	$a.b ==$ 'Chicago'; $a.b == 1984/01/01$	✓	✓
8			NonConstExp == ConstExp	$(a.b + 2{\cdot}c.d) == 7$; $log(a.b) == 7$	$\checkmark_+$	
9		Not Equal To	$a.b \neq c.d$		$\checkmark_+$	
10			$a.b \neq$ ConstExp	$a.b \neq 7$; $a.b \neq$ 'male'; $a.b \neq 1984/01/01$	$\checkmark_+$	
11			NonConstExp $\neq$ ConstExp	$(a.b + 2{\cdot}c.d) \neq 7$	$\checkmark_+$	
12		InEquality	$a.b$ Op $c.d$	$a.b < c.d$; $a.b \geq c.d$	$\checkmark_+$	
13			$a.b$ Op ConstExp	$a.b > 8$, $a.b > 1984/01/01$	✓	
14			NonConstExp Op ConstExp	$(a.b - 2{\cdot}c.d) > 25$	$\checkmark_+$	
15		FunctionCall	f(NonConstExp, ...)	$charAt(2, a.b)$	Func.-Dep.	Func.-Dep.

Table 2: Claim building blocks. ✓: Supported in current implementation. $\checkmark_+$: Supported with the extensions described in Section 6. AtomicExp: Any of the building blocks 6 – 15. ConstExp: Expression not containing attribute variables. NonConstExp: Expression containing at least one attribute variable.

corresponding *attribute identifier* that is declared by means of an `AttributeId` tag. Attribute identifiers specify whether the corresponding attribute remains unrevealed or whether it is revealed. In addition, attribute identifiers are used to express equalities among attributes. This is possible even if the attributes are certified in different credentials. In particular, the same attribute identifier is used for all attributes that are equal according to the claim. Note that attribute equality can only be proven if the data types of the attributes match.

Message Signatures A policy may request a user to sign a message, which will be reflected in a claim according to the example in Section 4.2. The *Idemix* proof specification provides a dedicated `Message` element, which allows for a direct translation from the claim.

Attribute Disclosure to Third Parties Conditional attribute disclosure to third parties is also included into the proof specification as a distinct primitive. Consequently, it can be almost employed as directly as message signatures. The essential difference is that a verifiable encryption requires the user to specify the public key of the trusted entity, which she has to (authentically) retrieve before being able to create the encryption. Furthermore, she needs to add the condition under which the decryption is released to the `VerifiableEncryption` element.

Note that to attain a transaction binding, that is, the binding among all attributes that are verifiably encrypted to one another, the verifiable encryption needs to contain all the transaction relevant attributes. This is of importance to make sure that none of the parties in the accountability transaction can change the context, that is, misuse the verifiable encryption. While the claim language allows for disclosing a list of terms, that is, attribute references or constants, the proof specification does not currently do so. Thus, for every attribute reference we create a separate verifiable encryption. Disclosure of constants is currently not supported by the proof specification.

Attribute Predicates Predicates over attributes range from simple expressions that can be directly achieved using a dedicated element in the proof specification to almost arbitrarily complex statements, which need extensions to the proof specification language to be expressable. We explain the transformations for the cases mentioned in Table 2 in Section 5.2. Note that attribute predicate proofs over revealed attributes are not supported. Therefore, all the predicate's attribute variables whose values are revealed have to be replaced with their disclosed values.

Generating Attribute Predicate Evidence

In the following, we introduce the mappings of a claim's attribute predicate to an *Idemix* proof specification. We refer to Section 6 for further details.

Equality Details of equality predicates in a claim are given on Line 6-8 in Table 2. As already mentioned, the proof specification expresses equality among attributes (Line 6) by using the same attribute identifier. Proving equality between an attribute and a given constant (Line 7) amounts to revealing the attribute value. Note that the verifier needs to check whether the value stated in the claim actually corresponds to the revealed value. Finally, equality statements involving arithmetic expressions have to be mapped to commitments and representations. We show a concrete example of the equality $a.b \cdot c.d == e.f$ in Section 6.2.

Not Equal To Statements that express that one value is unequal to another one, are generically hard to proof as technology like *Idemix* is built for proving equality of elements. Still, we can prove a statement $a.b - c.d \neq 0$ rather than $a.b \neq c.d$. While maintaining semantic equivalence we manage to change the statement such that the verifier can check it. This results as we handle the attributes as exponents and the verifier can check that the resulting exponent is not equal to zero.

Using this rational we can transform the statements of Line 9 and 10 into statements that are not equal to zero, that is, $a.b - c.d \neq 0, a.b - \mathsf{ConstExp} \neq 0$. Consequently, using commitments and representations in the generalized form as we explain in Section 6.2, we can express "Not Equal To" statements.

InEquality The inequality operators $<, \leq, \geq$, and $>$ are implemented using Boudot [4] interval proofs. This concept profits from a dedicated element called `Inequality` in the *Idemix* proof specification. A limitation with respect to the claim language is that only integers and dates (which are encoded into integers) are supported.

In addition, the current implementation only allows for comparing unrevealed attributes with (1) constants or (2) revealed attributes. For example, the formula $a.b < c.d$ (of the form indicated in Line 12 in Table 2) can only be proven if either $a.b$ or $c.d$ is revealed. To prove inequality expressions where both attributes are unrevealed, they need to be transformed to $a.b - c.d < 0$. We use commitments for each of the attributes to build a representation that contains the subtraction of the attributes, that is, $a.b - c.d$. Using this value we prove an inequality statement as if the value $a.b - c.d$ were a regular attribute value directly certified within a credential.

Non-constant Expressions In the Lines 8, 11 and 14 of Table 2, the non-constant expressions (NonConstExp) may either be an *arithmetic expression* or a *function call*. We address the former by generating commitments as well as representations such that the desired value, for example, $a.b - 2c.d$, is available as if it were a regular attribute in a credential. The example in Section 6.2 provides an intuition on how the commitments and representations have to be generated.

Support for function calls heavily depends on the concrete function. Subsequently, each function would require a dedicated mapping and possibly even special

algorithms for the functionality to be supported. We envision special functions only to become available once a convincing use case for a particular function is available.

Verifying Idemix Evidence

Once the *Idemix* evidence has been generated and transmitted to the server, the latter needs to translate the user-provided claim into a proof specification the same way the user did. As a result it will get an *Idemix* proof specification that, together with the evidence itself, serves as input to the *Idemix* library (cf. Figure 1). The first step of the verification consists of the *Idemix* library verifying the cryptographic properties of the evidence. The second step consists of the verification of the disclosed attributes, which have to be matched with the constants used in the claim. A particularity of the *Idemix* implementation lies in the fact that strings currently are encoded by employing a hash function. Thus, the disclosed attributes can, in case of strings, not be mapped to their original value, thus, they have to be transmitted from the user to the verifier. Still the verifier needs to assert that the transmitted values match the values revealed in the evidence.

5.3 U-Prove Evidence

To fulfill a claim with U-Prove, our claim language needs to be translated into a U-Prove token as specified in the U-Prove WS-Trust profile [17]. This profile defines which attributes are revealed (in WS-Trust attributes are called claims). Thus, to generate U-Prove evidence in our system, we would need to translate our claim into a set of U-Prove WS-Token specifications, one for each credential (U-Prove token) that shall be used. These specifications then define the attributes that are revealed. Finally, the different U-Prove tokens generated according to these specifications are assembled to build the final evidence.

6 Idemix Proof-Spec Extensions

Table 2 shows basic expression patterns that are theoretically supported by the *Idemix* library but cannot be expressed using the current proof specification language. As we only need to slightly change the languages proposed in [2] in order to provide a substantial improvement to the overall system, we describe those changes here. We provide an intuition on how to extend the *Idemix* proof specification language such that the concepts marked with a $\checkmark_+$in Table 2 are supported.

6.1 Generalized Issuance Process

The design of the proof specification considers a limited issuance scenario, namely, it does not consider that a credential structure is defined by an entity different from an issuer. As we presume that multi-national organizations will specify the format of widely used credentials, we need to specify credential structures independently from issuer-related values. We attain this independence by removing the issuer public key from the credential structure and adding it to each credential in a proof specification. The rational being that a credential structure is independent from an issuer and only a credential, that is, the instantiation of a credential structure, is linked to the issuer.

6.2 Generalized Representations

Considering the definition of representations in [2] we require a set of extensions. More specifically we require that a representation may (1) refer to other elements (e.g., commitments or representations) as its bases, (2) use constant exponents, and (3) re-use an already defined representation object. We use the first and second property to recursively build elements from arithmetic formulas involving certified attributes as referred to in Section 5.2. The last property is needed to establish an equality relation among different representations, which can be used to establish the equality of formulas.

For instance, assume that we need to prove that one attribute is the product of two other attributes, i.e., $a.b \cdot c.d = e.f$ (cf. Line 8 of Table 2). To realize this, we need to generate a commitment to each of these attributes and then prove that the commitment to the third attribute is equal to the commitments to the second attribute, raised to power of the value of the first attribute, times the group element used to randomize commitments raised to power of some integer (the value of which is not of relevance). Similar to this example we can implement more elaborate arithmetic expression by translating them into commitments and representations.

6.3 Relation between U-Prove and Idemix

The signature schemes that underlie U-Prove and *Idemix* are similar. That is, they are both schemes that allow an issuer to (blindly) sign messages where the messages are algebraically encoded as exponents of a representation of an element of an algebraic group. The selective disclosure of attributes is in both cases realized by revealing some exponents (messages) and using a zero-knowledge proof of knowledge of the undisclosed attributes. A zero-knowledge proof can, as the name suggests, convince a verifier of the fact that the prover holds some values without communicating any other information. The difference between the two schemes is that they are based on different cryptographic assumptions and that, due to its cryptographic properties, a U-Prove signature can only be used once

to in a proof (otherwise, the proof is no longer zero-knowledge and transactions become linkable).

The advanced features that *Idemix* provides are all realized with cryptography that uses discrete-logarithms mechanisms. Therefore, they can in principle also be employed for U-Prove if the U-Prove specification [16] were modified accordingly. As a result the specification will presumably become rather complex.

Alternatively, one could also embed U-Prove into the *Idemix* framework [18]. As the *Idemix* implementation treats different cryptographic building blocks such as signature, commitment, and verifiable encryption schemes as different modules and orchestrates them guided by issuing and proof specifications (cf. Section 3.2), the Brands signature scheme [5] could be integrated as an alternative to the CL signature scheme [8].

7 Implementation

We have implemented the data-minimizing authentication framework shown in Figure 1. In particular, we implemented the CARL policy and the claim languages, the pre-evaluation aspect of component (1a) as well as the components (2a), (2c), (3a) and (3b). Although the implementation is open for being used with any credential technology, the components (2c) and (3b) have been instantiated with the evidence generation and verification mechanisms that employ the *Idemix* cryptographic library. Note that message signatures and disclosure to third parties are currently not implemented. All components of the framework have been released as Open Source Software under the Eclipse Public License and are available for download at `http://www.primelife.eu/`, where also the *Idemix* cryptographic library is available.

We are currently working on the implementation of the claim selection (2b) that presents the possible claims to the user so that she can make a selection accordingly. Once this is finished, applications can be built that employ our framework for privacy-friendly authentication. Building such an application could for instance be realized by integrating our framework with an XACML access control engine. Ardagna et al. [1] give an intuition on how this could be done.

8 Conclusion

We presented an important reason that hinders privacy-friendly authentication to be used in practice today, namely, the lack of a framework that utilizes privacy-friendly credential technologies, such as anonymous credentials, for authentication purposes. In this paper we describe all necessary components that allow for an implementation of such framework. We propose a simple claim language that provides adequate expressivity to address the core functionality of anonymous credential systems. Further, we describe how those functionalities are mapped to the concrete evidence specification languages of *Idemix* and U-Prove.

We implemented the proposed framework and connected it to the existing *Idemix* implementation. We show how the latter should be amended to attain the full expressivity of our claim language. Using our implementation has the following advantages, namely, (1) users benefit from significantly increased more privacy, (2) service providers gain in data quality due to the certified data being used, and (3) service providers substantially reduce the risks associated with holding large sets of sensitive information.

We envision to continue this trail of thought and provide a mapping from the claim language to SAML. By using SAML as WS-Trust security token, our data-minimizing authentication scenario may be implemented by means of current standards, which would also benefit its adoption.

References

[1] Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Gregory Neven, Stefano Paraboschi, Franz-Stefan Preiss, Pierangela Samarati, and Mario Verdicchio. Enabling privacy-preserving credential-based access control with XACML and SAML. In *3rd IEEE International Symposium on Trust, Security and Privacy for Emerging Applications (TSP)*, Proc. of the 10th IEEE International Conference on Computer and Information Technology (CIT), pages 1090–1095, Bradford, UK, July 2010. IEEE Computer Society Press.

[2] Patrik Bichsel and Jan Camenisch. Mixing identities with ease. In Evelyne De Leeuw, Simone Fischer-Hübner, and Lothar Fritsch, editors, *IFIP Working Conference on Policies & Research in Identity Management (IDMAN)*, volume 343 of *IFIP Advances in Information and Communication Technology*, pages 1–17, Oslo, Norway, November 2010. Springer.

[3] Patrik Bichsel, Jan Camenisch, Franz-Stefan Preiss, and Dieter Sommer. Dynamically-changing interface for interactive selection of information cards satisfying policy requirements. *IBM Technical Report RZ 3756 (# 99766)*, IBM Research – Zurich, December 2009.

[4] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology: EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Bruges, Belgium, May 2000. Springer.

[5] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.

[6] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul Syverson, and Somesh Jha, editors, *Proc. of*

the 15th ACM Conference on Computer and Communications Security (CCS), pages 345–356, Alexandria, VA, USA, November 2008. ACM Press.

[7] 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, *Proc. of the 13th ACM Conference on Computer and Communications Security (CCS)*, pages 201–210, Alexandria, VA, USA, October 2006. ACM Press.

[8] Jan Camenisch and Anna Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology: EUROCRYPT '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, March 2001. Springer.

[9] Jan Camenisch, Sebastian Mödersheim, Gregory Neven, Franz-Stefan Preiss, and Dieter Sommer. A card requirements language enabling privacy-preserving access control. In Barbara Carminati James B. D. Joshi, editor, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 119–128, Pittsburgh, PA, USA, June 2010. ACM Press.

[10] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. `http://eprint.iacr.org/2002/161`, 2002.

[11] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[12] OpenID Consortium. OpenID authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`, December 2007. Specification.

[13] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[14] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology: CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.

[15] OASIS. Assertions and protocols for the OASIS Security Assertion Markup Language (SAML) v2.0. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`, March 2005. OASIS Standard.

[16] Christian Paquin. U-Prove cryptographic specification V1.1. Technical report, Microsoft Corporation, February 2011.

[17] Christian Paquin. U-Prove WS-Trust Profile V1.0. Technical report, Microsoft Corporation, February 2011.

[18] Security Team, IBM Research – Zurich. Specification of the Identity Mixer cryptographic library (a.k.a. cryptographic protocols of the Identity Mixer library). *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, April 2010.

[19] K. Zeilenga. Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map. RFC 4510 (Proposed Standard), June 2006.

Recognizing Your Digital Friends

Publication Data

Patrik Bichsel, Jan Camenisch, and Mario Verdicchio. Recognizing your digital friends. In *1st International Workshop on Security and Privacy in Social Networks (SPSN)*, IEEE International Conference on Privacy, Security, Risk and Trust (PASSAT), and IEEE International Confernece on Social Computing (SocialCom), pages 1310–1313, Boston, MS, USA, October 2011. IEEE Computer Society Press.

Contributions

- Principal author.
- Concepts.

Copyright

Original version: `http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.159`

Recognizing Your Digital Friends

P. Bichsel[1], J. Camenisch[1], and M. Verdicchio[2]

[1] IBM Research – Zurich,
Switzerland.
{pbi,jca}@zurich.ibm.com
[2] Università degli Studi di Bergamo,
Bergamo, Italy.
mario.verdicchio@unibg.it

Abstract. Personal relationships are more and more managed over digital communication media, and electronic social networks in particular. Digital identity, conceived as a way to characterize and recognize persons on the Internet, has thus taken center stage, although this concept still remains vague in many of its aspects. This work aims at shedding some light on this topic, by sketching a basic conceptual framework, analyzing the issues for Internet users, and proposing possible solutions that promote a better use of digital identity.

Key words: Computer Security, Communication System Security, Computer Network Management, Identity Management Systems, Social Network Services, Authentication.

1 Introduction

The Internet has radically changed the way people interact in recent years. The Web 2.0 wave, in particular, has widened the focus of the users' interest to include not only content, but also the people who have generated it. The huge success of electronic social networks has undoubtedly provided a boost for persons to take center stage on several websites.

Because of the differences between the physical world in which persons exist and interact and the electronic realm of the Internet through which their data is exchanged, several new challenges are posed. This work focuses on the gap between what traditionally characterizes a person and what is made available in digital form. In particular, we focus on how we can recognize a person when we interact with them over the Internet.

In the physical world, a person has some characteristics (e.g., name, hair color and length, facial features) that enable others to identify her, that constitute her *identity*. A question, then, rises on the role of these characteristics in the definition of an identity, and it is legitimate to wonder whether the same concepts and

mechanisms work on the Internet. We will not get into long-going philosophical debates on essential characteristics of entities [9], or attempt to classify them based on whether they change over time [13]. We think that such distinctions are not very significant in the usual practice of identity management: in our view, a characteristic c can enable us to identify a person P, as long as it has not changed since the encounter when we registered c as belonging to P, and c is uniquely characterizing P in the multitude of other people from which we are singling out P.

We take inspiration from how identities of persons are managed in the real world, to build a model of *digital identity* on the Internet, and see whether the criteria we use offline to recognize people are supported. Moreover, we tackle the new issues rising from the lack of physical interaction in the context of the Internet.

2 Basic Concepts

Let us now present the fundamental concepts we are going to use in our proposal. We focus on *persons* that exist in the real world, and whose identities are well defined in the traditional sense: all the characteristics that are normally associated to them are there, and can be checked in the usual way. An example of such a person is a US citizen with a driver's license. Making a comprehensive list of all the characteristics of persons that allow us to identify them is beyond our scope. We focus instead on the restricted set of such characteristics that can be translated in digital form and that a person is willing to divulge on the Internet.

Given a person P, we call *digital identity* (DiD) of P the set of all data that has been published by P (or by an authorized person) on the Internet, and that is the digital counterpart of what people in the physical world would normally use to identify and describe P. For instance, the DiD of a person can be comprised of a Facebook page, a blog on Tumblr, a Twitter page, or any other data created and managed by the person herself, including her email correspondence. A DiD can be a very complex and dynamic set of information that is hardly ever processed all at once. More often, people view only a small fraction of it, in the form of a social network profile, for instance. We call *facet* a subset of a DiD which is presented in a unitary way. The information provided by a facet, possibly in the form of text, pictures, or multimedia files, can be considered as the digital counterpart of what is presented when people meet in the physical world.

3 Digital Identity Issues

Several issues rise in a context where, for the lack of physical contact, persons present themselves through the facets of their DiDs.

3.1 Security

Like every other type of information that is transmitted through the Internet, a digital identity must deal with the problem of security. We can distinguish two types of possible attacks from malicious users, based on *alteration* and *duplication* of facets of a DiD, respectively. Alteration attacks target an already existing facet of a DiD to change it or add new content. For example, it still happens often, especially with small family-run hotels, that we are required to fax our credit card data to make a reservation. If the webpage of the hotel's owner has been altered to show a different fax number, it is easy to imagine the consequences. Duplication attacks aim at creating a new facet designed to look like it is part of the DiD of a person P. The most common example of duplication attack on the Internet are phishing websites, but the problem affects social networks as well. For instance, in front of a page representing P, users are naturally led to think that P has published the displayed data, and manages the page. Such supposition is true in most cases, but it cannot be taken for granted, especially if one considers that digital content can be easily copied and reused. A Facebook page presenting itself as the official fan club of a pop star could trick a considerable number of people into giving out their email address with the pretense of a competition.

The facets in these attacks, whether the result of an alteration or created from scratch, are not part of the DiD of the person P they are referring to, because they have not been published by P or an authorized person. In other words, these facets lack the property we call *authenticity*. To support a correct use of digital identities on the Internet, users need instruments that guarantee the authenticity of the facets they are viewing. That said, a multitude of facets, all showing the same picture, do not automatically imply an attack. This is a way for a person to show that these facets are part of her DiD, that is, a way to support her *recognizability* among users. However, there exists no universally accepted specification on how to represent the fact that different facets are all part of the same DiD, as the websites that host them do not share a uniform data structure for their users. Authenticity implies recognizability, that is, if we are guaranteed about the entities behind the facets, then we know which facets belong together. We need to understand whether the means to check authenticity can also be used to support recognizability without the burden of too restrictive standards for websites.

3.2 Privacy

It has been said that the best way to keep a secret is to never have it. In this context, we may say that the best way to avoid *privacy* issues is to never sign up for anything on the Internet. Still, many voices want to be heard without revealing whom they belong to. The issue here is the exact opposite of what we presented before: some, or possibly all, of the content published on the Internet by a person P may not be supposed to be ascribed to P; in other words, sometimes there is the need for avoiding *linkability* between different facets of a DiD. Such need can rise

in many different contexts: we are not only thinking about a controversial political blog in countries with controlled media, but also much more mundane cases like a teacher who manages a comic book discussion forum and does not wish to be recognized by his students. This may simply look like a call for *anonymity*, but in the context of digital identity, users often have more complex needs. Complete anonymity, in fact, would not serve the purposes of the above-mentioned blogger, for instance: the blog's existence itself relies on the connection between all the entries, which is normally given by the URL at which they are published. Should the blog be transferred to another address because of technical or safety reasons, how could the readers recognize it when it is back online at a different site? We are looking for solutions to support *pseudonymity*, a way to tackle the trade-off between having an easily recognizable DiD and keeping the details on the person behind it private.

4 Dealing with the Issues

Here follow the solutions we propose to tackle the above-mentioned problems with security and privacy of digital identities.

4.1 Secure DiDs

In the physical world, we recognize people mainly by means of their physical attributes. Let us first check the authenticity of a facet by comparing the features it shows with the attributes of the person it refers to.

Authentic attributes. We can identify three different categories of attributes that assert authenticity in the digital domain. Firstly, in case a facet offers the possibility to directly transfer physical attributes, authentication closely resembles the recognition process in the real world. An example is provided by the "hangouts" of Google+, where users can start a video chat. Secondly, even when physical attributes are not shown, certain features, that are tightly bound to a person and hard to copy, can be transmitted over the Internet. For instance, we may recognize a person based on her writing style, humor, or quirks that we perceive during a chat session, or through an email. Finally, if a user has already interacted with a person P, for example, met her in person or called her on the phone, all attributes that are coherent with the information exchanged during the interaction and published in a facet of P's DiD, increase the confidence in the authenticity of the facet. For example, if P mentions her vacations in Rome on the phone with Q, Q gains confidence that the Flickr account with the Colosseum pictures belongs to the DiD of P.

Certified attributes. Another way to approach the problem is to rely on certificates (e.g., X.509 [7], U-Prove [10], *Idemix* [12]), with which P can add certified attributes to her facets. Let us assume P has a credential from her government that certifies her name, first name, and birth date among other

attributes. When registering at a host (e.g., Facebook), P can provide the certified attributes instead of simply inserting them into a Web form. The host would provide a mechanism to distinguish certified from non-certified attributes and show which entity provided the certification. Consequently, a user visiting the facet can verify the set of certified attributes and decide how confident she is in the fact that it authentically represents P. However, this approach imposes several requirements. First of all, the hosts would need to adapt the registration process and incorporate mechanisms for showing the certification of attributes. Moreover, the requirement of possessing a certificate is today only practical for entities such as companies or larger organizations. As governments (e.g., Belgium, Germany) start distributing electronic identity (eID) cards, certified attributes may become available for the general public. Finally, users need to have trust in the certificate issuers of their digital friends as well as in the host of the facet. Note that while the increase in trustworthiness of attributes seems to be coupled with a loss of privacy of a user with respect to the host of a facet, the use of privacy-friendly authentication techniques [4, 8] can eliminate this issue.

Community-certified attributes. Previously, we have focused on mechanisms that are based on a single user verifying the authenticity of a facet. However, after a user has assessed the authenticity of an attribute she could share her findings, for example, by assigning a confidence rating. Such rating or recommendation requires the user to authenticate in order for other users to trust in the rating. Consequently, we may view such rating as a certification provided by a community of users. An approach in bootstrapping trust in the authenticity of attributes has been proposed in [3], where it is used to initiate a public key infrastructure (PKI). Differently from such proposal, we assume that users trust the host, thus, we can use a mechanism that does not rely on cryptography. Still, as with the externally certified attributes, this approach requires the host to offer a system where users can rate attributes and such ratings are properly displayed.

Let us now focus on mechanisms that support recognizability of facets, that is, help show that they belong to one DiD.

Unique reference. A straightforward solution to link several facets is to publish them endowed with a unique reference. This reference is supposed to work as an identifier, showing viewers uniquely of which DiD they are observing one facet, assuming that the reference works across multiple domains of the Internet. A public cryptographic key or a uniform resource identifier (URI) are possible ways to achieve such result. Let us consider the following example. If user U visits some blog on Blogger and sees a comment by John Smith, she should be able to recognize whether he is the John Smith that U knows from Facebook. A unique reference can be established in accordance with the trust model we rely on. If there are trusted hosts, then identity providers have the possibility to endow a DiD with a unique reference using technology like OpenID [11]. When relying on such hosts, we are assuming that the username of the DiD on the trusted host is unique. For the reference to be fully recognizable, it should explicitly include the

trusted host's name, but this is not part of the current practice of many websites, so that Web users see that a John.Smith entered a comment in a blog, but there is no way to automatically establish that it is John.Smith@facebook.com, that is, the John Smith U already knows. Publishing a unique reference requires to adapt the current practice of how facets of DiDs are handled. If hosts of the different facets agreed on a mechanism supporting such solution, this would allow for an automated detection of several facets. Such agreement, although very desirable, looks unlikely. Instead, alternative solutions have emerged. A dedicated service has been introduced that provides a unique reference to all social network activites of an entity called about.me[3]. Another, more simple practice is to publish as an information item within one facet a link to another facet (e.g., Facebook users often post a link to their Flickr account).

Corresponding attributes. The same information published under different facets seems to imply a link among such facets, although the simplicity of copying digital information makes it easy to create a facet that is seemingly equal to another one. It is then important to remark that relying on the equality of general attributes (e.g., the same name in several facets) or on similar information (e.g., different facets stating that they are leaving for vacation) is not *per se* a guarantee. However, the correspondence between a new Skype status message "Vacations in Rome. Yeah!" and new Colosseum pictures on a Flickr page can increase a viewer's confidence in the fact that those facets belong to the same DiD. The measure of such confidence increase should depend on how easily such evidence may be fabricated.

4.2 Private DiDs

The lack of physical contact may look like a disadvantage when it comes to identity management as it induces the need for verification of the authenticity that determines whether a DiD actually represents the implied person. However, this very lack of a physical touch can introduce new and interesting types of interaction. Unless we are in specific lawful contexts requiring a user to release her attributes according to some real-world definition, there is no limit in the choice of her published characteristics. In such situations users are free to create DiDs that present a meaningful coherence and make them look like they represent an existing entity, without actually corresponding to any real person. This is the case, for instance, of the above-mentioned blogger who wants to protect her real identity, but still wants to be represented on the Internet with a DiD. Such DiD thus works as a pseudonym for a person. One can also give up any pretense of realism, and create DiDs with such unrealistic features that it becomes obvious that they are fictional.

Pseudonymous DiDs are also affected by recognizability issues. For instance, the DiD of a blogger may also have a social network page. Users should be able to

[3] `https://about.me/`

recognize that such page belongs to the DiD that also has a blog, while the actual person's privacy is still guaranteed. The mechanisms described in Section 4.1 for recognizing that several facets belong to the same DiD work also for pseudonymous DiDs, although we are in a different context, in which the link between a DiD and the person behind it needs to be kept private. To achieve this goal, users are given two possibilities, according to the trust relations they have with the entity hosting their facets. Let P be a person whose DiD $\mathcal{P}$ has a facet on host H. When P trusts H not to leak any information, her real identity can be considered protected, and all P needs to do to manage her DiD is to authenticate to H. This is usually done by means of a username/password pair. When such trust is missing, P needs to rely on an authentication mechanism, that protects her identity also against H. A possible solution is offered by anonymous credential systems [4]. They prescribe the use of certified attributes, that could maintain the level of assurance H needs, while at the same time allowing P to remain pseudonymous.

Another cryptographic primitive that supports the management of pseudonymous DiDs is *verifiable encryption* [5]. It prescribes that, when entity S communicates an attribute type and its value to entity R, R receives the information encrypted in such a way that it can be decrypted only by a designated mediator, but the attribute type can be nevertheless verified by R. S and R agree on the terms under which the mediator is supposed to decrypt the encrypted attribute value. Verifiable encryption enables pseudonymous DiDs to be passed on. For instance, the above-mentioned blogger, whom we call A, can be substituted by a new author B, without anyone else knowing about the change, as follows. We assume that A verifiably encrypts her public key and publishes it with each post. The host of the blog, then, checks that the public key verifiably encrypted with the previous post matches the public key used to sign the current post, to be assured that the post was submitted by the legitimate author. All A needs to do to pass the authorship to B is to verifiably encrypt B's public key in her last post.

5 Related Work

Researchers from several fields have investigated deeply on the analogies and the differences in the concepts of identity in the physical and in the digital world.

Allison et al. provide an overview of the concept from several different perspectives: legal (authorship and ownership issues), philosophical (logical relations among digital objects), and historical (chronological models and records of the evolution of digital identities) [1]. Cameron attempts to provide a more unified definition of the concept, with a synthesis of all its aspects into a list of "laws of identity" [6].

Other efforts point at singling out the available technologies to implement the principles that are traditionally attached to digital identity. Windley, for instance,

considers the support of digital identity fundamental for businesses on the Internet to succeed, and provides several pointers to existing proposals and standards [13].

When it comes to standard proposals, two main research guidelines can be found in the literature. Low-level computational instruments keep on being elaborated in the context of cryptographic research, to expand the boundaries of what can be provided to users in terms of security and privacy. For instance, the endeavors of Lysyanskaya et al. aim at handling pseudonyms or anonymous access [8]. On a higher level, in the context of distributed system research, standards are proposed to support the expression of identity attributes for authentication and access control purposes, like in OpenID [11], and more and more of these works, see for instance Ardagna et al. [2], consider privacy issues as fundamental.

6 Conclusion and Future Work

Internet users deal with digital identities in a similar way to how people deal with each other's identity in the real world. Nevertheless, the lack of the physical dimension leads to a bigger freedom and anonymity, which allows for new types of identity to rise in the digital context of the Internet. People look for, and find each other based on the attributes that they exchange through their digital counterparts. This work aimed at shedding light on the basic concepts related to digital identities, and proposed solutions based on existing technologies to support recognition of people over the Internet, with an eye on both security against attacks, and privacy for users who intend to stay anonymous.

Our next steps on this research path will deal with digital identities of organizations, which have the peculiarity of either being managed by more than one person at the same time, or by different people throughout their life cycle. We consider this topic particularly interesting, because it calls for a compromise in the trade-off between anonymity of the users on the Internet and the accountability of their actions within their organization.

7 Acknowledgements

This work was partially supported by the EC within the 7th Framework Programme, under grant agreement 257129 (PoSecCo), and by the MIUR in the framework of the PRIN project Gatecom.

References

[1] Arthur Allison, James Currall, Michael Moss, and Susan Stuart. Digital identity matters. *Journal of the American Society for Information Science and Technology*, 56(4):364–372, 2004.

[2] Sabrina Ardagna, Claudio A. and De Capitani di Vimercati, Stefano Paraboschi, Eros Pedrini, and Pierangela Samarati. An XACML-based privacy-centered access control system. In *Proc. of the 1st ACM Workshop on Information Security Governance (WISG)*, pages 49–58, Chicago, IL, USA, November 2009. ACM Press.

[3] Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. Security and trust through electronic social network-based interactions. In *Workshop on Security and Privacy in Online Social Networking (SPOSN)*, volume 4 of *International Conference on Computational Science and Engineering (CSE)*, pages 1002–1007, Vancouver, Canada, August 2009. IEEE Computer Society Press.

[4] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.

[5] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. `http://eprint.iacr.org/2002/161`, 2002.

[6] Kim Cameron. The laws of identity. `http://www.identityblog.com/?page_id=354`, May 2005. *accessed: 15 December 2011.*

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[8] Anna Lysyanskaya, Ronald L. Rivest, and Amit Sahai. Pseudonym systems. In *Proceedings of SAC 1999, volume 1758 of LNCS*, pages 184–199. Springer Verlag, 1999.

[9] Gareth B. Matthews. Aristotelian essentialism. *Philosophy and Phenomenological Research*, 50:251–262, 1990.

[10] Christian Paquin. U-Prove cryptographic specification V1.1. Technical report, Microsoft Corporation, February 2011.

[11] David Recordon and Drummond Reed. OpenID 2.0: A platform for user-centric identity management. In Ari Juels, Marianne Winslett, and Atsuhiro Goto, editors, *Proc. of the 2nd ACM Workshop on Digital Identity Management (DIM)*, pages 11–16, Alexandria, VA, USA, November 2006. ACM Press.

[12] Security Team, IBM Research – Zurich. Specification of the Identity Mixer cryptographic library (a.k.a. cryptographic protocols of the Identity Mixer library). *IBM Technical Report RZ 3730 (# 99740)*, IBM Research – Zurich, April 2010.

[13] Phillip Windley. *Digital Identity.* O'Reilly Media, Inc., Sebastopol, CA, USA, 2005.

6
Publication

Security and Trust through Electronic Social Network-Based Interactions

Publication Data

Patrik Bichsel, Samuel Müller, Franz-Stefan Preiss, Dieter Sommer, and Mario Verdicchio. Security and trust through electronic social network-based interactions. In *Workshop on Security and Privacy in Online Social Networking (SPOSN)*, volume 4 of *International Conference on Computational Science and Engineering (CSE)*, pages 1002–1007, Vancouver, Canada, August 2009. IEEE Computer Society Press.

Contributions

- Principal author.
- Concepts, comparison with WoT, implementation details.

Copyright

Original version: `http://dx.doi.org/10.1109/CSE.2009.417`

Security and Trust through Electronic Social Network-based Interactions

P. Bichsel[1], S. Müller[2], F.-S. Preiss[1], D. Sommer[1], and M. Verdicchio[3] *

[1] IBM Research – Zurich,
Switzerland.
{pbi,frp,dso}@zurich.ibm.com
[2] ETH Zürich,
Zürich, Switzerland.
smueller@inf.ethz.ch
[3] Università degli Studi di Bergamo,
Bergamo, Italy.
mario.verdicchio@unibg.it

Abstract. The success of a Public Key Infrastructure such as the Web of Trust (WoT) heavily depends on its ability to ensure that public keys are used by their legitimate owners, thereby avoiding malicious impersonations. To guarantee this property, the WoT requires users to physically gather, check each other's credentials (e.g., ID cards), to sign the trusted keys, and to subsequently monitor their validity over time. This trust establishment and management procedure is rather cumbersome and, as we believe, the main reason for the limited adoption of the WoT. To overcome this problem, we propose a solution that leverages the intrinsic properties of Electronic Social Networks (ESN) to establish and manage trust in the WoT. In particular, we exploit dynamically changing profile and contact information, as well as interactions among users of ESNs to gain and maintain trust in the legitimacy of key ownerships without the disadvantages of the traditional WoT approach. We see our proposal as an effective way to make security and trust solutions available to a broad audience of non-technical users.

1 Introduction

The Public Key Infrastructure (PKI) paradigm aims at assigning asymmetric cryptographic keys to entities, such as people or organizations, to allow for several security features, including secret communication, rights delegation, and access control. A key distribution mechanism comes with a *trust infrastructure* that

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216483 for the project PrimeLife.

enables the establishment of the authenticity of the binding between a public key and a person. To achieve this result, mechanisms are prescribed to associate metadata with the public key, which allow users to identify the key's owner. One example of such a trust infrastructure is the Web of Trust (WoT) [9]. In a WoT, trust between users is ensured through mutual key signing. In particular, a user verifies the metadata by checking a physical witness, like an ID card or a driver's licence, at gathering events called *key signing parties*. The metadata attached to a user's key thus not only consists of personally identifying information (PII) but also of signatures from people that the user has met at an event and that were willing to personally guarantee that she is the legitimate owner of that key. These procedures are unfortunately far from simple, and thus do not provide an appealing and straightforward way to increase communication security for a wide audience.

The aim of this work is to exploit the widespread and successful Electronic Social Network (ESN) mechanisms for personal information exchange to support the trust establishment and management processes prescribed by the WoT. In this work, we rely on the WoT definition of trust: the confidence in the fact that the PII attached to a public key corresponds with the real identity of the possessor of the key. Our aim is to simplify the processes that establish and manage trust by leveraging the information provided by ESN users in the form of personal data attached to a user's profile and in the form of interaction happening through the ESN. Such information can be exploited by other users to identify the physical person controlling that particular profile in the social network, and thus, to trust that she is the legitimate owner of her public key. Indeed, to achieve this purpose on the sole basis of the WoT mechanisms, users must go through the afore-mentioned cumbersome procedures. The main idea of this work is that by linking an ESN profile to a WoT certificate, one is enabled to exploit the ESN-based information exchange mechanisms to establish trust in people.

The paper is organized as follows: Section 2 motivates our work and gives an overview of the underlying concepts; our solution is detailed in Sections 3 and 4, which show how ESNs can support trust establishment and management, respectively; implementation guidelines are provided in Section 5; the possibility to exploit multiple ESNs is described in Section 6; Section 7 discusses related work; finally, we conclude and give hints on future research in Section 8.

2 Motivation

The WoT has not seen a widespread adoption, and we believe that the complexity of its trust establishment process is the main reason. Moreover, an established WoT is static and its maintenance requires significant effort. For example, if a user loses her private key, she needs to re-establish all her trust relations, which means that she has to re-prove her identity to all the contacts that had trusted

her before the key loss. The second establishment entails the same investments (i.e. face-to-face meetings) as the creation of the original trust relations.

Our proposal for tackling these shortcomings is based on the idea that the interactions over an ESN can work as a valid substitute for the WoT-prescribed key signing parties. The underlying assumption is that mimicking the behavior of a user over a long period of time in an ESN is as hard as forging an identification document. An important characteristic of ESN-based interactions is that they do not consist of a single interaction, but take place over long temporal intervals. This allows for a fine-grained assessment of the ESN members a user interacts with. For example, it is easy to search for a detailed set of personal data which might allow for an episodic impersonation of a user, while it is much more difficult to impersonate her for a longer time span in an ESN as this possibly entails live chatting sessions, message exchanges or uploading of pictures.

In our view, as the information exploited to build a trust relation comes from an ESN, the traditional WoT methods should be integrated into the ESN management system to simplify the necessary key signing processes. After certain conditions have been met (e.g., a number of interactions over a given timespan), the ESN could query a user whether she currently believes that her communication partner is actually the person indicated by the relevant personal data. If the user confirms that, the key signing process could be handled transparently. Our approach prescribes the exploitation of the information exchange over an ESN to establish a trust infrastructure.

The ESN-based approach is beneficial also to the management of the trust infrastructure. Let us again consider the case of a user losing her private key. Assuming that the user can still authenticate herself towards the ESN, it is sufficient to convince the other users that she has lost her key and that they should sign the newly produced key. The users prompted to sign the new key can re-authenticate the person through the ESN in a simple and fast way. This method clearly mitigates the investments of a user after a key loss while maintaining an acceptable level of security.

These considerations shed light on the main advantage of the proposed approach: ESNs provide a simple way for a wide audience, which already exploits its interaction mechanisms, to achieve security and trust properties that traditionally rely on far more complex mechanisms. The following sections show how this result can be achieved.

3 Trust Establishment

In our model, the establishment of trust consists of two steps: (1) *trust assessment*, by which a user u evaluates the confidence she has in the identity of another user v, and, in case this confidence is enough for u to believe that the identifying metadata belongs to v, (2) *trust declaration*, whereby u makes her trust explicit. Therefore, when we say that u trusts v, u has assessed and declared her trust in v.

In the following we define our concept of trust, we describe trust assessment and declaration as performed intuitively by a user and we elaborate on trust assessment through ESN interaction. Finally, we depict how trust assessment can be simulated on the basis of ESN data to allow for meaningful suggestions to the users.

3.1 Definition of Trust

We consider a trust infrastructure E, a set of ESNs $W = \{w_1, w_2, \ldots\}$, a set of users $U = \{u_1, u_2, \ldots\}$ as well as a set of attribute names $A = \{a_1, a_2, \ldots\}$. The trust infrastructure comprises a set of users $U_E \subseteq U$, a set of attributes $A_E \subseteq A$ as well as a (possibly partial) function $a_{E,u}$, mapping the attributes in A_E onto concrete values for $u \in U_E$. Similarly, an ESN w comprises a set of users $U_w \subseteq U$ and a set of attributes $A_w \subseteq A$. Let $F_w \subseteq U_w \times U_w$ be a set of symmetric friendship relations between users in w: if a user $u \in U_w$ is in a friendship relation with a user $v \in U_w$, we have $(u, v) \in F_w$, as well as $(v, u) \in F_w$ because of the symmetry of F_w. Every user $u \in U_w$ has a profile $P_{w,u}$ that contains (1) a (possibly partial) function $a_{w,u}$, mapping the attributes in A_w onto concrete values for u, as well as (2) the set of u's friends in w, defined as $F_{w,u} = \{v \mid (u, v) \in F_w\}$.

We now formalize our concept of trust. A user u trusts another user v when she has *enough confidence* about the fact that v is indeed the person she claims to be according to her trust infrastructure attributes, i.e., that the values $a_{E,v}$ indeed correspond with the values of v's PII. More formally, let $T \subseteq U_E \times U_E$ be a trust relation, where $(u, v) \in T$ means that u trusts v and is denoted as $u \twoheadrightarrow v$. Moreover, let $T_u = \{v \mid (u, v) \in T\}$ be the set of users that u trusts. Trust relations are, in contrast to friendship relations, not symmetric, i.e., $u \twoheadrightarrow v$ does not entail $v \twoheadrightarrow u$. Trust relations are not transitive and we assume trust propagation to be handled by the trust infrastructure (e.g., the WoT). For now, we consider trust to be binary. We will introduce different trust levels in Section 4.2. Where clear from the context, we will omit the indexes that identify the ESN. Note that T is not ESN-specific since we consider the trust relations to be publicly available through the trust infrastructure (like the WoT) the ESNs rely on.

3.2 Trust Assessment

The confidence of u that a given set of attributes belongs to v is strongly evidence-based, i.e., the more evidence u gathers, the higher her confidence about v's identity becomes. The evidence indicating that a user v is indeed who she claims to be takes various forms, such as information, credentials, or characteristics v proves to have as well as actions v performs. Information that v has could be the content of a previous conversation with user u. A valid passport is an example of a credential v might possess, and a recognized voice or style of expression are characteristics that v might prove to have in a phone or chat conversation. An action that v performs might be responding to an e-mail sent to her mail account.

Different pieces of evidence contribute differently to the confidence in a user's identity. For example, the presentation of a valid passport is, due to the high trust in the issuer, considered a much stronger identity evidence compared to a membership card from the local gym. Therefore, the presentation of the passport leads to a higher increase in confidence than showing the membership card. Such increase is very subjective as different users may ascribe different values to the same piece of evidence. In the above mentioned example, people who know the very strict identification procedures at a particular gym will value the relevant membership card more than people who are not familiar with these procedures. In addition, the level of confidence necessary for trusting another user is also a very subjective factor.

In fact, there may also be counter evidence for a user v's identity, i.e., evidence that indicates that a user v is *not* the one she claims to be. However, the only factor we consider for decreasing a user's level of confidence is time, i.e., the level of confidence decreases gradually with time in case u and v not having any interaction.

Our formalism includes $c_u(v) \in \mathbb{R}$, denoting u's level of confidence that the values $a_{E,v}$ correspond to the values of v's PII, and the threshold $t_u \in \mathbb{R}$, indicating how much confidence u needs to consider another user as trusted. Note that because of the before-mentioned considerations on the subjectiveness of these concepts, it is not possible (not even for u herself) to frame either $c_u(v)$ or t_u into an absolute quantitative scale.

3.3 Trust Declaration

Let $w \in W$ be an ESN and $u, v \in U_w$ be users. As soon as $c_u(v) \geq t_u$ holds, u may make this explicit by adding v to her set of trusted users T_u. In order to do so, however, we require $u \in U_E$ and $v \in U_E$. To ensure that u can identify v in both the ESN w and the trust infrastructure E, we require a dedicated attribute which is mapped to the same value (e.g., the hash value of v's public key) in $a_{w,v}$ and $a_{E,v}$.

3.4 Trust Assessment through ESN Interaction

The growing list of data managed by common ESNs comprises, in addition to the profile attributes, friends lists, blog entries, messages, comments, pictures and relevant tags, videos, status messages, etc. These data serve as the evidence that is necessary to perform trust assessment. We also regard the mere interaction between u and v in the ESN, such as conversations, tagging of pictures involving the other user, or commenting on the other user's content, as evidence for their identities. For example, consider u assessing trust in a user v who introduces herself as a former work colleague. The profile, including photos showing common friends, seems legitimate to u. Not yet fully convinced, u engages in a chat conversation with v talking about a past joint event. This information, together with v's writing

style increases u's confidence level significantly such that $c_u(v) \geq t_u$ holds and u declares her trust, i.e., $u \twoheadrightarrow v$.

To keep trust relationships up to date, the ESN might notify a user u about the possibility of user v having reached u's trust threshold and propose to establish a trust relationship. In addition, the ESN should prevent u from taking unwise decisions like declaring trust in a user she did not assess the attributes closely. To do so, the ESN must be enhanced with trust simulation mechanisms.

3.5 Simulating Trust Assessment

The model illustrated in Section 3.2 describes the trust assessment as it is *intuitively* performed by a user. Thus, a way for the ESN to assist the user in her trust decisions is to simulate her assessment process. The challenge for the ESN is to provide an appropriate calculation model for estimating the confidence levels as well as the trust threshold. To make this explicit, in the following we only consider the simulated level of confidence and trust threshold, denoted as $\hat{c}_u$ and $\hat{t}_u$, respectively. As the confidence level is driven by the available evidence, the different types of evidence accessible to the ESN need to have assigned appropriate values that gradually increase the level of confidence.

We prescribe the trust threshold of a user to be initially determined by and automatically adjusted according to her trust declarations towards other users. The interaction that leads user u to declare trust towards user v will be used by the ESN as an estimate of the confidence level needed by u to trust other users she interacts with at a later stage. We provide details of these mechanisms in Section 4.2.

The advantage of this approach is avoiding the burden of elaborating a computable definition of trust, or, more precisely, enumerating and dealing with all conditions that cause persons to trust each other. An exhaustive list of such factors is very hard to compile, as many of them are very subjective.

Many factors influencing a person's trust lie outside an ESN, e.g., phone calls, work meetings, dinner parties, and thus cannot add up to whatever metrics the social network relies on to register the electronic interactions among its users. These considerations show why the user's autonomous decisions must play a fundamental role in any ESN-based solution to support a trust infrastructure.

4 Trust Management

The trust establishment process enables a user to build trust relations. As important as the establishment is the management of such relations, for two reasons. Firstly, social relations are very dynamic and so is the evidence that an ESN uses when suggesting to build a trust relation. This triggers the need to describe how changes in the communication frequency affect existing trust relations. Secondly, trust comes with a propagation effect: the confidence one

has in a very trusted friend of a very trusted friend is clearly higher than the confidence in a complete stranger with no attachments. Both aspects will be discussed in this section.

4.1 Dynamics of Relations

Relationships in real life change over time for various reasons such as people getting new jobs or hobbies. The change in a relationship can also be observed in an ESN. In particular, the evidence used for the calculation of $\hat{c}_u(v)$ can show the modifications in how close u and v are. Let us discuss how the trust between u and v evolves, while $\hat{c}_u(v)$ changes its value. As developed in Section 3.2, the level of confidence can increase as well as decrease. In the case of an increase, the user goes through a trust establishment process. Thus, whenever the condition $\hat{c}_u(v) \geq \hat{t}_u$ holds, the ESN proposes to u to enter in a trust relation with v.

We propose that a decrease of $\hat{c}_u$ below $\hat{t}_u$ should not trigger the ESN to suggest a change to an already established trust relation. This is because our trust definition is a statement that u once had the possibility to assess the correspondence of the values of $a_{E,u}$ with v's PII. Thus, we propose that only a change in metadata attached to a key might trigger the ESN to suggest a new trust assessment among users having an established trust relation but not enough interaction to fulfill $\hat{c}_u(v) \geq \hat{t}_u$.

4.2 Trust Levels

Some trust infrastructures rely on a more detailed model in which not only the fact that u trusts v ($u \twoheadrightarrow v$) is formalized, but a degree is also assigned to such relation. For example, the WoT includes two *trust levels*, namely, *marginal* and *full*, where the latter indicates stronger trust. Those trust levels are exploited in the trust propagation process. Let us show how to integrate such trust levels in our formalism.

We assume that the trust infrastructure provides a totally ordered list of $n+1$ increasing trust levels $R = \{r_0, \ldots, r_n\}$, corresponding to trust relationships that increase in strength. In the case of the WoT, we have $R_{WoT} = \{marginal, full\}$. We denote a trust relation of level r_j between u and v as $u \overset{r_j}{\twoheadrightarrow} v$. Let $T_{i,u} = \{v \mid u \overset{r_i}{\twoheadrightarrow} v\}$ be the set of users that u trusts with level r_i.

Let us remind that the ESN's estimate of u's level of confidence $\hat{c}_u(v)$ increases with the amount of interaction u has with v, and that when the condition $\hat{c}_u(v) \geq \hat{t}_u$ is reached, this may cause u to make her trust explicit to $u \twoheadrightarrow v$. Similarly, we define *trust level thresholds* that, once reached by the estimated confidence, can cause u to increase the level of a trust relation accordingly. In particular, these trust level thresholds are used by an ESN to make proposals for advancing trust relations to a higher level. For every $r_i \in R$ we define a threshold $\hat{t}_{i,u}$ and $\hat{t}_{0,u} = \hat{t}_u$, i.e., the threshold of the lowest trust level is equal to the user's general

trust threshold. As soon as condition $\hat{c}_u(v) \geq \hat{t}_{i,u}$ is reached, the ESN proposes to assign the trust level r_i to $u \twoheadrightarrow v$.

In accordance with the computational model for a user's general trust threshold, we also prescribe the values of the trust level thresholds to be defined by the user's behavior.

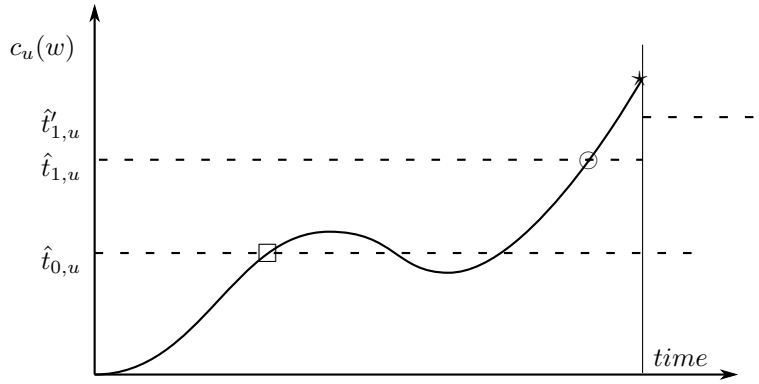

Figure 1: Level of confidence $c_u(w)$ between users u and w. We assume that u accepts the proposal of the ESN to add v to $T_{0,u}$ at $\square$. The proposal to add w to $T_{1,u}$ is rejected (at $\odot$) but u manually assigns w the next trust level (at $\star$). Thus, the ESN adapts $\hat{t}_{1,u}$ to $\hat{t}'_{1,u}$.

Let us illustrate how trust level thresholds are set in more detail. The basic idea is that the ESN establishes these thresholds on the basis of the user's past behavior, in terms of the interaction she previously needed before assigning another user a certain trust level. At the beginning, a user u has no trust relations nor any trust level set. In particular, we assume that all trust relations of u start at a very high level in order not to propose trust relations too early. Due to her interaction with v through the ESN, the level of confidence increases and at a certain instant the user declares that trust has been established: $u \overset{r_0}{\twoheadrightarrow} v$. The ESN records u's current $\hat{c}_u(v)$ and uses the value to make a first estimate of u's lowest trust level $\hat{t}_{0,u}$. Further interaction with v increases $\hat{c}_u(v)$ and at a certain instant u assigns the next trust level to v, resulting in $u \overset{r_1}{\twoheadrightarrow} v$. Again, the value of $\hat{c}_u(v)$ is used by the ESN as an estimate for $\hat{t}_{1,u}$.

The ESN uses these estimated trust level thresholds to assist u in finding an appropriate amount of interaction before entering trust relations. So, as soon as interaction with w makes u's level of confidence reach the previously established $\hat{t}_{0,u}$, the ESN will propose u to grant w trust level r_0. Should u accept this proposal, $\hat{t}_{0,u}$ is confirmed. In another case u might refuse the proposal of granting w trust level r_1, waiting for more trust-building interaction to take a decision. Thus, $\hat{t}_{1,u}$ needs some adjustment. If $\hat{c}_u(w)^*$ is the level of confidence at which u finally grants w trust level r_1, the new trust level threshold $\hat{t}'_{1,u}$ can be set as follows:

$\hat{t}'_{0,u} = \alpha \cdot \hat{t}_{0,u} + \beta \cdot \hat{c}_u(w)^*$, where α and β are parameters that weigh the contribution of the gap to the new threshold estimation (e.g., $\alpha = \beta = 0.5$, see Fig. 1).

A situation like the following, $\hat{t}'_{i,u} \geq \hat{t}_{j,u}$ with $r_i < r_j$, in which the gap is so wide that the new threshold for trust level r_i is above the threshold of level r_j, although $r_i < r_j$ needs to be handled with special care. All thresholds that are affected, that is, are overtaken by the newly estimated threshold $\hat{t}'_{i,u}$, must be adjusted. Let r_k be the first trust level whose threshold is above $\hat{t}'_{i,u}$. The estimation of trust level r_k should not be changed, as there has not been any significant clue on its inadequacy. All thresholds of levels between r_i and r_k must be then repositioned in the interval $\hat{t}_{k,u} - \hat{t}'_{i,u}$. The repositioning can be performed with a uniform distribution of the new thresholds in the interval, or by keeping the proportions between the relative positions that the old thresholds occupied with respect to $\hat{t}_{i,u}$ and $\hat{t}_{k,u}$. Should $\hat{t}'_{i,u}$ be above also the maximum trust level $\hat{t}_{n,u}$, then all thresholds between level r_i and r_n can be shifted accordingly. Analogous considerations hold for changes that lower the trust level thresholds.

One might object against the use of the trust levels established with a user v to perform estimations involving another user w: any motivation leading u's decisions with respect to v regards v, and v only. However, this position entails that user u should decide individually for the trust levels of all the users she connects to through the ESN, which is clearly an undesirable burden for most users, while the use of estimated trust level thresholds allows for the decision process to be supported by the ESN.

4.3 Trust Propagation

Trust propagation is important in scenarios where a user u wants to use another user x's public key which she has not signed. In fact, the trust propagation enables users to benefit not only from their direct trust relations but also from a larger network of people they might gain confidence in. In addition to the standard trust propagation mechanisms as the one used in the WoT or relevant improvements as described in [3], we propose to use the additional ESN information to create more confidence. For example, adding the information about the friendship and communication frequency between two people in the chain of the trust propagation may improve to a user's confidence.

5 Architecture and Implementation Guidelines

Let us now focus on the technical aspects of our approach. The description of the architecture relies on concepts related to standard WoT principles. Our aim is to shed light on the added value provided by the ESN.

5.1 Key Generation

The key generation process consists of (1) the generation of a private/public key pair, (2) the binding of PII metadata to the public key resulting in a certificate, and (3) the publication of the certificate to a publicly accessible repository.

Security considerations allow for the first step only to take place on a device trusted by u. Still, the ESN can enable u to initiate the key generation process on a user-trusted device via the ESN itself. The collection of the metadata as well as the publication of the certificate can be entirely performed by the ESN. For the generation of the certificate, however, a device holding the user's private key is needed. The ESN can facilitate this process with a specific request to the device. Finally, the ESN must add a reference to the certificate of u's profile.

5.2 Key Signing

When building trust relations as described in Section 3.4, users finally need to declare their trust. Given u willing to express $u \twoheadrightarrow v$, from a technical perspective, this entails that u signs v's public key together with the relevant attributes and uploads the resulting certificate to a publicly accessible repository. Thus, u confirms the binding of v's attributes as stated in the certificate and allows other people, who are not necessarily part of an ESN, to access this information.

The ESN can facilitate the key signing process by automatically comparing v's ESN attributes with the ones given in her self-signed certificate. In case of a mismatch the trust conditions are not met and u is warned. Signing v's certificate must rely on a device trusted by u as in the case of key generation, and the ESN can provide an analogous support.

5.3 Key Management

Key management turns out to be a complex task due to the following issues: (1) the availability of *all keys in* T_u to u on all devices that u uses even if only temporarily (e.g., a computer at an Internet cafe), (2) the availability of the *user's key pair*, especially her private key, from all her devices (devices not owned by u are excluded here), and (3) the *correct usage* of all keys, i.e., renewal of the own key, timely revocation, and refraining from the use of expired keys.

The ESN-based approach allows for optimization compared to the WoT approach in all those aspects. Firstly, u's *key ring* (the set of the public keys of the user u in T_u) can be downloaded transparently by the ESN whenever u connects with a new device. Thus, this problem boils down to a connectivity problem. Secondly, the portability of u's private key and thus of the possibility of executing transactions such as decryption or key signing is more critical. One solution is to let the user have the key on a portable device (e.g., a smart card). Another possibility is to use a group signature scheme where the user might register several devices which can all execute the transactions traditionally requiring the

private key. Note, that the current WoT implementation does not allow for group signature keys to be used. Thirdly, correct usage again boils down to a connectivity problem as whenever the user u is online, the ESN can update revocation lists or the expiration of u's key. Thus, we propose that the ESN provides a mechanism to allow for a timely replacement of a key coming close to its expiration. The renewal itself could consist of a proof of possession of the old private key and the generation of a new key pair.

The revocation of public keys in a PKI is a known issue. However, our proposal allows for improvement in that area by using short key life-cycles with automatic ESN-based renewal that involves the user's host.

6 Integration of multiple ESNs

So far, we focused on the benefits of a single ESN to the WoT. Let us now further potential of our approach by considering scenarios in which the WoT trust infrastructure is connected to multiple ESNs.

An issue arises in the task of establishing and managing trust when having several ESNs as opposed to one. Given that users u and v have profiles in a number of ESNs w_k where $k \in K = \{1, \ldots, \ell\}$, all ESNs w_k estimate $c_u(v)$ individually as $\hat{c}_{w_k,u}(v)$, whereas u's perception would be better modeled by $\sum_{k \in K} \hat{c}_{w_k,u}(v)$. A solution to this problem would be the communication of the respective confidence levels between the involved ESNs, either directly or via u's host. However, this seems unrealistic as ESNs currently do not allow for automatic information flow outside their own network.

Alternatively, ESNs could indirectly infer information by observing changes in the WoT. Let us assume that $\hat{c}_{w_m,u}(v) \geq \hat{t}_{w_m,u}$ holds for $m \in K$. ESN w_m then asks u whether v should be added to T_u. Should u accept the ESN's proposal, u would issue a certificate on v's public key. This change in the WoT can be noticed by all the other ESNs, which can infer that (1) u uses at least another mechanism (e.g., another ESN) to assess the trust in v, and (2) such mechanism has been used more frequently with respect to the interactions with user v. The second statement relies on the assumption that interactions affect the establishment of trust in all ESNs in a similar way. When a change in the WoT shows that user v has gained trust in some other network, an ESN can adjust its current estimated level of confidence by adding u's threshold, i.e., $\hat{c}_{w_k,u}(v) = \hat{c}_{w_k,u}(v) + \hat{t}_{w_k,u}$, for all $k \in K \backslash \{m\}$, to factor in interaction between u and v that is sufficient to insert v in T_u that has taken place in some other ESN. Expanding this example to a trust model with n levels is straightforward.

7 Related Work

We agree with Hogben [4], where he hints at the possibility to establish trust by means of the information provided by an ESN. Our work goes further though and

illustrates in more detail how such information can be exploited to actually build a trust relationship.

In [7], an alternative methodology to the WoT is discussed. A PKI with limited dimensions is required to establish trust in URIs. Trust is modeled after a transitive relation, and such transitivity is considered to be sufficient to ensure that the proposed mechanism is effective. Any aspects involving people, the essential component of a key signing party, is left out.

Bootstrapping an open ESN is discussed in [6], where trust aspects are not taken into account, but the focus is on the concept of security, interpreted here as the possibility to have protected personal information, as opposed to public and available to any member of the ESN.

Several works aim at exploiting the users' behavior in ESNs to establish trust relations. In [5], a policy-based approach is proposed, where access to private ESN data is granted on the basis of the ESNs by which the requester is linked to the data owner, and the interaction frequency on those channels. Although we share the authors' approach in considering dynamic aspects of users' behavior over time, we part from their effort in the following respect: such aspects are considered only in the process of writing access control policies to private data (e.g.: "only people who commented on my blog at least 10 times in the last 2 weeks can see the pictures"), while in our view, the behavior of users is continuously assessed to update the trust relationship with them.

In [1], a trust metric is proposed that not only takes third-party opinions into account, but also considers 'aging' as a factor that weakens a previously established link, unless it is refreshed with new interactions or mediated opinions. From the authors' perspective, the trust built by means of iterated interaction is to be interpreted as a measure of how reliable is the information provided by the ESN users. Our work, instead, is closer to the basic concepts of the WoT, as we are more focused on exploiting such exchanges to assess the link between an ESN entry and the person that created it.

Zhang et al. in [8] provide a more complex model of trust, where 'trust rating' to choose interaction partners is distinguished from a 'reliable factor' that assesses the believability of their acquaintances' assertions. Nevertheless, the evaluation of the link between the ESN entry and the real person is once again not considered. This work is strongly related to Goldbeck and Hendler's [2], where the authors aim at providing a trust rating system that allows for the creation and evaluation of links between people who are not directly connected in an ESN.

8 Conclusions and Future Work

We presented an approach that integrates widely-popular electronic social networking technology with the hitherto unsuccessful Web of Trust paradigm to overcome the main obstacle to WoT adoption: cumbersome establishment and management of trust in the legitimacy of key ownership. Our proposal

leverages available ESN profile and contact information, as well as interactions between users for establishing and managing a sufficient degree of trust for use in the WoT. Thereby, trust is established to a large extent by assessing ESN user behavior, requiring little to no extra effort on the side of the participating users. Moreover, our approach addresses key revocation and key renewal, two common key management problems of the traditional WoT. Overall, combining ESNs with the traditional WoT paradigm has the potential to provide security and trust solutions to non-technical users in a largely transparent manner.

To assess the practical feasibility of our ideas, it would be instructive to implement our model on top of existing ESN platforms. An implementation would also provide the basis for collecting empirical data to fine-tune the parameters of our model. A further area of future work concerns the study of privacy issues associated with the progressive integration of communication devices and disparate data sources as implied by our ideas.

References

[1] V. Carchiolo, A. Longheu, M. Malgeri, G. Mangioni, and V. Nicosia. An approach to trust based on social networks. In *Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE 2007)*, volume 4831 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2007.

[2] J. Goldbeck and J. Hendler. Inferring binary trust relationships in web-based social networks. *ACM Transactions on Internet Technology (TOIT)*, 6(4):497–529, 2006.

[3] R. Haenni and J. Jonczy. A new approach to PGP's web of trust. In *EEMA'07, European e-Identity Conference*, Paris, France, 2007.

[4] G. Hogben. Security issues in the future of social networking. In *W3C Workshop on the Future of Social Networking*, 2009.

[5] A. Passant, P. Karger, M. Hausenblas, D. Olmedilla, A. Polleres, and S. Decker. Enabling trust and privacy on the social web. In *W3C Workshop on the Future of Social Networking*, 2009.

[6] H. Story. Building secure, open and distributed social network applications. Technical report, Sun Microsystems, 2008. `http://blogs.sun.com/bblfish/entry/building_secure_and_distributed_social`.

[7] H. Story. FOAF+SSL: Creating a web of trust without key signing parties. Technical report, Sun Microsystems, 2009. `http://blogs.sun.com/bblfish/entry/more_on_authorization_in_foaf`.

[8] Y. Zhang, H. Chen, and Z. Wu. A social network-based trust model for the semantic web. In *Proceedings of the 3rd International Conference on Autonomic and Trusted Computing (ATC 2006)*, volume 4158 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2006.

[9] Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Electrical Engineering (ESAT)
COmputer Security and Industrial Cryptography (COSIC)
Kasteelpark Arenberg 10 — 2446, 3001 Heverlee, Belgium