KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Engineering
Department of Computer Science

Minimizing Information Disclosure in Authentication Transactions with Attribute-Based Credentials

Franz-Stefan PREISS

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

September 2012

Minimizing Information Disclosure in Authentication Transactions with Attribute-Based Credentials

Franz-Stefan PREISS

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

September 2012

D/2012/7515/108
ISBN 978-94-6018-574-8

This should be doable in half an hour.

Jan Camenisch

Acknowledgements

THE exciting endeavor of obtaining a PhD can only be accomplished with the support of others. In the following, I want to acknowledge the tremendous help that I received from colleagues, friends, and family. First of all, I want to thank my promotor, Prof. Bart De Decker, for supervising me as student under the extraordinary circumstances of me residing abroad, for his scientific input and feedback, for his help in dealing with all the administrative and bureaucratic issues, and for his motivating words of encouragement in the very situations when they were needed the most.

I am very grateful to my supervisor at IBM Research, Dr. Jan Camenisch, for animating me to pursue an industrial PhD in the first place, for the excellent guidance of my research, and for teaching me that diplomatic pragmatism with a portion of honesty is often the best (and fastest) path to success. I am also very grateful to Dr. Gregory Neven who often acted as informal supervisor giving me outstanding advice in scientific and many other matters, for being such an excellent teacher who is never short of illustrative and convincing examples, and for the countless interesting, insightful, and sometimes hilarious discussions.

I want to thank Prof. Ronald Leenes, Prof. Frank Piessens, and Prof. Bart Preneel for being members of my examination committee and for their helpful feedback on the initial version of this thesis. Thanks go also to Prof. Martine Wevers for acting as chair of my examination committee.

Further, I am indebted to all my coauthors. In particular, I want to thank Sebastian Mödersheim, Patrik Bichsel (yes, Patrik, we both made it), Dieter Sommer, and Mario Verdicchio, for making every paper an intellectual adventure, and for making the last hours before paper submission deadlines so amusing and truly unforgettable, over and over again.

This work was performed at IBM Research – Zurich, Switzerland, in cooperation with KU Leuven. I am grateful to my employer, IBM Research, and to my managers Phil Janson, Andreas Wespi, Michael Osborne, and Peter Buhler who made this industrial PhD possible by trusting in the delivery of this very document. Many thanks also go to Christian Cachin whose university lecture was the entry point to an IBM internship that led to my position as doctoral student.

I am very thankful to my past and present colleagues including my fellow doctoral students at IBM. Their openness and brilliance made working there a real pleasure. I want to thank Anja Lehmann and Günter Karjoth for their input to my university lecture on privacy and data-minimizing authentication. I further thank Samuel Burri for going trough the lectures and exercises at ETH Zurich together with me. Lydia Chen and Dieter Sommer deserve thanks for proofreading my initial thesis draft. Special thanks also go to my colleague Jorn Lapon from KU Leuven, for introducing me to the research group at a summer school, and for making my stays in Leuven so enjoyable.

Moreover, I want to express my gratitude to my family, for its unconditional support, and for the steady supply of delicious traditional Austrian food and excellent wine. Thank you for tolerating that I was often busy and unavailable due to work in the past years. Finally, I want to thank all my friends who have contributed to a wonderful time as doctoral student.

Franz-Stefan Preiss
Zurich, September 2012

Abstract

WITH the rise of information and communication technologies, the need to authenticate individuals to authorize their access to online services or to hold them accountable for their actions has induced the development of a wide variety of authentication systems. Although for determining sole authorization it is mostly sufficient to verify an individual's unlinkable non-identifying properties, virtually all of these systems involve the disclosure of personally identifiable information. This raises numerous security and privacy issues because an uncontrolled dissemination of these data makes individuals vulnerable to identity theft, financial fraud, profiling, monitoring, discredit, or embarrassment. These issues are greatly amplified by technologies that ease data collection, aggregation, analysis, and distribution, by legislation that stipulates the retention of communication data, and by increasingly frequent data breaches where vast amounts of (personal) data records are compromised.

Although there exist cryptographic techniques—namely *anonymous credentials*—that allow individuals to authenticate in a secure and privacy-preserving manner without having to disclose any personal or identifying information, there are no authentication systems that utilize these techniques. While the reasons for this are manifold, there are two major technical inhibitors. On the one hand, the available implementations are very complex and only usable with cryptographic expert knowledge, and, on the other hand, the cryptographic mechanisms alone do not suffice for building an authentication system.

In this work, we overcome these inhibitors and present a functional authentication system on the basis of anonymous credentials that is usable without expert knowledge. With our system, service providers can formulate authentication requirements in terms of the minimal properties that users' certified attributes must have, and users can prove that their attributes fulfill these properties without disclosing their values. In situations where accountability is required, users can disclose personally identifying information such that it is only accessible if they misbehave or cause damage—which allows honest users to remain unidentifiable.

The main building block of our system is a language framework with formal semantics for expressing the service providers' minimal authentication requirements as well as users' cryptographically backed claims in terms of attribute-based credentials. The framework abstracts away from cryptographic details and focuses solely on easily intelligible concepts. We also provide algorithms for transforming claims expressed in our language into the complex input specifications of the cryptographic implementations—which significantly eases their use for application developers—and for verifying claims with respect to a given policy. On the basis of these results, we develop a full-fledged prototype implementation to prove the concept and its efficiency: we show that our algorithms entail negligible computational overhead with respect to the time needed to generate and verify the cryptographic evidence that supports users' claims.

Our system allows for reducing the information that is disclosed in authentication transactions to the necessary minimum and thereby mitigates the aforementioned issues of excessive data release. Its use is advantageous for both users and service providers in that the former benefit from privacy preservation and the latter from reducing the risks associated with holding large sets of sensitive personal information.

Samenvatting

De opkomst van informatie- en communicatietechnologieën en de daarmee gepaard gaande noodzaak individuen te authenticeren en hen verantwoordelijk te houden voor hun acties, heeft de ontwikkeling teweeggebracht van verscheidene authenticatiesystemen. Alhoewel in principe niet-traceerbare en niet-identificerende eigenschappen meestal volstaan voor het nemen van autorisatiebeslissingen, onthullen vrijwel alle authenticatiesystemen persoonlijk identificeerbare informatie. Dit brengt talrijke veiligheids- en privacyrisico's met zich mee, gezien een ongecontroleerde verspreiding van deze gegevens individuen blootstelt aan identiteitsdiefstal, financiële fraude, profilering, afluisteren, of het in diskrediet brengen van het slachtoffer. Deze problemen worden nog versterkt door technologieën die het verzamelen, analyseren en verdelen van gegevens vergemakkelijken, door wetgeving die de bewaring van bepaalde gegevens verplicht, en door de steeds vaker voorkomende datalekken waarbij massieve hoeveelheden persoonsgegevens worden gecompromitteerd.

Alhoewel er cryptografische technieken bestaan—met name, *anonieme credentials*—die individuen in staat stellen zich op een privacy-vriendelijke manier te authenticeren zonder daarbij persoonlijke of identificerende informatie vrij te geven, bestaan er geen authenticatiesystemen die deze technieken gebruiken. Dit heeft verschillende oorzaken, maar er zijn twee belangrijke technische hinderpalen. Ten eerste zijn de beschikbare implementaties zeer complex en enkel bruikbaar mits cryptografische vakkennis. Ten tweede volstaan de cryptografische mechanismes niet om een volledig authenticatiesysteem te bouwen.

In deze thesis overwinnen we deze obstakels en stellen een functioneel authenticatiesysteem voor gebaseerd op anonieme credentials dat bruikbaar is zonder cryptografische vakkennis. Dankzij ons systeem kunnen dienstverleners authenticatievereisten formuleren als de minimale eigenschappen waaraan de gecertificeerde attributen van de gebruikers moeten voldoen. Gebruikers kunnen aantonen dat hun attributen aan deze vereisten voldoen, zonder de waarde van die attributen te onthullen. In situaties waarbij toerekenbaarheid een vereiste is, kunnen gebruikers persoonlijk identificerende informatie op een zodanige manier versleutelen dat die enkel toegankelijk wordt zodra ze zich misdragen of schade veroorzaken—terwijl eerlijke gebruikers anoniem blijven.

De belangrijkste bouwsteen van ons systeem is een taalraamwerk met een formele semantiek waarin dienstverleners de minimale authenticatievereisten kunnen uitdrukken en waarin gebruikers hun cryptografisch ondersteunde beweringen kunnen uitdrukken op basis van attribuut-gebaseerde credentials. Ons raamwerk bevat bovendien algoritmes om beweringen uitgedrukt in deze taal om te zetten in invoerspecificaties voor de cryptografische implementatie—hetgeen het gebruik ervan door applicatie-ontwikkelaars aanzienlijk vergemakkelijkt—en om beweringen te verifiëren ten opzichte van een gegeven beleid. Op basis van deze resultaten ontwikkelen we een volwaardig prototype dat de haalbaarheid en efficiëntie illustreert: we tonen aan dat onze algoritmes een verwaarloosbare computationele ballast met zich meebrengen ten opzichte van de tijd om het cryptografisch bewijsmateriaal te controleren dat de beweringen van de gebruikers ondersteunt.

Ons systeem laat toe de onthulde informatie in authenticatietransacties tot een minimum te herleiden en beperkt daarmee de eerder vermelde risico's van buitensporige vrijgave van gegevens. Dit is voordelig voor zowel gebruikers als dienstverleners gezien de eersten profiteren van een behouden privacy en de laatsten van verminderde risico's verbonden aan het opslaan van grote hoeveelheden gevoelige persoonlijke informatie.

Contents

List of Figures

List of Tables

List of Abbreviations

ABC	Attribute-Based Credential
AJAX	Asynchronous JavaScript and XML
CARL	Credential-Based Authentication Requirements Language
CBAFx	Credential-Based Authentication Framework
CRL	Certificate Revocation List
DNF	Disjunctive Normal Form
ESN	Electronic Social Network
GUI	Graphical User Interface
ICT	Information and Communication Technology
IdP	Identity Provider
OASIS	Organization for the Advancement of Structured Information Standards
PII	Personally Identifiable Information
PIN	Personal Identification Number
PKI	Public-Key Infrastructure
RAP	Rich Ajax Platform
SAML	Security Assertion Markup Language
SPI	Sensitive Personal Information
TTP	Trusted Third Party
UI	User Interface
URL	Uniform Resource Locator
XACML	eXtensible Access Control Markup Language

Part I

Data-Minimizing Authentication

Chapter 1

Introduction

THE Internet is the most empowering tool that ever existed in terms of information, communication, commerce, social interaction, education and entertainment. It enables us to perform more and more activities electronically: online shopping, travel booking, paying bills, electronic banking, and tax declaration are just a few of the countless examples that were only recently performed without the use of computing infrastructure and mobile devices. The need to authenticate individuals to authorize their accesses to the respective Internet applications or to hold them accountable for their actions gave rise to the development of increasingly advanced authentication systems. These systems range from simple passwords to smart cards, to biometrics, to public-key infrastructures (PKIs) with digital signatures, to online authentication mechanisms, and to combinations thereof [60].

Although there exists a wide variety authentication technologies, virtually all of them involve the disclosure of personal information—or even personally identifiable information (PII)—which raises numerous security and privacy issues. Amongst others, these data can be used for social engineering, identity theft, or financial fraud—or their undesirable disclosure can lead to discredit or embarrassment. The issues are amplified by the increase of technologies that facilitate data collection, storage, aggregation, analysis, and distribution [60] as well as by legislation that stipulates the collection and retention of communication data. Consider, for example, the surveillance cameras in public and private areas, the "loyalty" programs collecting data about customers' buying behavior, the content shared in electronic social networks (ESNs), the communication data of Internet service providers, and facial recognition technology used in ESNs that are trained every time a person is *tagged*. Moreover, consider that most of the online services we use are only seemingly "free". In fact, we are paying them with personal and behavior data disclosed through the use of the services; these data are then monetized by advertisements based on profiling of the same, effectively making our personal data the new currency of the digital world. Another important factor that worsens the security and privacy issues are the increasingly frequent data breaches where vast amounts of (personal) data records are stolen by hackers. The latter clearly illustrate the need for a paradigm shift in the way how enterprises and governments treat both cybersecurity as well as personal privacy [19].

In search of a basis for such shift, one just has to recall what authentication is, why it is done, and what the motives for collecting personal data during authentication are. Authentication is the process of establishing confidence in the truth of some claim. In general, user authentication is not an end in itself: users are authenticated to authorize their requests to perform actions or to hold them accountable for the actions they perform. Accountability is the ability to hold an individual responsible for a past action and thus requires unambiguous identification [60]. For solely determining authorization, in many cases the verification of unlinkable non-identifying properties of the users suffices. However, current authentication systems collect identifying information as if accountability is required. For example, to determine a user's eligibility to enter a teenage chat room, neither her name nor her date of birth nor any other personal data are needed: being convinced of the very fact that the user indeed is between 13 and 19 years old is sufficient. Thus, the conflict between privacy and the need to involve personal data in authentication is only a seeming one.

The good news is, technological means allowing users to prove to service providers that they are, for example, teenagers—without disclosing any of their personal data—do already exist. More precisely, by means of sophisticated cryptographic techniques [37, 15, 28], users can prove properties about their attributes without disclosing them—that is, they can minimize the disclosed information necessary for being granted access to the services they requested. In situations where authentication is performed to hold users accountable, these techniques can even solve the issues that originate from the need to collect personally identifiable information. In particular, this information can be collected in a way such that it is only accessible if a user misbehaves or causes damage—which allows honest users to remain unidentifiable.

Despite the existence of general mechanisms for performing authentication in a privacy-preserving manner as well as of freely available implementations of those mechanisms that are efficient for practical use cases, they have not found real-world adoption in a usable authentication system yet. While the reasons for this are manifold—as we believe—the following two are the primary ones that inhibit the adoption from a technical point of view (see Chapter 5 for non-technical inhibitors). On the one hand, the available implementations are usable only for experts in cryptography: although the implementations exist since years, they are still virtually impossible to use for application developers because knowledge about the cryptographic specifics is necessary. On the other hand, the cryptographic mechanisms and their implementations alone are not sufficient for building a working data-minimizing authentication system: additional building blocks such as a policy language to express the minimal authentication requirements as well as corresponding algorithms to automatically generate and verify cryptographic proofs on the basis of the policies are crucial.

In this work, we provide all the missing building blocks and show how a functional and usable data-minimizing authentication system can be implemented on the basis of existing cryptographic mechanisms. In particular, we provide a language framework for expressing authentication requirements that abstracts away from cryptographic details and rather focuses on intelligible concepts. We further provide the necessary algorithms for transforming statements in our language to the input languages of the cryptographic implementations; as those input languages drive the execution of the cryptographic protocols, we effectively make the implementations usable for application developers by means of the high-level concepts. On top of that, we provide the components that are necessary for verifying cryptographic proofs with respect to authentication policies.

In the following two sections, we describe the issues of authentication as it is performed today for most online services, and why our data-minimizing system does not suffer from these issues.

1.1 Authentication Today

There exists a multitude of different authentication mechanisms used for a wide variety of activities, which range from physical access to buildings (for example, employer premises) to electronic commerce to the use of smartphone applications. The activity where authentication is most prominently noticed in everyday life is the use of online services such as online stores or music streaming platforms. For the use of virtually any such online service, prior creation of a user account is either a strict requirement or strongly encouraged by the service provider. To create a user account, users must choose a username/password pair and generally have no other choice than revealing an extensive amount of personal information such as their name, date of birth, sex, e-mail address, mailing address, and phone number. After account creation, users authenticate themselves with the previously established username and password to get access to the service. However, this authentication technique—involving passwords and personal information disclosure—has a number of drawbacks for both users and service providers concerning security, privacy, trust, and convenience.

1.1.1 Issues for Users

First of all, there is the imminent danger that the accumulated data sets are accidentally leaked or actively stolen by hackers. In fact, the list of data breaches is tremendously long and steadily growing. There is hardly a week going by where some major enterprise or government agency is not breached for fun or profit [53, 50]. For example, in April and May 2011, hacker attacks on the Sony PlayStation Network exposed more than 100 million data sets [50] including names, dates of birth, credit card data, and mothers' maiden names. While *WikiLeaks* and *Anonymous* were once known only to a subset of cyber geeks, those are household names now that appear frequently in news reports [19]. According to the yearly Verizon data breach investigations reports [86], only between 2008 and 2011 at least 683 million data records containing personal information were compromised worldwide, from which 174 million happened in 2011 throughout 855 reported data breaches. As these staggering and disturbing figures reflect only the known breaches, the dark figure, which includes the number of unreported cases, is likely to be much higher. This is because in the United States—where a majority of the worldwide data is stored and processed—many states do not have a centralized data loss incident reporting legislation, or even no data loss reporting legislation at all [50].

Security

The nature of passwords, the disclosure of personal information during user account creation, and the resulting loss of massive amounts of personal data by service providers have serious security consequences. In particular, passwords are inherently insecure because of their vulnerability to guessing and theft, as they do not change without human intervention, and because they can be easily shared—either intentionally or by mistake [60]. Because people have trouble remembering different passwords (and usernames), many users improvidently reuse the same username/password combination across different service providers. This erodes the security of passwords, however, as malicious service providers can easily impersonate users at other service providers. Users who are more considerate and choose different usernames and passwords for different service providers are prone to compromising their own passwords: almost every such user was presumably already in the situation of trying different combinations with a single service provider, being unsure about which is the right one. Such behavior inherently compromises all wrongly tried combinations, however.

Further, the involuntary disclosure of sensitive personal information to service providers during account creation makes users potentially vulnerable to impersonation, identity theft, blackmailing, financial fraud, and social engineering. This is not only due to possibly dishonest service providers, but also to the previously described data theft. For example, hackers can use the obtained usernames, passwords, and personal information to obtain even more information such as online banking login data through spear phishing and to empty their victims bank accounts. Such spear phishing, which is a form of social engineering that is tailored to the target, is heavily facilitated by the availability of personal information about the victim. Credulous users are blindsided by the accurate and comprehensive knowledge about them—that is, information that in their belief only legitimate parties can have—and tricked into disclosing even more information.

Convenience

Users who are considerate with respect to the choice of their usernames and passwords quickly face the difficulty to remember all the different combinations. Additionally, passwords are supposed to be long involving special characters—which makes them cumbersome and even more difficult to remember—and should or even must be changed regularly. The tedious and inconvenient nature of user account creation and the associated passwords also comes with a lock-in effect for users: users may stay with one service provider (an online shop, for example) rather than setting up a fresh user account with a different provider of the same service. To mitigate these issues, there are service providers (for example, OpenID [41] and newly also Facebook) whose service is to exchange the information in their user accounts with other service providers—which employment effectively makes them to online identity providers. Although the advantage for a user is that only the account with the identity provider must be maintained, such a centralized authentication model raises severe privacy concerns as the identity provider has notice of each of a user's authentication transactions by virtue of its involvement in the transaction.

Privacy

In addition to the described security issues, authentication as it is performed today as well as the described data breaches raise considerable concerns in terms of privacy. Once personal data are released—for example, during user account creation—they can no longer be controlled: they can consequently be distributed to others and a deletion after the fact is practically impossible because computers do not forget. Different data records or pieces of disclosed information can be linked and user profiles can be created [25]. Employing the same username across different service providers abets this problem because a collusion of service providers allows them to link the individual user profiles. The privacy risks associated with the linkage and profiling are embarrassment and discredit—for example, caused by the unwanted disclosure of particularly sensitive personal information, including facts about location, interests, age, sexual orientation, religion, medical history, health concerns, or political affiliation. Moreover, the built profiles are used for targeted advertising. Although this form of advertising can be very useful for consumers and is nothing to object to in general, there are also certain associated dangers. While traders do not care about actual names or exact physical addresses, they are interested in who we are and a way to reach us. For a young, sporty, healthy person, it is unproblematic when advertisers can show targeted ads on her screen, but somebody who is unemployed, in debt, and sick is a target for predatory loans and fraudulent financial or personal advice [62].

The linkage and profiling of information is facilitated by the (not always lawful) availability of additional user information that is intentionally, unintentionally, knowingly, or unknowingly disclosed and collected. For example, consider the data traces that we all leave in everyday life: smartphone traffic, email content, shopping behavior with respect to credit and loyalty cards, web browsing behavior as determined by browser scripts, closed-circuit television footage, and travelling habits by virtue of flight tickets and hotel bills are just a few examples. The recent developments in EU legislation boost these issues because the controversial Data Retention Directive (2006/24/EC) prescribes the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks. Member states have to store citizens' traffic data (for fixed and mobile telephony, Internet access, Internet email and Internet telephony) between 6 and 24 months where police and security agencies are able to request access to details such as IP address and time of use of every email, phone call and text message sent or received.

Other factors that ease data linking are, for example, the price development of data storage and the advances of data mining techniques. As storage is becoming cheaper, nowadays data seem to be stored by default. The wireless router traffic picked up by Google Street View cars as well as the location data unknowingly persisted by iPhones are examples of this. To ease privacy concerns that are related to storage of (sensitive) personal information, corresponding data sets are often anonymized. However, relating anonymized records from different sources can easily lead to de-anonymization of the concerned individual [69, 90]. Sophisticated and highly efficient data mining techniques perform not just trend detection but also trend prediction (for example, flu pandemics). The potential application of such techniques to mortgage defaults or criminal behavior is concerning from a privacy point of view.

Another privacy issue associated with the collection of data is commercial price discrimination. Companies collect data on consumer behavior and use information about past purchases to tailor how they treat them in the future. For example, the personal and behavioural information can be used to extract the maximum price possible for somebody to pay for a service. The consumers are mostly not aware that companies can trade and mine this information and, in particular, alter prices and offers based on these data [62].

In discourse about privacy, a popular retort is the "nothing to hide" argument: "If you have nothing to hide, then what do you have to fear?" [80] or "What is it that you are doing that you need to hide?" This argument essentially balances security concerns against privacy interests—making the latter appear trivial and thus giving security a clear victory [80]. As everybody probably has something to hide from somebody, this argument can easily be refuted by calling attention to matters that the people do want to hide—at least to a certain audience—such as discreditable or embarrassing details about their private life (for example, health issues, sex life—"So do you have curtains at home?"). The person bringing the nothing to hide argument is likely to realize that neither of these details concerns bad nor illegal behaviour, yet it is simply not anyone else's business. While privacy can be motivated with philosophical arguments, after all, privacy is a fundamental right according to legislation ("I don't need to justify my position.

You need to justify yours. Come back with a warrant." [80]). For example, the European Data Protection Directive (95/46/EC) requires all EU countries to adopt similar comprehensive privacy laws that recognize privacy as fundamental human right. It seeks to protect the individual about whom data are processed through a combination of rights for data subjects and obligations on data processors. As privacy is crucial for a functioning democracy (voting, for instance), we should be able to exercise this right also in digital contexts.

Trust

On the basis of all the security and privacy issues related to the use of the Internet, the general trust in it is decreasing. The fact that users are required during account creation to disclose personal information that is often even of no direct relevance for the service at hand, makes the users—for whom the motivation of the data collection is not apparent—suspicious. Users usually neither know nor can control what happens to their data, how they are used, and with whom they are shared, but still they are forced to provide them. This, and the frequent reports about the loss or theft of their data, leads to general mistrust of the Internet as information and communication medium as well as a marketplace. Maglena Kuneva—former European Commissioner for Consumer Protection—compares the use of the Internet under these conditions to drinking water while thinking it might be slightly toxic: despite knowing that it can harm us, we have no choice but drinking it. But we will drink just enough and no more than necessary, and the water market will not thrive [62]. This illustrates, that the Internet cannot unfold its full potential as marketplace and information medium as long as privacy concerns inhibit this.

1.1.2 Issues for Service Providers

When the security and privacy issues of authentication (and its implementation) are discussed, mostly only the ones concerning end users are brought forward and the ones concerning service providers are left out. In particular, it is often not mentioned that there are also risks and side effects for service providers with respect to collection of personal data. These risks include the costs associated with data breaches, for example.

Consequences of Data Breaches

The previously mentioned data breaches do not only impact users whose data is compromised, but also have implications for the entities who are losing the data. In particular, data loss may not only have serious legal and financial consequences (for example, fines, lawsuits, and stock price drop) but also result in damage to the company's reputation (for example, bad press and loss of customer's trust), loss of customers and thus business and profit loss. With the rise of cloud-based infrastructures, which allow for easy and cost-efficient outsourcing of data storage to external parties, there is also the danger that the assigned storage provider is breached whereas the service provider will have to take the blame by his customers.

The costs that were reportedly associated with last year's Sony incidents are 170 million US dollars [2]. In 2010, the costs arising from malicious or criminal attacks were on average 318 dollars per customer record that was compromised, which number grew by 43 percent with respect to 2009 [74]. According to the Symantec Corporation, the total annual losses resulting from cybercrime are valued at 388 billion dollars—by contrast, illicit sales of marijuana, heroin and cocaine are an estimated 288 billion dollar industry [81].

Data Quality and Quantity

Further, service providers struggle with quality issues of account data as users can easily provide incorrect—that is, erroneous, deliberately falsified, or fraudulent—data during account registration. Erroneous data entries provided by customers cause more than 25% of the US data quality problems [82]. The reasons for users to provide falsified data are twofold. On the one hand, users are frustrated with the tedious registration procedures where they have to enter their data over and over again. One the other hand, some users simply strive for better privacy and thus enter adulterated registration information. Their ability for doing so comes from the fact that data entered into web forms are mostly unverified. In the United Kingdom, almost 2% of accepted online orders later turn out to be fraudulent [44].

According to Gartner, data growth is the biggest data center challenge faced by enterprises [54]. While the cost of storage is decreasing, this challenge arises from the increasing data management and protection expenses, compliance costs, and legal risks associated with the continuously growing amount of data. To reduce costs and risk, enterprises start to look into performing defensible data deletion—where their attention is also turned to the disposal of privacy-related data such as sensitive personal information.

1.2 Authentication of Tomorrow

In order to mitigate the security and privacy issues of authentication as it is performed today, a new kind of authentication system is needed that allows for reducing the amount of (personal) information that is disclosed during authentication transactions to the necessary minimum. The less information we share about us, the better we are protected—because less information can be linked, lost, or used for identity theft. According to Kent et al. [60], the implications of authentication to privacy do not necessarily equate to violation of privacy: it rather depends on how a particular authentication system is designed and implemented. In this work, we present a secure authentication system that was designed with privacy in mind—that is, a system that has the potential to significantly improve users' security and privacy in authentication transactions and at the same time assures service providers of fulfillment of their service, legal, and compliance requirements.

One of the distinguishing features of this system is that it enables users to just *prove* that they fulfill a service provider's authentication requirements without having to disclose *any* of personal information. By doing so, the service provider learns nothing except for *the fact* that the user fulfills the given requirements. For example, in an online poll where only adult citizens with particular postal codes are eligible to vote, a user could prove that she is of age and resident of the designated area without disclosing her age, date of birth, postal code, home address, or any other personal information—which is clearly in her privacy interest. Assuming, as further example, that a company offers discounted health insurance to customers whose body mass index (BMI)—which is calculated as a person's weight divided by the square of its height—is between 18 and 25, a user could prove this fact without disclosing her exact weight, height, or BMI.

Another feature of the system is that users are enabled to act under a pseudonym while their real identities remain hidden to the service providers. To limit the risk of service abuse involved for the service providers, pseudonymous users can be identified under—and only under—well-defined conditions that are agreed upon beforehand (that is, before the authentication takes place and before any identifying information is disclosed by the user) where possibly multiple external trusted authorities are assigned to judge over the applicability of the conditions. This enables service providers to hold misbehaving users accountable for their actions, while honest users can act without being identifiable.

The seemingly contradictory nature of these features—that is, proving users' attribute properties without disclosing the attributes' values as well as acting anonymous yet accountable—are enabled by advanced cryptographic mechanisms. In particular, they are made possible by *anonymous credentials* [37, 15, 28], a privacy-preserving form of digital attribute-based certificates that allow for exercising absolute control over the data released during an authentication transaction. The idea is that identity providers certify users' attributes (date of birth, for example), and service providers rely on the trustworthiness of the identity providers for authenticating the attributes and their properties. Various entities such as governments (for the date of birth), registration offices (for the home address), medical institutions (for height and weight), or designated companies may act as identity providers vouching for the attributes they certified. After issuance, the credentials are kept and managed by the users themselves, who can selectively disclose attributes from their credentials (to the service provider directly or to trusted third parties to establish accountability) or derive unlinkable attribute predicate proofs—all that, without involvement of the identity provider.

In the following paragraphs, we discuss why an authentication system based on anonymous credentials does not suffer from the issues described in the previous section.

1.2.1 Benefits for Users

In our proposed system, users mainly provide unlinkable cryptographic proofs that the service provider's authentication requirements are fulfilled. By using such proofs as means of authentication, there are no more security issues related to usernames and passwords. In particular, unlike passwords, the cryptographic proofs cannot be guessed and they are worthless when being stolen from a service provider. Users are still responsible for protecting their credential repository, which may involve the use of a personal identification number (PIN) or password, but this password never leaves the user's device. Service providers, eavesdroppers, or hackers cannot replay the proofs, thus, they cannot impersonate users—which is possible with with username/password combinations. Because mainly unlinkable proofs are used to authenticate and personal data is only disclosed in situations where this is strictly necessary, the possibility to link user profiles as well as the likelihood of being victim of identity theft or social engineering decreases.

With the abandoning of passwords as means of authentication, users no longer face the hassle of having to come up with, remembering, and changing username/-password combinations for different service providers. Even in situations where personal data disclosure is inevitable, the data do not have to be entered manually but may be taken automatically from the certified credentials.

As the cryptographic attribute property proofs that users derive do not contain or reveal any personal data and are thus worthless for identity thiefs, the motivation for hackers to attack enterprise systems and databases practically disappears: what does not exist cannot be stolen. The proofs are derived locally on user-controlled devices without involvement of the credential issuer. Issuers therefore do not have notice of the authentication transactions that users perform with their credentials, which is different from authentication with online identity providers who can monitor the user's authentication behavior. Further, different proofs created for an authentication transaction by the same user are both unlinkable to other authentication transactions and untraceable to the issuance transaction of the credentials underlying the proof, even in the face of collusion of service providers and credential issuers. Profiling of users by means of attribute property proofs is hence impossible, which is beneficial for the users' privacy. However, users still have the possibility to intentionally establish linkability of their actions—that is, re-authenticate—without being identifiable. To this end, a user generates a cryptographic pseudonym from a user secret, provides a service provider with the pseudonym and proves—whenever she wants to be recognized as the same user—that she knows the corresponding secret without revealing it.

Moreover, the trust in the Internet in general is likely to rise as soon as users get back control over their personal data, that is, as soon as they do not have to disclose personal data any more to use their preferred online services, or as soon as they just have to disclose the amount that is reasonable in the given context. A resulting decrease in data breaches due to less circulating personal data as well as less privacy concerns in general will presumably aid restoring this confidence.

1.2.2 Benefits for Service Providers

When using anonymous credentials as means of authentication, the resulting information stored in enterprise databases (that is, cryptographic eligibility proofs and pseudonyms) becomes significantly less worthwhile for hackers to be stolen because neither passwords nor exact personal data values are stored. Hackers accordingly have less reason to attack service providers for data that is usable for identity theft and the data breaches will diminish. This mitigates the negative consequences associated with these breaches. The underlying principle for service providers is: you cannot loose what you do not have. Thus, performing authentication in a data-minimizing way is a means of risk avoidance for service providers.

With current authentication mechanisms, companies who generally have no interest in collecting personal data (for example, because their business model is not based on online advertisements) may still be forced to do so to hold users accountable in case of service abuse. For instance, an online game platform provider whose business model is to sell games and not advertisements, must collect personal data to fulfill youth protection requirements (consider games that must not be played by children younger than 16 years), so as to assure audit of legal compliance. Another example is a car rental company that collects renters' drivers license and insurance data only to get reimbursed in case of an accident. With our authentication system, service providers no longer have to store data that they need for user accountability reasons themselves, but can outsource this burden to (possibly multiple) designated trusted third parties who are specialized in data protection.

For the data that must be released to service providers, the data quality will increase because it originates from credentials that are vetted by trusted identity providers. Finally, the less personal data service providers have to store, the less costs for data management and protection (compliance with privacy laws, for instance) incur.

1.3 Outline

This doctoral thesis is based on publications and consists of two parts. Before we present reformatted reproductions of a selection of our publications in Part II starting from page 88, we provide a general introduction to data-minimizing authentication systems in Part I.

Part I consists of four chapters. In Chapter 1, we motivated our work by reviewing how authentication to online services is performed today and by pointing out what the associated drawbacks are. We also described how authentication can be performed in a data-minimizing manner with anonymous credentials and what the benefits of this approach are.

In Chapter 2, we give a comprehensive overview of anonymous credential systems, which are the basis of the data-minimizing authentication systems that we propose in this work. This overview first outlines attribute-based credentials as well as pseudonyms and then elaborates on credential presentation, issuance and revocation.

In Chapter 3, we describe data-minimizing authentication systems—which are the centerpiece of our work. First, we introduce data-minimizing authentication in general. Next, we depict the sequence of data-minimizing authentication transactions and detail each of the components involved. Afterwards, we present our open-source authentication framework as well as a prototype implementation, which is based on this framework, including time measurement results.

In Chapter 4, we provide an extensive summary of our scientific contributions, which we divide into five contribution groups. For each group, we describe which of our publications account for this contribution and how these publications build on each other.

In Chapter 5, we draw conclusions and discuss open problems.

Chapter 2

Anonymous Credential Systems

BEFORE we can describe data-minimizing authentication systems which are the main contribution of this thesis, it is essential to convey the basics of anonymous credentials because they are the foundation of such authentication systems. Accordingly, in this chapter, we provide a comprehensive overview of the features and capabilities of anonymous credential systems.

A credential system allows users to obtain digital attribute-based credentials from organizations and accordingly prove possession of these credentials to third parties. A credential system where transactions performed by the same user cannot be linked is an *anonymous credential system* [37, 38, 45, 40, 66, 28] as it allows the user to remain anonymous across multiple transactions because of the unlinkability. In particular, in such a system, a user can unlinkably prove possession of a credential without involving the issuing organization, while the transaction cannot be related to the initial issuance transaction. Depending on the implementation, possession of a credential may even be unlinkably proved an unlimited number of times (see Section 2.6). Due to their unlinkability and anonymity properties, anonymous credential systems are particularly relevant in scenarios where the privacy of users must be protected.

Apart from the (multi-show) unlinkability, some anonymous credential systems [28, 15] have a number of additional powerful privacy-preserving features. For example, when proving credential possession, users can disclose just a subset of the attributes that are certified in a credential, or even prove *the mere fact* that the attributes from one or multiple credentials have certain properties—without disclosing the underlying attribute values. The latter feature is particularly relevant with respect to anonymity retention as attribute-based credentials may contain identifying attributes such as a user's name and date of birth. By means of the mentioned features, a user can, for example, choose to disclose only the fact that the date of birth is in a certain range, rather than disclosing the date itself.

To mitigate the risks arising from transactions in which users remain anonymous, anonymous credential systems optionally allow for *anonymity revocation* where a user's anonymity can be lifted by a trusted third party (TTP) under well-defined conditions that the user agrees to (see Section 2.3.4) beforehand. This features adds accountability to otherwise anonymous transaction.

The anonymous credential systems that we consider have strong security properties. In particular, anonymous credentials are *unforgeable* in the sense that users cannot show credentials that they have never obtained, and are *consistent* in the sense that each credential belongs to a well-defined user such that multiple users cannot collaborate to obtain privileges that one user alone would not have gotten. At the same time, different levels of *non-transferability* discourage a user from sharing their credentials with others in the sense that sharing a credential entails sharing a valuable secret—such as the information to access the user's bank account—and/or all of the user's other credentials.

The privacy-preserving features of anonymous credential systems together with their security properties enable strong authentication while reducing the information disclosed in the authentication transaction to a necessary minimum, and thus allow for implementing data-minimizing authentication systems (see Chapter 3).

Figure 2.1 shows the entities involved in anonymous credential systems as well as the interactions among them. Throughout this chapter, we will introduce all of these entities and give detailed descriptions of the possible interactions. In the following, we first discuss the general topics of attribute-based credentials, pseudonyms, and key binding. Then, we elaborate on credential presentation including all advanced features such as attribute predicate proofs and attribute disclosure to third parties. Afterwards, we describe credential issuance; although credentials need to be issued before they can be presented, we discuss the—even—more exciting presentation aspect first. Then, we show mechanisms for rendering credentials unusable, and finally we discuss two well-known implementations of anonymous credential systems.

2.1 Attribute-Based Credentials

A digital credential is a certified bundle of attribute-value pairs issued by a credential *issuer* to a credential user. By issuing a credential to a user, the issuer vouches for the correctness of the certified attributes with respect to the intended user. The attributes certified in a credential are meant to affirm qualification, typically in the form of *identity* and/or *authority*, that is, the credential serves as a means to prove such qualification. For example, an electronic driver's license certifying the holder's name and the permitted vehicle category is used to prove identity and the right to drive motor vehicles of the respective category. As

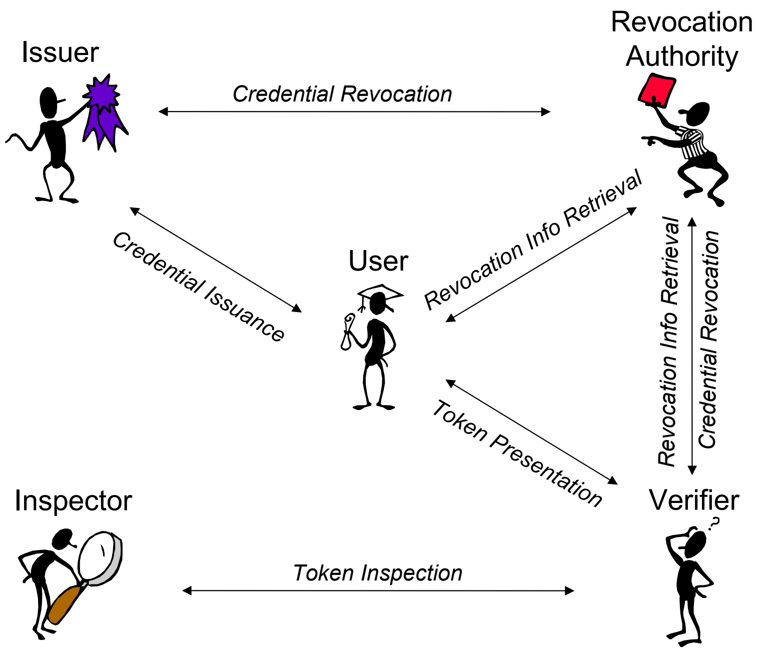


Figure 2.1: Entities and interactions in anonymous credential systems [31, 9]. Credential issuance: see Section 2.4. Token presentation: see Section 2.3. Token inspection: see Section 2.3.4. Credential revocation: see Section 2.5.

the certified attributes are often related to the user's identity, issuers are also called *identity providers* (IdPs). From a technical point of view, a credential is a particular kind of digital signature on a set of attributes that is verifiable under the issuer's public key.

Due to the attribute-based nature of credentials, they are also referred to as *attribute-based credentials* (ABCs). ABCs of an anonymous credential system are referred to as *anonymous credentials*, or *Privacy-ABCs* because of their privacy-preserving potential. An anonymous credential is thus a credential that can unlinkably be used across multiple transactions.

As the meaning of the information certified in credentials has no technical relevance and is subject to interpretation of the party relying on it, the attributes in credentials are described by an *attribute type* that determines the semantics of the attribute (for example, first name) and an *attribute value* that determines its content ("John", for example). Each credential is of a specific *credential type* where—similar to record types in programming languages—a *credential type specification* defines the list of attribute types together with respective data types (for example, first names are strings) that are encoded in credentials of that credential type. For example, credentials of type *Driver's License* may contain the attributes: first name, last name, date of birth, license number, issuance date, and permitted license categories. In addition to defining the contained attributes, a credential type specification may also state whether the respective credentials are revocable and whether they are bound to a secret key (see Sections 2.2 and 2.5, respectively).

A credential specification may be created by the issuer, or by external authorities so that multiple issuers can issue credentials according to the same credential type specification. For verifiers to rely on the semantics of the (ordered) attributes encoded in a credential, the credential type specifications must be published and distributed over a trusted channel—for example, by employing digital signatures.

2.2 Pseudonyms and Key Binding

Anonymous credential systems are also called *pseudonym systems* [66, 65, 37] because verifiers may know users only under (cryptographic) pseudonyms. In traditional public-key cryptography, users generate a public/private key pair that can be used to authenticate the users. Unlike traditional public-key authentication schemes, however, in anonymous credential systems, users may generate as many public keys as they wish from a previously generated *secret key* [28]; these public keys are called *pseudonyms*.

Pseudonyms generated from the same secret key are cryptographically unlinkable, that is, for two different pseudonyms it cannot be determined whether they are generated from the same secret key. However, in case different pseudonyms are associated with the same secret key, the user can (optionally) convince a verifier about this fact without revealing the secret key. When a user authenticates to a party, depending on the required level of linkability, the user can choose to re-authenticate with a previously established pseudonym, or use a fresh pseudonym. By using a different pseudonym for every party a user authenticates to, the user can be known to different parties under different unlinkable pseudonyms, yet use the same secret key to authenticate to all of them. Conversely, to recognize returning users, a verifier may require that users authenticate with an established pseudonym.

There are situations, however, where the possibility to generate an unlimited number of unlinkable pseudonyms is undesirable. For example, in an online opinion poll, users should not be able to bias the outcome by casting multiple votes under different pseudonyms. In such situations, the poll host can request a *scope-exclusive pseudonym*, which is unique for the user's secret key and a given *scope string*. Like standard pseudonyms, however, scope-exclusive pseudonyms generated from the same secret key for different scope strings are cryptographically unlinkable. By using the URL of the opinion poll as the scope string, for instance, the poll host can ensure that each user registers just a single pseudonym to vote.

An anonymous credential can optionally be *bound* to a secret key in the sense that the credential cannot be used without knowing the secret key. Such *key binding* of a credential is indicated in the credential type specification so that verifiers are aware that they must authenticate the respective secret key. To some extent, the key binding is analogous to traditional public-key certificates where the certificate contains a user's public key; but unlike traditional public-key certificates, anonymous credentials are bound to a secret key, rather than a public key. To prevent users from sharing their credentials, a verifier can require a user to show that certain credentials are bound to the same secret key.

If a pseudonym was generated from the secret key that underlies a particular key-bound credential, the user can (optionally) convince the verifier of this fact—that is, that the pseudonym was derived from the same secret key that underlies the credential.

Although it is sufficient for users to generate a single secret key, they may also have multiple secret keys. A secret key can also be generated by trusted hardware (for example, a smart card) that securely stores the key and uses it in computations—for example, to generate pseudonyms—but never reveals the key. The secret key is thus *bound* to the hardware in the sense that it can only be used in combination with the hardware.

2.3 Credential Presentation

After a user has obtained an anonymous credential from an issuer, she can convince a *verifier* of this fact by engaging in an interactive cryptographic zero-knowledge[1] protocol—which is called *proof protocol*. In this protocol, only the user and the verifier are involved. Because the credential issuer is not involved, anonymous credential systems are in terms of privacy protection superior to traditional online authentication mechanisms (such as OpenID [41, 77, 56] or SAML [72]) where the identity provider participates in the authentication and thus learns that the user is transacting with the verifier (see p. 129 for more details). A proof protocol execution is *untraceable*, that is, no collusion of issuers and verifiers can correlate a proof protocol execution to the issuance of the underlying credentials. Depending on the implementation, any two executions of a proof protocol are also cryptographically *unlinkable*, that is, no collusion of issuers and verifiers can determine whether two protocol executions were performed by the same user or by different users (see Section 2.6). We also call this feature *multi-show unlinkability*.

By employing a cryptographic technique that is known as Fiat-Shamir heuristic [49], the proof protocol can be performed non-interactively. That is, the user independently derives a cryptographic object, which we call *token evidence*, and sends it together with an accompanying *proof specification* as *presentation*

[1] Zero-knowledge protocols [46, 76, 12, 8]—also known as zero-knowledge proofs—allow a party to convince another party of knowing a secret without revealing the secret.

token to the verifier for verification.[2] The proof specification is an implementation-specific description of the token evidence and is necessary for verifying the validity of the evidence. As the evidence derivation process is performed without involvement of the credential issuer, it is also called *offline token derivation*. The validity of a derived token evidence, however, is verifiable under the public key of the issuer of the credential underlying the presentation token. This holds true for any party who trusts in the authenticity of the issuer's public key, that is, the verifier can forward the evidence to third parties such as auditors to convince them that he properly authenticated his users. We also refer to the process of providing a verifier with a presentation token as credential presentation or *token presentation*.

In its simplest form, the proof protocol only incorporates a proof that the user owns a particular credential. In addition to performing such *proof of ownership* during protocol execution, users have a number of possibilities for disclosing further information from or about their credential in a privacy-preserving manner. In the following sections, we first elaborate on the proof of ownership and then discuss the additional features of the proof protocol.

2.3.1 Proof of Ownership

When a user performs a *proof of ownership*, the verifier learns nothing more than the fact that the user owns a credential of a particular credential type from a particular issuer. Depending on the semantics of the credential type, this may be sufficient for a verifier to grant certain rights to a user. For example, to enter the premises of a user's employer, it may be sufficient to prove to a physical door lock that the user owns an *Employee* credential issued by the respective employer, without revealing any other information about the credential—such as the certified attribute values. With traditional certificate technologies such as X.509 [42], users have to reveal *all* attributes certified in a credential to prove ownership of it as otherwise the digital signature cannot be verified.

A credential's type and issuer are the minimal amount of information that a verifier must learn for being able to verify a user's credential ownership: the issuer's public key is necessary for the cryptographic verification, and the credential's type is needed because it determines the semantics of the credential and its attributes. In fact, it is even possible to merely disclose that a credential is issued by a member of a group of issuers, rather than a particular issuer [43]. In this work, we assume that issuers may issue credentials of different types; in case an issuer only issues credentials of a single type, then the disclosure of the credential's type is redundant because this semantic information is implicitly carried by the public key. For key-bound credentials—that is, credentials whose credential type

[2]Note that one contribution of this thesis is an implementation-agnostic language for describing token evidence on the one hand [9, 21], and algorithms for transforming *token descriptions* expressed in this language to implementation-specific proof specifications [9] on the other hand (see Section 4.1).

specification indicates that they are bound to a secret key—the proof of ownership may also incorporate the authentication of the underlying secret key. Accordingly, the verifier also learns that the user indeed knows the corresponding secret key. Optionally, within a proof of ownership, users may also show that a credential is bound to the same secret key as another credential or a particular pseudonym.

During one protocol execution, ownership of multiple credentials may be proved. This is relevant in cases where relations between attributes from different credentials are proved, for example (see Section 2.3.3).

2.3.2 Selective Attribute Disclosure

When a user proves ownership of an anonymous credential, she can individually decide for every attribute that is certified in the credential whether its value is disclosed or not, that is, the credential's attributes can be disclosed *selectively.* In case the user does disclose one or more attribute values, the verifier can verify the authenticity of these values and can thus rest assured that they are indeed correct with respect to the credential that was originally issued. As already mentioned in the previous section, the possibility to disclose only a subset of a credential's attributes is in contrast to conventional digital certificate technologies (for example, X.509 [42]) where a certificate can only be revealed as a whole together with all the attributes' values in an all-or-nothing fashion.

To retain the unlinkability properties offered by anonymous credential systems, it is important to consider that an authentication transaction is only as unlinkable and anonymous as the information that is intentionally revealed in this transaction. More precisely, in case a user reveals, for instance, a combination of her first name, last name, and date of birth in two different authentication transactions with the same verifier, then these transactions are obviously linkable and not anonymous—although the cryptographic unlinkability is still guaranteed.

2.3.3 Attribute Predicate Proof

Rather than revealing the exact values of attributes that are certified in credentials, anonymous credential systems also allow for proving *the mere fact* that certified attributes have certain properties—without disclosing the underlying attribute values. For example, a user can prove that her date of birth according to her passport is before September 24, 1994—that is, that she is older than 18 years—without revealing her date of birth.

The proved properties may also involve multiple attributes. For example, assuming that a company offers premium health insurance to customers whose body mass index (BMI)—which is calculated as a person's weight divided by the square of its height—is between 18 and 25, a user can prove this fact without disclosing her exact weight, height, or BMI [70].

Further, the proved properties can involve attributes from multiple different credentials. For example, a user can prove that the name on her credit card matches the one on her driver's license—without revealing the actual name.

Properties that a user can efficiently prove include equality and inequality (that is, not equal to, greater than, less than, etc.) relations between attributes and between attributes and constants, membership relations between an attribute and a list of constants, as well as logical combinations (that is, conjunction and disjunction) thereof. Although in theory any mathematical property can be proved, only for certain classes of properties—such as the preceding ones—techniques for proving them *efficiently* are known [3].

The properties that a user proves about her credentials are expressed as logical predicate in the presentation token description. In addition to such *attribute predicate proof*, a user has to provide proofs of ownership for all the credentials that are involved in the proved predicate. In terms of privacy preservation, *attribute predicate proofs* are of paramount importance in anonymous credential systems because they enable users to fulfill authentication requirements without revealing any information beside the mere fact that the requirement is fulfilled—modulo the requirement itself and the involved credentials' type and issuer.

With other offline authentication technologies (for example, X.509), the only way to show that an attribute satisfies a certain condition is to reveal its value. Although with online authentication mechanisms (OpenID or SAML, for example), an identity provider could also issue a statement that a user's attributes have certain properties, such approach compromises the user's privacy as the identity provider is involved in—and thus knows about—every of the user's authentication transactions.

2.3.4 Attribute Disclosure to Third Parties

Absolute and unconditional user anonymity in online authentication transactions can easily lead to abuse of the used services. Therefore, anonymous credential systems offer the option to add *accountability* to otherwise anonymous transactions, so as to hold misbehaving users accountable for their actions. This is enabled through a feature called *verifiable encryption* [20, 32, 5] that allows a user to *verifiably* and selectively encrypt pseudonyms or attribute values from her credentials such that they can only be decrypted by a designated trusted third party (TTP)—which is also called *inspector*—under previously well defined conditions. Verifiability here means that the inspector can check whether indeed the stated certified attributes are encrypted. An *opening label* describes the condition under which the inspector is permitted to decrypt the attributes. This label is inseparably tied to the ciphertext with cryptographic means such that the label cannot be altered—for example, by the verifier. Note that verifiable encryption in anonymous credential systems is also referred to as *attribute disclosure to third parties* [31] or *inspection* [21].

As verifiable encryption is such a crucial feature for mitigating the risks arising from the use of anonymous credentials, in the following we provide a detailed description of how attributes are verifiably disclosed to a third party. At first, the user and the verifier agree on an opening label, a set of attributes or pseudonyms to encrypt, and a commonly trusted inspector. For example, the opening label in a car rental transaction may be “In case of damage to the rental car with license plate number LPN-1234 in the time between July 7, 2012 and July 9, 2012.” To ensure accountability, the attributes that are to be encrypted must be strongly identifying, such as the document number of a driver’s license credential, or a combination of the user’s first name, last name, and date of birth according to a passport credential.

Then, the user encrypts the chosen attributes under the public key of the inspector and the opening label, which procedure ensures that the opening label is inseparably connected with the resulting ciphertext and that only the inspector can perform the decryption of the attributes. In particular, the inspector can only decrypt the ciphertext with a combination of the opening label and its private key.

Finally, the ciphertext is provided to the verifier in the course of the authentication transaction, together with other cryptographic evidence that supports the proofs of ownership of the credentials underlying the verifiably encrypted attributes. Upon receiving the ciphertext, the verifier can verify that indeed the agreed attributes with respect to the agreed credentials are encrypted and that the agreed opening label is used—however, without learning the cleartext values of the attributes. The received and verified ciphertext accordingly acts as security deposit for the verifier, which is forwarded to the inspector for decryption in case the opening condition is fulfilled. Note that the inspector is neither involved in the attribute encryption nor in the authentication transaction, that is, the inspector is not required to be online at the time when these steps are performed.

In case the rental car is crashed and the verifier needs to identify the responsible user, the verifier provides the inspector with the ciphertext, the opening label, and evidence that the opening condition is fulfilled (that is, the damaged card). The inspector then checks by means of the evidence whether the opening condition is indeed fulfilled, that is, he inspects the condition of the car, decrypts the ciphertext accordingly and discloses the retrieved identifying information to the verifier.

The described scenario is possible under the assumption that both the user and the verifier trust the inspector. The user has to trust the inspector in two ways. First, in that the inspector decrypts the ciphertext (together with the opening label) only in accordance with the opening condition that is specified in the opening label (that is, not without a legitimate reason) after having thoroughly examined the fulfillment of this condition. Second, in that the inspector forwards the decrypted information to the verifier only if the opening condition is fulfilled. The verifier has to trust the inspector in that it will indeed decrypt the ciphertext and forward the retrieved information in case the opening condition is fulfilled.

The fact that the user forwards the ciphertext to the intermediary verifier and not directly to the inspector—that is, the entity who owns the decryption key—

can be seen as additional layer of security. To lessen the necessary trust in the inspector and thus avoid being dependent on the honesty of a single inspector, the responsibility can be distributed over multiple inspectors in the sense that at least k out of $n \geq k$ inspectors must participate in decrypting the ciphertext [47, 59] to successfully revoke the users anonymity.

2.3.5 Message Signatures

In electronic commerce, it is often often required to consent to certain contractual clauses such as a merchant's general terms and conditions. Depending on the necessary level of security, merchants may obtain consent by requiring a user to simply tick a designated check box or by requiring her to provide a digital signature [78, 63, 67], for example. While the former lacks non-repudiation properties as the merchant obtains no cryptographic evidence, the latter is unsuitable in scenarios where the user's privacy must be protected because traditional digital signatures make the user identifiable and linkable.

When authenticating with anonymous credentials, a user can include a statement (or *message*) in the presentation token to express agreement with that statement and thus can effectively digitally *sign* that statement. The difference to traditional digital signatures is that the verifier does not necessarily learn who consented to the signed statement: the verifier only knows as much about the signing party as is revealed by the presentation token that contains the message. For example, in an online shopping transaction, a verifier might only learn that *somebody* who is of age and whose credit card limit exceeds 1000 euro has signed the general terms and conditions. Further, the unlinkability of the cryptographic token evidence guarantees that the signature is unlinkable on the one hand, and that it cannot be repudiated on the other hand.

Message signatures with anonymous credentials, however, can only be made if the proof protocol involves a presentation token, that is, if the protocol is performed non-interactively. Unlike traditional digital signatures, with anonymous credentials there is no actual signature object created; the presentation token acts as signature on the statement. The presentation token is also *transferable* in the sense that the verifier can use it to convince third parties (for example, auditors) of the fact that the user who performed the proof consented to the contained statement.

2.4 Credential Issuance

Credential issuance is the process in which an identity provider issues a new credential to a user and thus vouches for the correctness of the certified attributes. Prior to performing the issuance, the identity provider determines the values that are to be certified in the new credential by authenticating the receiving user and her attributes in some way. This can be done by carrying out some secondary means of authentication—which may involve physical presence of the user and her official documents, for example—or by performing a credential presentation as described in Section 2.3. In the latter case, anonymous credential systems allow for very advanced issuance scenarios: the attribute values from the user's existing trusted credentials can *blindly* be carried over to attributes of the newly issued credential—that is, without the issuer learning these values; the newly issued credential can be bound to the same secret key to which a user's existing credential or pseudonym is bound—without the issuer learning the secret key; and attributes can be assigned jointly generated uniformly random values such that the issuer does not learn the values and the user cannot bias them.

The possibility to carry over attributes from existing credentials is useful in situations where a user does not want the issuer to know the certified value or where an issuer is not interested in learning the value. For instance, for obtaining a digital membership credential of a university library, the library may require the user to first present a valid student ID from which the *name* attribute is blindly carried over to the library card. In this case, the library trusts in the university in that it has thoroughly verified the attributes that it certified in the student ID credential. Issuers can also blindly certify values that are chosen by the receiving user. To do so, a user first issues a credential with the desired attribute value to herself and then presents it to the issuer, who blindly carries over the value to the newly issued credential. For example, assuming that a writers club issues membership credentials that contain an attribute expressing a member's pen name, a member may choose its pen names itself without the club learning it.

Binding a newly issued credential to the same secret key as an existing credential ties these credentials together and thus allows for preventing users from sharing their credentials. In particular, a user can show to a verifier that the presented credentials are owned by the same user—that is, not shared—by proving that these credentials are bound to the same secret key. A newly issued credential may also be issued to a (cryptographic) pseudonym by binding it to the same secret key that underlies the pseudonym, that is, by carrying over the secret key from the existing pseudonym.

An issuance transaction that supports the previously described advanced scenarios is an interactive cryptographic protocol—called *issuance protocol*—between a user and an issuer. Before the protocol starts, the issuer provides the user with a specification that describes which credentials and pseudonyms a user has to present prior to being issued the new credential, and what the relation between the (possibly undisclosed) attributes of the presented credentials and the newly issued credential and its attributes is. Then, the user initiates the actual protocol by providing the issuer with a special presentation token—which is also called *issuance token*—at the end of which the user obtains the new credentials.

The authentication requirements that an issuer applies to a user who requests a new credential are part of the overall security measures that the issuer applies to a particular issuance process. The exact security measures that the issuer applies are usually dependent on the type of the credential that shall be issued. For example, the security requirements for issuing a *Passport* credential are obviously more stringent than the ones for a *Membership* credential of a library. Continuing this example, for issuing passports, an issuer may demand the physical presence of the requesting user as well as two witnesses who guarantee that the claimed identity of the user matches with her true one.

Depending on the applied security measures, the overall issuance process for a particular credential type including all organizational issues may be very involved and time-consuming. Therefore, under certain conditions, anonymous credentials provide a non-interactive *update mechanism* [27] that allows for changing attribute values without having to engage in a complete interactive issuance process. To this end, an issuer provides a user with update information that the user can employ to derive an updated credential. Such update mechanism is for instance useful to prolong the validity of a credential by updating an encoded expiration date attribute.

2.5 Credential Revocation

In situations where a user looses the right to carry a credential or where the security of a credential is compromised, it has to be *revoked*—that is, rendered unusable for subsequent credential presentations—to prevent misuse. This includes scenarios where the attribute values of the credential's user have changed (for example, change of name because of marriage) or where the corresponding user secret was compromised. In these cases, the credential must be revoked *globally*; as the authority for global revocation is the credential issuer, we call this *issuer-driven revocation*. There are also situations, however, where a credential shall be revoked only for a specific context. For example, a hooligan's national identity card may be revoked for entering sport stadiums, while it should still be usable for all other purposes. Also, consider that in some countries a driver's license may be used for both proving the right to drive a vehicle and personal identification, yet revoking the right to drive leaves the identification capability intact. The authority for such context-specific revocation is the credential's verifier, thus we call this *verifier-driven revocation*. As issuers and verifiers may delegate or outsource credential revocation, revocation is described in terms of dedicated *revocation authorities* (see Figure 2.1)—they are separate entities in general, but may be under the control of the issuer or the verifier in particular settings.

For conventional certificate systems, the revocation of one particular certificate is straightforward because each certificate contains a unique identifier (for example, a user's public key in public key certificates): this identifier can be used to *blacklist* (by means of a certificate revocation list (CRL) [42]) or *whitelist* the credential with the corresponding identifier. For anonymous credential systems, however, revocation of credentials is significantly more challenging because presentation tokens do not necessarily reveal any linkable information that can be used as basis by a verifier for recognizing a revoked credential.

Nevertheless, a number of cryptographic techniques have been proposed [29, 71, 16, 26, 68, 68, 13] that enable the revocation of anonymous credentials. While the various approaches differ in their revocation strategy and efficiency (see Lapon et al. [64] for a more detailed analysis), most of them are usable for both issuer-driven and verifier-driven revocation. The general idea of the described mechanisms is that the revocation authority periodically publishes *revocation information* and users prove as part of their presentation tokens that their credentials are still valid with respect to the published information.

Whether a credential is subject to issuer-driven revocation is defined by the corresponding credential type specification (see Section 2.1). In the case a credential is issuer-revocable, presentation tokens related to this credential always have to contain a proof that it has not been revoked yet. Global issuer-driven revocation is typically achieved by making use of a revocation handle, which is a dedicated unique value that issuers embedded in credentials but which is never revealed by the user. Rather than revealing the revocation handle, the user proves that it is in a published set of valid handles—or not within a set of invalid handles.

In addition to and independent of issuer-revocability, any credential can be subject to verifier-driven revocation. This revocation kind can be based on any combination of attribute values, possibly even from different credentials. For example, a sport stadium operator may revoke combinations of first and last names according to national identity cards for known hooligans. Rather than revealing their first and last name, users who want to enter the stadium can prove that their combination of first and last name is not among the blacklisted ones. In this way, the privacy of respectably behaving visitors is protected while troublemakers are prevented from entering the stadium.

An always viable approach for rendering a credential unusable after a predefined amount of time is to include a dedicated *expiration date* attribute in it. When such a credential is presented, the user must accordingly provide an attribute predicate proof that the encoded expiration date is after the date of the credential presentation. A combination of an updatable (see Section 2.4) expiration date attribute and a short validity period of a credential may accordingly be employed as alternative revocation mechanism in scenarios where the previously mentioned approaches are not feasible.

2.6 Implementations

Although a number of schemes for realizing anonymous credential systems have been proposed [37, 38, 45, 40, 66, 28, 30, 7], the schemes of Brands [15] and of Camenisch and Lysyanskaya [28] are the most famous ones and have been implemented in the Microsoft U-Prove credential system [17, 73] and in the IBM Identity Mixer (Idemix) [33, 58], respectively. Both implementations are freely available and efficient enough for practical use. To motivate the decision of using Idemix as basis for the prototype implementation of our data-minimizing authentication system (see Chapter 3), in the following we briefly compare the two mentioned schemes.

Depending on the scheme, the unlinkability necessary in anonymous credential systems is established either during the issuance transaction or during the token derivation process of the presentation transaction. Brands' scheme is based on *blind signatures* that were proposed by Chaum [35, 36, 37]. A blind signature scheme allows a party to obtain a digital signature on a message without the signer learning anything about the content of the message. An often used physical world analogy is to sign the outside of a carbon paper lined envelope while a letter containing the message to sign is enclosed in the envelope. By doing so, the signer cannot see the signed message, yet third parties can later verify the signature. To create such signature in the digital world—for example, by using common public key signature schemes such as RSA [79] or DSA—the message first needs to be disguised, that is, blinded. This blinding provides the *unlinkability* that is necessary for implementing anonymous credential systems because the signer cannot link the blinded message she signs to an un-blinded version.

In contrast to blind signatures, the signature scheme of Camenisch and Lysyanskaya does not create unlinkable credential signatures during issuance, but instead allows for transforming the generated signature—which is also called *CL signature*—into an unlinkable one that remains verifiable under the signer's public key. This way, the necessary unlinkability is established in the token derivation process of a credential presentation transaction.

Accordingly, CL signatures allow users to unlinkably prove possession of one and the same credential *any number of times* without involving the issuer, while one and the same U-Prove credential can be used only once without becoming unlinkable. A possible workaround to this issue is to certify not just one but a number of U-Prove credentials (with the same attributes) such that the user can use one at a time without ever using the same credential twice. In scenarios where credentials are stored on resource-constrained devices such as smart cards, however, the increased bandwidth and memory consumption may make such workaround infeasible. The *one-show* properties of U-Prove credentials, however, are beneficial in electronic cash scenarios [39, 24] where money must be spent only once.

Chapter 3

Data-Minimizing Authentication Systems

THE goal of this doctoral project was to build a functional, usable, and privacy-preserving authentication system on the basis of anonymous credentials. Recall that authentication is the process of establishing confidence in the truth of some claim (see Chapter 1). Further recall that anonymous credential systems offer a number of outstanding privacy-friendly capabilities, most notably the ability to prove to a verifier that a user's certified attributes have certain properties without disclosing identifying information (see Chapter 2). Consequently, with anonymous credentials, users can establish confidence in the truth of a claimed attribute property—that is, authenticate—in a privacy-preserving manner. Such privacy-preserving authentication, however, is only possible in authentication systems where the user knows the minimal authentication requirements in advance, because otherwise she runs the risk of disclosing too much information and thus of forfeiting part of her privacy. In this work, we designed and implemented such *data-minimizing authentication system* where users are informed about the authentication requirements beforehand such that they can prove nothing more than *the fact* that they fulfill the requirements.

The most essential ingredient for a functioning data-minimizing authentication system is a *policy language* to express and communicate authentication requirements in terms of attribute-based credentials and pseudonyms—such that users can fulfill the stated requirements by means of anonymous credential proofs. Moreover, adequate components for processing these policies are needed, so as to generate and verify corresponding technology-specific proofs on the user and the service provider side, respectively. Because a user may have multiple options on how to fulfill a given policy with her credentials, also a component for selecting the preferred option is required. Additionally, a communication protocol that specifies both the interaction between the components and the involved messages must be defined.

In the following sections, we first describe general characteristics of data-minimizing authentication. Then, we discuss the sequence of data-minimizing authentication transactions. Afterwards, we elaborate on the policy and token languages required in such transactions, and next we explain all the components involved in data-minimizing authentication systems in detail. Subsequently, we discuss the security properties of such systems and finally we provide details on our prototype implementation together with timing results.

3.1 Introduction

To fully unfold the privacy-preserving power of anonymous credentials by disclosing only as much information as strictly necessary—that is, as little information as possible—for a successful authentication, a user must be aware of the exact authentication requirements *prior* to performing the authentication. For example, consider an online poll where only citizens with certain postal codes are eligible to vote. For such poll, it is in the privacy interest of a voter to merely reveal that she *is* resident of the designated area without disclosing her exact home address or her name or any other personal information. Accordingly, for the user to just prove residence in the eligible area, it is obviously essential to have knowledge about the respective authentication requirements beforehand.

Existing authentication systems are not designed to inform a user about the exact authentication requirements as they typically perform *identity authentication*. That is, the user is first associated with a previously established identity (for example, a user account) by authenticating an identifier (a username, for example) that refers to this identity, and then the service provider decides on the basis of—the attributes associated with—this identity whether access to a requested service or resource is granted. One the one hand, this decision is taken only after the user has established her account—which is unfavorable for the user's privacy—and on the other hand the user is usually left in the dark about the actual authentication requirements that apply to the current service request. Moreover, these authentication requirements may be different from the account creation requirements in the sense that the latter demanded disclosure of more personal data than the former would have done.

Consequently, for supporting fully privacy-preserving authentication settings with anonymous credentials, a prerequisite is a paradigm shift on how authentication is approached—namely, a shift towards systems that are open with respect to their minimally necessary authentication requirements in the sense that those are communicated to the user in advance. Part of this necessary paradigm shift also concerns the way how authentication requirements are expressed; specifically, this must be done in terms of attribute properties rather than in terms of attributes whose full values must be disclosed. Continuing the previous polling example, the poll host should state the authentication requirements in terms of the range in which the user's postal code must lie (for instance, to entitle all residents of the region Zürich, Switzerland, the postal code must be between 8000 and 8999), rather than requesting disclosure of the exact home address.

In this chapter, we describe an authentication system that lives up to the described expectations. Because of its potential to limit the disclosed attribute information to a minimum with respect to some given authentication requirements, we call it a *data-minimizing authentication system*. More specifically, we say that an attribute-based authentication system is *data-minimizing* under the following two conditions: (1) authentication requirements are expressed in terms of the minimal claim that must be authenticated and (2) it allows for establishing confidence in the fulfillment of authentication requirements without disclosing any information other than this fact. For the cryptographic protocols that we consider in this work, the second condition implies that the authentication requirements—which are expressed in *authentication policies*—are known to the user beforehand.[1] It also implies that the system has the necessary technical capabilities for minimizing the information disclosure. In a data-minimizing authentication transaction, a user makes an informed decision on whether to authenticate under the given requirements or not. Accordingly, data-minimizing authentication systems are *user-centric* in the sense that the user has control over the information that is disclosed by performing this affirmative step.

We say that *the information disclosed in an authentication transaction is minimal* if it does not reveal more information than was requested by the stated authentication requirements. For example, if access to some service is granted to users who are at least 60 years of age according to a national ID card, proving that one is over 65 according to such card or even the revelation of the exact date of birth would not be minimal.

Whether the authentication requirements themselves are minimal, or even legitimate, is a separate concern. We say that *authentication requirements are minimal* if they demand only the amount of information that is strictly necessary in the given context. In this respect we rely on an interplay between the server's fairness and the user's judgement. That is, we assume that users make a conscious decision on whether certain authentication requirements are suitable and acceptable for a given authentication scenario. One may also envision designated authorities who define and certify minimal authentication requirements for certain standard scenarios such as buying a book online. Users would then have the possibility to compare whether the required information is minimal according to a trusted policy certification authority; the provided policy could be digitally signed by the authority, or the user compares the content of the given policy with an authentic reference policy published by the authority.

The authentication transactions performed in a data-minimizing authentication system potentially preserve the authenticating user's privacy. Whether a particular authentication transaction is privacy-preserving or not depends both

[1] There exist oblivious transfer schemes where the user does not learn the policy yet can obtain access to a resource without disclosing the values of her credentials [23]. However, we do not consider these schemes here because they are tailored to database query scenarios.

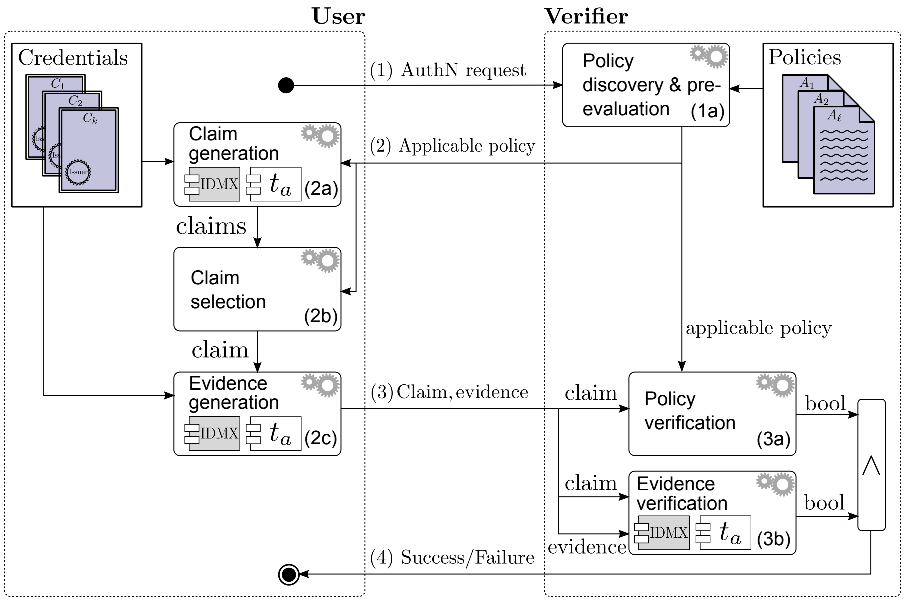


Figure 3.1: Sequence and components of a data-minimizing authentication transaction with anonymous credentials [21]. IDMX: Identity Mixer software module. t_a: module placeholder for additional technologies. $\wedge$: logical conjunction.

on the authentication scenario and the respective authentication requirements. In any case, the level of privacy preservation is also dependent on how the system is implemented. For example, consider an online identity provider who confirms to a service provider—after a user's explicit agreement—that some attribute property of the user is fulfilled. Although such setup is data-minimizing according to the preceding definition, the user's privacy is clearly less preserved (because the issuer is involved in the authentication transactions) than in a system that functions with anonymous credentials (as they allow for offline evidence derivation).

Note that the previously mentioned identity authentication is not necessarily in conflict with privacy interests. The general problem with such identities is that they usually consist of, or at least include, sensitive personal information (SPI). In the data-minimizing and privacy-preserving authentication system that we propose, however, identities can merely consist of non-sensitive attribute properties where a (cryptographic) pseudonym (see Section 2.2) acts as identifier for these identities.

3.2 Data-Minimizing Authentication

In this section, we discuss the sequence of a data-minimizing authentication transaction with anonymous credentials. The entities involved in such a transaction are a *user* and a service provider (which we simply call *server*). Figure 3.1 depicts the sequence of such a transaction as well as the individual components required by the user and the server. We assume that the user already holds anonymous attribute-based credentials that she accordingly obtained from identity providers, and that she wants to use a service (for example, a teenage chat room) that is hosted by the server. For using the service, the server requires the user to authenticate with respect to a service-specific authentication policy. As mentioned in the previous section, an important aspect of data-minimizing authentication is that the policy is formulated in terms of properties of the user's attributes, which are certified in her credentials (see Section 3.3). For example, the policy could state that only teenagers according to an ID card—that is, users who are between 13 and 19 years old—may use the service.

Upon receiving a request (Message 1 in Figure 3.1) for a service, the server determines and pre-evaluates the applicable authentication policy (Component 1a in Figure 3.1) and sends it to the user (Message 2). During the pre-evaluation, dynamic policy elements such as the current date and time are resolved (see Section 3.4.1). For each piece of information whose disclosure is demanded by the policy, the server may optionally specify in a *usage control policy* how the disclosed information will be treated (for example, how long the data will be retained or for which purposes it will be used). Having received the policy, the user's system determines which *claims*—that is, statements about a subset of attributes certified in one or more of her credentials—can be made that fulfill the given policy (Component 2a, see Section 3.4.2). 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. Similar to the policy, the statement of being a teenager is made by means of an attribute property, that is, by stating that the user's age is between 13 and 19 years. (As ID cards typically certify a date of birth rather than an age attribute, the user would state that the date of birth lies withing the required range.) The favored claim is then selected (Component 2b, see Section 3.4.3) interactively by the user [10] or automatically by a heuristic that is capable of finding the most privacy-preserving one [87]. As soon as a particular claim is determined, the respective credential technology is instructed to generate corresponding token evidence that reflects this claim (Component 2c, see Section 3.4.4). The claim—which acts as token description—is then sent together with the accompanying token evidence to the server (Message 3) who verifies whether the claim implies the previously sent policy (Component 3a, see Section 3.4.5) and whether the claim's evidence is valid (Component 3b, see Section 3.4.6). If both tests succeed, the user has successfully authenticated with respect to the policy and is granted access to the requested service (Message 4).

One may also imagine scenarios where a user already knows the authentication policy from previous service requests and accordingly sends a token together with her service request.

3.3 Language Framework

To implement the authentication scenario sketched in the previous section on the basis of anonymous credentials, a number of languages must be available. First and foremost, a generic, technology-independent *authentication policy language* that is capable of expressing the server's minimal authentication requirements in terms of attribute-based credentials and pseudonyms is needed. For policy authors—who are not expected to understand the cryptographic specifics of anonymous credentials—it is crucial that the policy language is only concerned with intelligible high-level concepts, but not with cryptographic details. Secondly, a corresponding *claim language* for a user to make statements about her pseudonyms, her credentials, and the properties of attributes that are certified in these credentials is necessary. Thirdly, an *evidence specification language* that directs the token evidence generation and verification in the underlying anonymous credential system is required. While the latter type of language exists for Idemix and U-Prove, the necessary policy and claim languages were developed in this doctoral project because existing languages did not provide the desired expressivity.

For application developers who build data-minimizing authentication systems with anonymous credentials, it is important that the anonymous credential systems can be operated with languages that hide their cryptographic details as much as possible. The currently existing anonymous credential system implementations (of Idemix and U-Prove), however, are operated with implementation-specific evidence specification languages which are complex, unintuitive, and even require cryptographic background knowledge. This makes it almost impossible for application developers to use these implementations. To this end, our generic, implementation-independent policy and claim languages—which abstract away from the cryptographic mechanisms and focus on the underlying concepts—as well as corresponding algorithms for an automatic transformation from the high-level languages into the low-level evidence specifications are indispensable. For this reason, we also show in this work how these transformations are performed [9]. This enables application developers to control the anonymous credential systems solely by means of the generic languages that merely require an understanding of the general credential concepts, but not of the underlying cryptographic mechanisms. Accordingly, lowering the burden for using anonymous credential systems also facilitates the implementation of data-minimizing authentication systems.

To allow for utilizing the advanced *issuing* features of anonymous credentials, adapted versions of the mentioned policy and claim languages are also needed. In fact, to properly operate all aspects of anonymous credential systems that are relevant for data-minimizing authentication, also languages to express credential type specifications, issuer specifications, as well as issuer- and revocation authority parameters are required. On that account, we developed a whole *language framework* [31, 9, 21] (see Section 4.1), that is, a set of coherent languages, that addresses system setup, credential presentation, and credential issuance—including the revocation of credentials. In this framework, the policy and claim languages that are concerned with the presentation of credentials are called *presentation policy languages* and *presentation token languages*, respectively. The ones that are concerned with issuance are called *issuance policy languages* and *issuance token languages.*

Another aspect in data-minimizing authentication systems that is concerned with language development is *usage control.* For each piece of information whose disclosure is demanded by the authentication policy, the server may optionally specify by means of a *usage control policy language* how the disclosed information will be treated (for example, how long the data will be retained or for which purposes it will be used). In case the user's evaluation of the usage control policy shall be automated, also a language to express a user's privacy preferences is needed. In Bussard et al. [18], see page 157, we propose a two-sided usage control language allowing service providers to specify how they intend to treat the requested data on the one hand, and users to specify their usage control preferences on the other. The authentication policy language that we propose contains placeholders for usage control policies that service providers can use to state the intended data usage.

In the following two sections, we briefly introduce the languages associated with credential presentation and issuance.

3.3.1 Presentation Policy- and Token Languages

The presentation policy language that we propose allows for expressing authentication requirements in terms of credentials and pseudonyms in a technology-agnostic way. More precisely, the language is capable of expressing the following types of requirements: which attribute-based credentials of a certain issuer of a certain type a user has to own; which attributes of these credentials have to be revealed; which predicate has to hold over the credentials' attributes—no matter if the attributes are revealed or not; whether the user has to provide pseudonyms—which possibly must have been established in a previous transaction; which pseudonyms and credentials must be bound to the same secret key; which attributes have to be revealed to third parties; and which statements a user has to consent to.

One of the key features in terms of data minimization is the attribute predicate, which is a Boolean expression that allows for applying logical, comparison, and arithmetic operators to attributes, constants, and further expressions. This also includes predicates involving attributes from different credentials. The most important aspect in terms of privacy preservation is, however, that predicates may also involve attributes that are not revealed. The support for pseudonyms

```
own dl::DriversLicense issued-by DeptMotorVehicles
own mc::MemberShipCard issued-by CarRentalCo
own cc::CreditCard issued-by Amex, Visa
own li::LiabilityInsurance issued-by InsuranceCo
reveal cc.number
reveal li.pNo to EscrowAgent under ‘in case of damage’
where dl.issueDate ≤ dateSubtrDuration(today(), P3Y) ∧
      li.guaranteedAmoutUSD ≥ 30.000 ∧
      (mc.status == ‘gold’ ∨ mc.status == ‘silver’) ∧
      dl.name == li.name
sign ‘I agree with the general terms and conditions.’
```

Figure 3.2: Example CARL authentication policy for car rental service [9].

enables users to authenticate pseudonymously while service providers may dictate the use of previously established pseudonyms.

The development of our presentation policy language took place in two steps, where the pseudonym and key binding capabilities were only added in the second step. Moreover, while the initial version of the language—which is called *credential-based authentication requirements language* (CARL) [31]—has a natural-language like syntax, the version supporting pseudonyms is given in XML format [21]. To give the reader an intuition into our policy language, in Figure 3.2 we provide an example CARL authentication policy for a car rental service that illustrates the most relevant features. We refer to Chapter 4 and the individual publications [31, 9, 21, 18] for further details on the full policy and token languages including pseudonyms and key binding.

According to the example policy, users must fulfill the following six requirements: (1) have their driver's license for at least three years, (2) have *gold* or *silver* membership status with the car rental company, (3) reveal their American Express or Visa credit card number, (4) 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, (5) consent to the general terms and conditions, and (6) have the driver's license and the insurance issued to the same name.

Users who fulfill this policy, ideally reveal—aside from their credit card number—merely the *facts* that they do own credentials of the stated types from the stated issuers, those credentials do fulfill the stated predicate, and 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 attribute predicate 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 certain policy elements (for example, Visa) represent aliases (for example, to cryptographic keys) that are resolved by the credential technology that is used to fulfill the policy.

Our presentation token language, which is used to express a user's claim, follows generally the same schema, however, exhibits three differences. Firstly, for credential ownership, presentation policies allow for specifying a list of issuers and credential types where ownership can be proved for any of those issuers and types. To guide the underlying credential technology on which cryptographic key to use for generating the claim's evidence, the presentation token language states just one issuer and type per credential. Secondly, disclosure requests for attributes that are to be revealed to the service provider directly are only meaningful in policies. In the presentation token description, users must rather comply with those requests by disclosing the corresponding attribute values. Thirdly, similar to the disclosure requests for attributes, while presentation policies request one or multiple of a user's pseudonyms, it is the purpose of presentation token descriptions to provide concrete pseudonym values.

3.3.2 Issuance Policy- and Token Languages

In Section 2.4, we mentioned that anonymous credential systems allow for advanced issuance scenarios if the issuance is performed on the basis of a preceding credential presentation transaction. This capability can be employed to implement *data-minimizing credential issuance* scenarios—that is, scenarios where a user discloses only as much information as strictly necessary for obtaining a new credential. Consider a data-minimizing authentication scenario (as described in Section 3.2) with the resource, which is requested by the user and for which she authenticates, being *a credential*. In such a scenario, the authentication requirements become issuance requirements that—in addition to the standard authentication requirements—state the relation between the credentials to present and the new credential that is about to be issued. In particular, this relation is specified in a *credential template* that contains the following information: the *credential type* of the new credential, which attributes of the new credential are *carried over* from which attributes of existing credentials, to which credential or pseudonym the newly issued credential is *bound*, and which attributes are assigned jointly generated *random values*. We call the combination of such a template and a presentation policy an *issuance policy*. Users respond to issuance policies with special presentation tokens—called *issuance tokens*—that contain additional cryptographic evidence so that the data-minimizing issuance can take place.

To enable data-minimizing credential issuance scenarios, adequate languages to express issuance policies and issuance token descriptions as well as respective algorithms that execute the cryptographic issuance protocols on the basis of such policies must be available. In Camenisch et al. [21], see page 93, we take the first step in enabling these scenarios by proposing the necessary languages.

3.4 System Components

In this section we provide detailed descriptions of the individual components involved in a data-minimizing authentication transaction—that is, in a token presentation transaction. In particular, we describe the components shown in Figure 3.1 one by one.

3.4.1 Policy Discovery

As mentioned in Chapter 1, authentication is usually not an end in itself. Most commonly, information systems authenticate users so that their requests to perform actions can be authorized or that they can be held accountable for the actions they perform [60, p. 34]. Therefore, authentication requirements are typically not standing by themselves, but are incorporated into *authorization policies* that govern who is permitted to perform which action on which resource or service. In data-minimizing authentication systems, the *who*-part of such authorization policies states the authentication requirements and is expressed in terms of the characteristics that users must have (for example, being a teenager) for being permitted to access some service (a teenage chat room, for example).

When a service provider receives a service request, the system has to discover the authorization policy that is applicable to the request, extract the authentication requirements, possibly pre-process them, and send them as *authentication policy* to the user. The pre-processing step ensures that all non-constant elements of the authentication policy are evaluated to concrete values so that there is no confusion during policy verification with respect to which values the verification is performed. For example, age requirements are usually expressed by relating the user's date of birth attribute to the current date—that is, the date of the access request—where the policy only contains a placeholder for the current date; this placeholder has to be resolved accordingly.

In situations where multiple policies from possibly different sources govern the access to the requested service, the authentication requirements must be adequately combined. The exact details of such combination, however, are out of the scope of this work.

3.4.2 Claim Generation

When a user receives the authentication policy applicable to her service request (or resource request), the user's system must determine whether and how this policy can be fulfilled. In particular, the user's system locally determines which claims that fulfill the given policy can be made with respect to the credentials available in the user's credential portfolio. To this end, first the different options of assigning credentials from her portfolio to the *credential variables* occurring in the policy such that the policy is fulfilled are computed. We call one such option a *credential assignment*; more precisely, a credential assignment of a user is a total mapping from the credential variables occurring in an authentication policy to instances of credentials that the user owns. We further call a credential assignment *valid* for a given policy if the policy can be fulfilled by assigning the credential variables according to the assignment. All valid credential assignments that are found are then the basis for generating technology-specific claims that fulfill the policy. In case no assignments are found, the user is informed that her credentials are not sufficient for fulfilling the authentication requirements. With the currently available cryptographic techniques, only for credential assignments that are *technology consistent*, that is, that map to credential instances that all have the same underlying credential technology, cryptographic evidence can be generated. Therefore, inconsistent assignments are disregarded during claim generation.

Computational Complexity Of Assignment Determination

An important aspect of claim generation is to determine *all* valid claims—and thus credential assignments—of a user for a given policy, because otherwise a user cannot make an informed decision on what is the most privacy-preserving way to fulfill the policy. The computational complexity of determining all valid credential assignments depends on two factors: the user's credential repository and the authentication policy. The more credentials are required by the policy and the more credentials a user has in her repository, the more assignments must be tested on validity. This is because initially all credentials in the repository are candidates for every credential variable in the policy. In realistic use cases, however, filtering the possible candidates for a credential variable by credential issuer and credential type before testing the validity is sufficient for limiting the number of possibilities to a size that is easily processable. This is because users typically have only very few credentials from the same issuer of the same type.

The computational complexity of assignment determination may only become a problem in situations where these two prime filter criteria (issuer and type) are not sufficient, that is, where a user has a multitude of credentials from the same issuer of the same credential type. In such situations, an optimized variant of the assignment algorithm can be used that employs additional filter criteria on the variable candidates on the basis of the policy's attribute predicate: first, the predicate formula is transformed into disjunctive normal form[2] (DNF) and monomials that only involve a single credential variable—which sub-predicates are called *variable-specific*—act as additional filter criteria for the respective variable. This filter technique may be extended to *cross-credential* monomials, which are sub-predicates that involve two or more credential variables. Depending on the availability of variable-specific or cross-credential sub-predicates, various strategies for combining different filter criteria are conceivable.

[2]An atomic proposition or its negation is called a *literal.* A *monomial* m is a finite conjunction of literals, that is, $m = \ell_1 \wedge \ldots \wedge \ell_n$ where $\ell_1, \ldots, \ell_n$ are literals. *Disjunctive normal form* (DNF): finite disjunction of monomials, that is, $\bigvee_i \bigwedge_j \ell_{ij}$.

A general alternative to pre-computing all valid assignments is to compute them *on the fly* on the basis of the user's input during claim selection.

Technology Dependence Of Claim Generation

In an ideal system implementation, for each valid credential assignment one *minimal* claim is generated, that is, a claim that only discloses as much information as demanded by the policy. However, the currently available anonymous credential system implementations support only a subset of the theoretically—that is, cryptographically—provable statements. Until this deficiency is mended, claim generation algorithms that are tailored to the capabilities (and restrictions) of the implementation at hand must be employed accordingly (see Section 4 on page 141 in Bichsel et al. [9] for detailed restrictions of the current Idemix and U-Prove implementations). Figure 3.1 depicts this as technology-specific software modules (for example, IDMX represents an Idemix module) that are part of the claim generation component; such modules must be available on the user's system for all the technologies (that is, anonymous credential systems) that were used to issue her credentials.

Depending on the particular restrictions, technology-specific algorithms may generate multiple claims from one credential assignment. For example, the current Idemix implementation is not capable of proving disjunctive statements—which is why a claim must be generated for every valid monomial in the DNF of the policy's attribute predicate. It is important to consider the implementation deficiencies already during claim generation rather than during evidence generation because users need to know about these limitations to make informed decisions during claim selection. For example, a user must be informed about the fact that she is about to disclose *which* part of a disjunctive statement she fulfills.

Implicit Information Disclosure Detection

When authentication policies are not verifiable by the user with respect to a trusted policy certification authority (see Section 3.1), users may fall for maliciously formulated authentication policies that are intentionally formulated in an overly complex or confusing manner. For example, if the policy demands that one of the user's attributes must be revealed and the attribute predicate requires this attribute to be equal to another attribute, the latter is obviously revealed too. Another evident example is a policy with an attribute predicate where the value of a numerical attribute is restricted from two sides such that the value is effectively disclosed. Although patters like these may remain unrecognized by credulous or unobservant users, they can easily be discovered by a computer system. Hence, to avoid unintentional *implicit information disclosure*, the user's system must detect such patterns, inform the user during the claim selection accordingly (for example, by displaying the respective attributes as if they were explicitly disclosed), and notify the user about potential malicious intentions of the service providers.

Informative Insufficiency Feedback

In case a user's credentials are not sufficient for fulfilling the given authentication requirements, simply learning that the policy cannot be fulfilled would be very unsatisfying for the user. The assignment algorithm therefore must be implemented in a way such that the user can be provided with error feedback that informs her about the reason *why* she cannot fulfill the policy. Possible reasons are that she simply does not have a credential from the required issuer and/or the required type, or—in case she does—which of the required attribute properties are not fulfilled by her credentials. Having such information, the user has to possibility to obtain new credentials accordingly and authenticate successfully afterwards.

3.4.3 Claim Selection

After the user's system has determined the possible claims that can be made on the basis of the user's credentials to fulfill the given policy, one of these claims must be chosen. For the user to make an informed decision about which of these claims is most suitable for the current authentication transaction—for example, in terms of privacy preservation—a graphical user interface (GUI) assists the user in making her choice. (As claims may disclose information related to a user's identity, such graphical user interface is sometimes referred to as *identity selector*.) Also in situations where the user can make just a single claim to fulfill the policy, this interactive step of involving the user is necessary—or at least strongly advisable—because it informs the user about the information that is about to be disclosed when continuing the transaction. Before the information related to the selected claim is effectively disclosed, the user's system should explicitly obtain confirmation from the user via the GUI, so as to avoid any confusion about whether and when personal information is revealed.

A further purpose of the claim selection GUI is to visually present to a user which information is about to be disclosed to whom in case the transaction is continued. That is, it shall be conveyed which information is about to be revealed to the service provider directly, and which information is indirectly revealed in encrypted form to trusted third parties (that is, inspectors). To minimize the burden on the user, we recommend that the claim selection information and the disclosure information are combined in the GUI. For example, whenever a user selects a claim, the disclosure information related to this claim could be displayed (see Section 3.6).

As described by Wästlund et al. [89], the design of user-friendly interfaces that convey anonymous credentials' privacy benefits to users is a major challenge: users are unfamiliar with the new and rather complex concept of anonymous credentials and additionally no obvious real-world analogies exists that can help them create the correct mental models. The authors recommend an approach where an *adapted card* is used to convey the notion that only the information in the adapted card is sent to the service provider and nothing else. On top of their approach, we recommend that an adapted card is displayed for every information recipient, and that the information in an adapted card is arranged at least in the following categories: proofs of ownership, pseudonyms, disclosed attribute values, and attribute properties. The *proofs of ownership* category merely comprises credential issuer and type information for the credentials of which the service provider will learn that the user owns them; the *pseudonym* category tells the user which pseudonyms—preferably in terms of a human-readable name that is associated with the pseudonym—the service provider will learn about; the *attribute values* category lists all the attributes together with their values which are fully disclosed to the service provider; the *attribute properties* category itemizes all facts that the service provider will learn about the user's credentials (for this purpose, the claim's attribute predicate may have to be transformed into DNF whose monomials act as items in the list that is presented to the user). In addition to the categorization, we further recommend that the GUI assists the user in making an informed decision by indicating whether a selected claim is minimal with respect to the policy or not. Another challenge in conveying to a user which information is actually disclosed and in designing respective GUIs are *cross-credential requirements*, that is, attribute predicates that involve attributes from different credentials. For assisting the user in making the optimal choice in terms of privacy preservation, anonymity metrics may be employed as proposed by Verslype [87].

If a user frequently authenticates to the same service provider under the same authentication policy, the user's system may store the user's decision on which claim is selected and—with the user's explicit agreement—accordingly disclose the related information automatically. Such automatism makes the authentication transparent and more convenient for the user.

In situations where an authentication policy also comprises data handling information, in addition to the information that is about to be disclosed, the GUI must also provide the user with details on how the information is treated *after* it is disclosed. For example, it is relevant for the user for what purpose the disclosed data is used, and with whom and under what conditions it may be shared with whom. In Bussard et al. [18], see page 157, we have developed a dedicated usage control language that is capable of expressing such facts. However, policies expressed in this language first have to be prepared for visual presentation to the user.

3.4.4 Evidence Generation

As soon as a user has explicitly confirmed via the claim selection GUI that a particular claim (which is expressed in a generic token description language) shall be proved to the service provider, the user's system must generate respective cryptographic evidence that supports this claim. Because the existing implementations of Idemix and U-Prove both produce evidence on the basis of evidence specifications expressed in nongeneric languages, the technology-agnostic token description expressing the claim must be transformed into implementation-specific evidence specifications that are semantically equivalent. (For Idemix and U-Prove these evidence specifications are called *Idemix proof specification* and *U-Prove token specification*, respectively.) Thus, for generating evidence for a given claim, the user's system first determines the credential technology underlying the claim and utilizes a respective anonymous credential system module (see Figure 3.1) to transform the claim into a proof specification and to produce cryptographic evidence by means of this specification. Finally, the generated token evidence is sent together with the token description as presentation token to the service provider for verification.

In Bichsel et al. [9], see page 129, we show how a generic token description can be transformed into a semantically equivalent Idemix proof specification that supports all attribute properties except logical disjunctions. This is a crucial step for making Idemix usable for application developers because they no longer have to deal with its complex specifics but can rather control the library solely by means of technology-agnostic and easily understandable token descriptions.

3.4.5 Policy Verification

When a service provider receives a token that is presented by a user to gain access to some service, the service provider verifies whether the statements made in the token description fulfill the authentication policy that was initially provided to the user for this service request. In particular, the token description fulfills the policy if (all of) the following conditions hold: the requested proofs of ownership are provided for credentials of the correct type and issuer; the attribute value disclosure requests are fulfilled—that is, the requested attribute values are disclosed; the proven attribute predicate implies the one requested by the policy; the correct message is signed; the correct information is verifiably encrypted for third parties; and the requested pseudonyms are provided and also have the correct key binding relations to other pseudonyms and/or credentials. In Camenisch et al. [21, Section 5.5], see page 117, we provide precise formal definitions on when a presentation token description fulfills a presentation policy.

Note that verifying whether the preceding conditions hold does not involve the performance of any cryptographic operations, that is, checking whether the token description fulfills the policy is done on a pure logical level, rather than a cryptographic one.

3.4.6 Evidence Verification

The validity of the presentation token (consisting of a token description and token evidence) that is provided by the user must be verified by the service provider. More precisely, the service provider verifies whether the claim that is made in the token description is supported by the accompanying token evidence—that is, whether the claim is authentic. This includes verifying whether the credentials that were used to derive the evidence are valid in the sense that they are not revoked (see Section 2.5). In Camenisch et al. [21, Section 5.4], see page 114, we mathematically define the conditions on when a user can prove a given presentation token with respect to her credential and pseudonym portfolio.

For verifying a token, the service provider's system first transforms the user-provided token description into a technology-specific evidence specification in the same way as this was done by a user's system to generate the evidence (see Section 3.4.4). For the service provider to know which technology was initially used to generate the evidence and hence to know which technology module must be used to verify it, the token description contains a corresponding hint. Then, both the technology-specific evidence specification obtained by the transformation and the token evidence serve as input to the respective cryptographic library for verifying the evidence. The library answers in a Boolean fashion stating whether the evidence supports the evidence specification and thus whether the evidence supports the token description.

3.5 Security Properties

In the following, we define the security properties of our data-minimizing authentication system. For showing that these properties are fulfilled, we rely on the unforgeability property of anonymous credentials (see Chapter 2): users can convince verifiers of the fulfillment of technology-specific evidence specifications only if they own the necessary credentials and pseudonyms. Additionally, we assume that adequate sharing prevention techniques are employed that prevent users from passing their credentials on to others (see Chapter 2).

Security Property 1. A user must not be able to convince a verifier of fulfilling a claim expressed in a token description if the token evidence does not support the claim, that is, a user can convince a verifier of fulfilling a given claim only if this claim is semantically equivalent to the accompanying evidence.

Security Property 1 is fulfilled because the user does not provide the service provider with the technology-specific evidence specification that is used by a cryptographic library for verifying the token evidence, but rather sends the generic token description (Message 3 in Figure 3.1), which is transformed into a technology-specific evidence specification independently by the service provider. The only way how this property can be violated is if a malicious transformation algorithm is unknowingly employed by the service provider, which algorithm does not translate token descriptions to semantically equivalent evidence specifications. Accordingly, in a service provider's uncompromised computing infrastructure, evidence verification (Component 3b in Figure 3.1, see Section 3.4.6) ensures that Security Property 1 holds.

Security Property 2. A user must not be able to convince a verifier of fulfilling an authentication policy if the user's credentials or pseudonyms do not fulfill the policy, that is, a user can convince a verifier of fulfilling an authentication policy only if her credentials and pseudonyms indeed fulfill the policy.

Given a user's token consisting of a token description and token evidence, Security Property 2 is fulfilled because, one the one hand, the service provider's policy verification algorithm (Component 3a in Figure 3.1, see Section 3.4.5) ensures that the token description fulfills the given authentication policy and, on the other hand, Security Property 1 ensures that users can only create token descriptions that are in accordance with their credentials and pseudonyms. This property can only be violated if Security Property 1 is violated, or if the service provider's computing infrastructure is compromised in that a malicious policy verification algorithm is employed without the service provider having knowledge about it. Said differently, as long as the user cannot influence the service provider's policy verification algorithm, Security Property 2 is fulfilled.

From the user's point of view, a desired security property would be to have some form of assurance that, upon successful fulfillment of the stated authentication policy, access to the requested service or resource is granted. On the one hand, the guaranteed permission to computing services seems impossible because in the end users are always dependent on the goodwill of the service provider. On the other hand, for resources that can be transferred to the user before the authentication (for example, in encrypted form), oblivious transfer techniques [22] may be employed. However, this is out of the scope of this work.

3.6 Implementation

To show that our ideas are feasible and efficiently realizable, and to make data-minimizing authentication more accessible for the general public, we have implemented an open-source software framework for rendering data-minimizing authentication decisions on the basis of attribute-based credentials. This *credential-based authentication framework* (CBAFx)[3] provides a toolbox that application developers can use to implement data-minimizing authentication solutions. By means of the framework, we also developed a full-fledged prototype implementation [9, 11] of a data-minimizing authentication system including a graphical user interface for claim selection (Component 2b in Figure 3.1). On the basis of the holistic prototype implementation, we performed time measurements that show the efficiency of our approach. In the following, we describe our framework, the prototype, and the timing results.

3.6.1 Open-Source Authentication Framework

The CBAFx is open to be used with any attribute-based credential technology in that it provides a plug-in mechanism for integrating software modules which

[3]In software development, the term *Framework* is sometimes abbreviated with *FX* or *Fx*.

are responsible for a particular credential technology such as Idemix (see the IDMX modules in Figure 3.1). Our framework is operated by variants of the CARL policy language [31] and a corresponding CARL-based claim language [9]; both variants are subsets of CARL and support the three most crucial features for realizing privacy-preserving authentication: proofs of ownership, selective attribute disclosure, and attribute predicate proofs.

The software framework itself comprises all *technology-agnostic* components: the CARL policy language variant, the corresponding CARL-based claim language variant, language parsers, a type system for attribute predicates, a pre-evaluation function of CARL policies (Component 1a in Figure 3.1), credential assignment determination—together with a mechanism to call the correct technology-specific claim generation module (Component 2a), mechanisms to invoke the technology-specific evidence generation and verification modules (Components 2c and 3b), and policy verification (Component 3a).

In addition to the technology-agnostic components, we also provide the necessary Idemix modules (for Idemix version 2.3.3). These modules are capable of generating claims that are tailored to the restrictions of the Idemix implementation (see Section 3.4.2) as well as of producing and verifying Idemix-specific token evidence. This includes the ability to transform CARL-based token descriptions into Idemix proof specifications. The automatic transformation is a crucial step in facilitating the use of Idemix for application developers. In Bichsel et al. [9], see page 129, we describe the exact details on this transformation.

Our assignment algorithm uses a credential's issuer and type as filter criteria (see Section 3.4.2). It further features a simple insufficiency feedback mechanism that is capable of reporting for which credential variables no candidates were found.

The plugin-mechanism of the framework also allows for the use of credential technologies that are not privacy-preserving (X.509, for example) or not fully privacy-preserving (for example, the current U-Prove implementation): properly implemented claim generation modules can ensure that data-minimizing authentication policies can be fulfilled with these technologies nevertheless. For example, to fulfill a policy demanding proof of an attribute predicate, with U-Prove credentials the attributes that are involved in the predicate have to be revealed, and with X.509 certificates all attributes of all credentials involved in the attribute have to be revealed (see Bichsel et al. [9, Section 4.1] on page 141).

We implemented the CBAFx in Java and employed the Eclipse Xtext language framework [52] (version 1.0) to create both a language parser and the Java object model for CARL, as well as the Xtext Typesystem [88] (version 1.5) to define the type system for the CARL attribute predicates. The CBAFx has been released as Open Source Software under the Eclipse Public License and is available for download at the website of the EU project PrimeLife [75], where also the Idemix cryptographic library is available.

The EU project ABC4Trust [1] is currently building on the results (and partially on the source code of) the CBAFx to implement an extended software framework that supports also the privacy-preserving features of anonymous credential systems that are not supported by the CBAFx—that is, pseudonyms, key binding, message signatures, and verifiable encryption—including issuance and revocation.

3.6.2 Prototype and Time Measurement

As mentioned previously, the CBAFx provides a general toolbox for implementing data-minimizing authentication transactions. For an executable and presentable solution, however, components that handle all the involved messages as well as a claim selection GUI is needed. Therefore, we implemented a prototype that adds these necessary elements.

In particular, our setup comprises the following elements: a static example credential repository, a static web page with an editable CARL policy input field that simulates the server's policy discovery, a Java servlet for the user that is waiting at a well defined local URL for input consisting of one or multiple CARL policies (with disjunctive semantics) as well as a URL to send the presentation token to, and a Java servlet for the server that is waiting for presentation tokens in order to verify them.

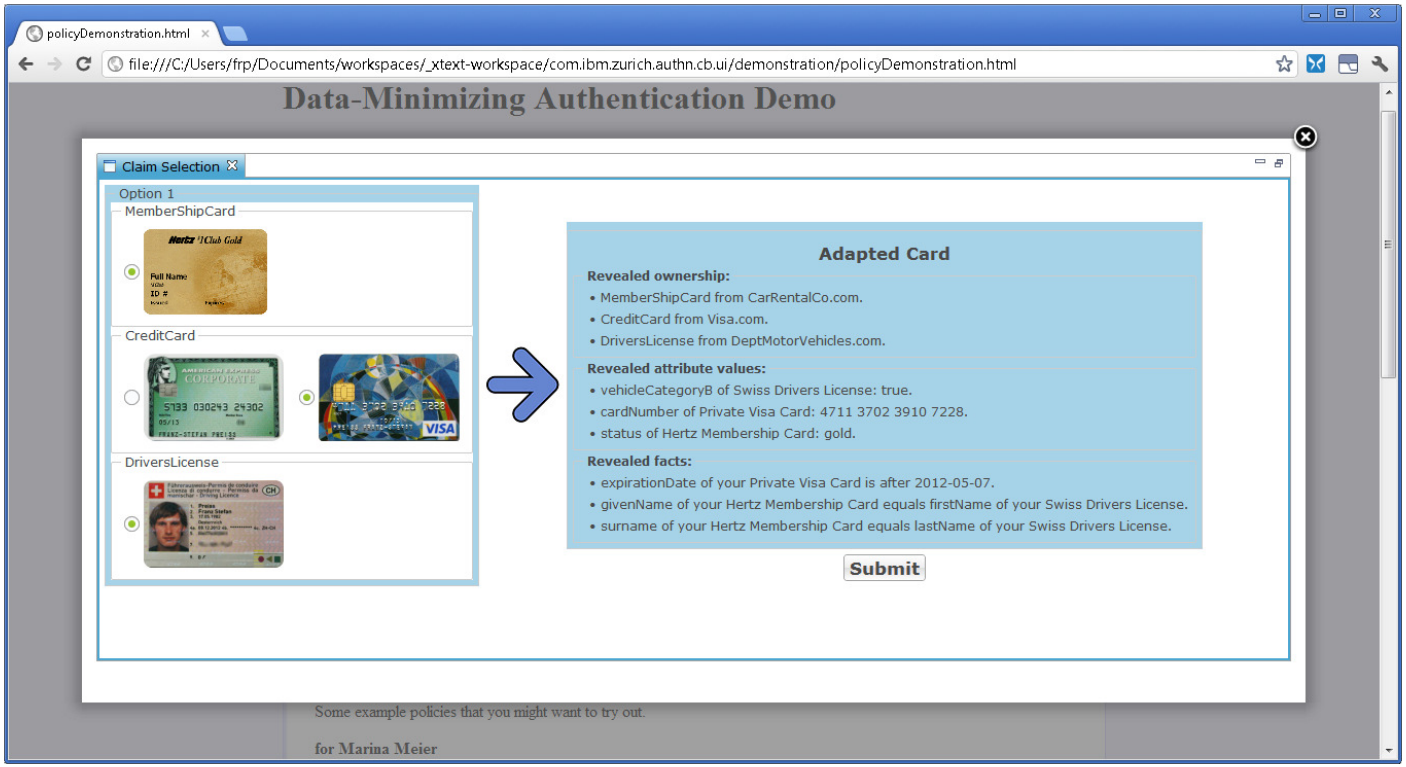
Submitting the input form on the static web page on the user's host in a web browser calls the user's servlet and thus simulates an authentication policy coming from a server (Message 2 in Figure 3.1). The servlet uses the CBAFx to compute the possible claims and accordingly displays a GUI[4] that is shown as overlay in the user's browser. In case the user's credentials are sufficient for fulfilling one of the policies, a claim selection GUI is displayed (see the following subsection). In case the credentials are not sufficient for fulfilling the policies, the GUI informs the user about the credentials that are missing (see Figure A.3 in Appendix A). When the user's servlet URL is called with an empty policy, a GUI displaying the user's credential repository is shown (see Figure A.1 in Appendix A).

After a user confirms the information disclosure in the claim selection GUI, the user's servlet generates a presentation token, transforms it into Base64 encoding—so that it can be forwarded within an HTTP request—and submits the encoded token to the service provider's servlet (residing at the initially provided URL) for verification. The service provider's servlet forwards the user to a web page that shows the success (or failure) of the verification.

Graphical Claim Selection User Interface

Our prototype claim selection GUI follows the *adapted card* proposal of Wästlund et al. [89] as well as the recommendations that we put forward in Section 3.4.3.

[4]We implemented our GUI by means of the Eclipse Rich Ajax Platform (RAP) [51] that allows for implementing GUIs in terms of standard Java code which is compiled to a Java servlet. Interactions with the GUI are translated in real time to *Asynchronous JavaScript and XML* (AJAX) messages: these messages are processed by the servlet container which instantly responds with GUI updates that are shown by the web browser.


policyDemonstration.html
file:///C:/Users/frp/Documents/workspaces/_xtext-workspace/com.ibm.zurich.authn.cb.ui/demonstration/policyDemonstration.html
Data-Minimizing Authentication Demo
Claim Selection
Option 1
MemberShipCard
CreditCard
DriversLicense
Submit

Adapted Card

Revealed ownership:
- MemberShipCard from CarRentalCo.com.
- CreditCard from Visa.com.
- DriversLicense from DeptMotorVehicles.com.

Revealed attribute values:
- vehicleCategoryB of Swiss Drivers License: true.
- cardNumber of Private Visa Card: 4711 3702 3910 7228.
- status of Hertz Membership Card: gold.

Revealed facts:
- expirationDate of your Private Visa Card is after 2012-05-07.
- givenName of your Hertz Membership Card equals firstName of your Swiss Drivers License.
- surname of your Hertz Membership Card equals lastName of your Swiss Drivers License.

Some example policies that you might want to try out.

for Marina Meier

Figure 3.3: Credential selection GUI in prototype application.

In particular, in addition to claim selection functionality, it is visualized which information is revealed with the currently selected claim in case the user proceeds.

Figure 3.3 shows such GUI as browser overlay for a policy that has three credential variables (see Appendix A.1 for the exact policy). For every credential variable, a *variable box* is shown that contains icons for all credentials of the user's repository that appear in any of the computed credential assignments for this particular variable. To proceed, users have to make a selection in every box where only one icon per box can be selected at a time (as indicated by the radio buttons in Figure 3.3). For the user's convenience, the GUI automatically pre-selects the credentials according to the first credential assignment that is found. At the same time, the GUI ensures that only valid combinations of credentials can be selected across the different boxes, that is, that there exists a credential assignment that contains all the currently selected credentials.

What is actually chosen in the GUI by means of the variable boxes are credential assignments. Because there is not a one to one relation between credential assignments and claims (for example, because Idemix cannot prove disjunctive statements, see Section 3.4.2) an additional selection mechanism is needed if a selected assignment maps to multiple claims.

When there is a credential icon selected in every variable box—and it is unambiguous which claim corresponds to that respective credential assignment—that claim is displayed in a structured form as *Adapted Card* consisting of three parts. The first part shows what information is disclosed in terms of credential ownership—that is, it contains information of which type and from which issuer the selected credentials are. The second part shows the information revealed to the service provider in clear text, and the third part indicates the attribute properties (that is, facts) that are revealed to the service provider. Thus, it is clearly displayed what information is disclosed to a service provider in case the user proceeds and proves the claim.

In case multiple disjunctive policies are provided to the claim selection GUI, a set of variable boxes is displayed for every policy (see Figure A.2 in Appendix A).

Timing Results

Table 3.1 shows time measurement results that concern all aspects of a data-minimizing authentication transaction performed with our prototype implementation. We structure our timing results in three phases: a user's *preprocessing* phase—covering the time period from receiving an authentication policy to finishing the rendering of GUI elements for the claim selection;[5] a user's *postprocessing* phase—covering the time period from a user's confirmation of information disclosure in the browser (via a *Submit* button) to being ready to submit the generated presentation token to the service provider; and a server's *verification* phase—covering the time period from receiving a presentation token

[5] This does not include the time that the web browser takes for rendering the generated AJAX update. However, this factor is negligible.

Phase	1. Run	2. Run	3. Run	**Mean**
Preprocessing (user)	1296 ms	1301 ms	1272 ms	**1290 ms**
Policy parsing	923 ms	961 ms	915 ms	72.3 %
Assignment determination	16 ms	16 ms	17 ms	1.3 %
Claim generation	35 ms	34 ms	33 ms	2.6 %
Claim selection GUI rendering	322 ms	290 ms	307 ms	23.8 %
Postprocessing (user)	1464 ms	1484 ms	1496 ms	**1481 ms**
Claim transformation	28 ms	24 ms	33 ms	1.9 %
Idemix evidence generation*	1357 ms	1382 ms	1385 ms	92.8 %
Token serialization to Base64	79 ms	78 ms	78 ms	5.3 %
Verification (server)	1499 ms	1547 ms	1516 ms	**1521 ms**
Token de-serialization from Base64	58 ms	76 ms	67 ms	4.4 %
Claim transformation	7 ms	7 ms	8 ms	0.5 %
Idemix evidence verification*	1289 ms	1306 ms	1287 ms	85.1 %
Policy verification	145 ms	158 ms	154 ms	10.0 %

Table 3.1: Prototype implementation time measurements. Indentation in the first column indicates sub-phases. Policy parsing is the transformation of the textual CARL policy into Java object form. Legend: * cryptographic operation.

to reaching a decision on whether the authentication is successful. The table shows the timing results for three consecutive independent test runs together with arithmetic means in absolute numbers (for the totals) and as percentage (for the sub-phases). For simplicity, we run both the user's servlet and the server's servlet on the same host in the same servlet container. To minimize caching influences, the web server hosting the respective Java servlet was restarted between the individual test runs.

During the test runs, the user's credential repository comprised 6 Idemix credentials (see Appendix A.2.1), and the authentication policy contained 4 credential variables and an attribute predicate with 9 monomials (see Appendix A.2.2). The cryptographic keys that were used to issue these Idemix credentials had a 1024-bit RSA modulus. The exact test data and cryptographic parameters that were used for performing the time measurements are given in Appendix A.2. As test environment served a Lenovo notebook, model X220, with an Intel Core i5-2540M processor (2.6 GHz clock speed, 2 cores, 4 threads), 4 GB memory, a 64-bit Windows 7 Professional operating system (Service Pack 1), and 32-bit Java version 1.6.0_29.

For the user's preprocessing phase, the results show that policy parsing (that is, transformation of the textual CARL policy into Java object form) and the rendering of the claim selection GUI amount together to average 96.1 percent of the total average 1290 milliseconds. That is—at least for the given test data—the time spent on assignment determination and claim generation is with approximate total 50 milliseconds and 3.9 percent temporally as well as proportionately very short. For the reasons given in Section 3.4.2 on computational complexity, we expect that this desirable result also scales to scenarios with larger credential portfolios and more complex policies than the ones used in our test environment. Noteworthy, however, is the considerably long policy parsing sub-phase. This can be explained by the use of a highly flexible language framework [52] (which is capable of handling user-defined domain-specific languages) for parsing, rather than a dedicated parser that is specifically optimized towards our policy language. The total 1.29 seconds for the preprocessing phase are a reasonable period of time, but future research should focus on reducing this figure (for example, by employing an optimized language parser) to improve the user's experience.

In the user's postprocessing phase, the Idemix evidence generation takes as cryptographic operation with 92.8 percent of the total 1481 milliseconds as expected clearly the major part. The time spent on claim transformation is with 1.9 percent negligible. A reduction of the reasonable total 1.48 seconds is mainly dependent on efficiency improvements of the cryptographic evidence generation operation—but this is out of the scope of this thesis.

In the server's verification phase, again, the cryptographic Idemix evidence verification operation amounts with 85.1 percent as expected to the main part of the total 1521 milliseconds. The non-cryptographic policy verification takes with approximate average 152 milliseconds 10 percent of the total time spent on token verification. Also in this phase, the total 1.52 seconds are reasonable and a time improvement is mainly dependent on the underlying cryptographic mechanisms. The average 7 milliseconds for claim transformation in this phase are remarkably faster than the average 28 milliseconds that the transformation of the same claim took in the postprocessing phase. This speed increase can be explained by caching dynamics because both the user's servlet and the server's servlet are running in the same servlet container.

Summarized, the computational overhead entailed by the algorithms proposed in this thesis are negligible with respect to the time needed to generate and verify the cryptographic evidence that supports users' claims.

Chapter 4

Contributions

Summarized, the contribution of this work can be described as the development of the missing building blocks that are necessary for implementing an authentication system that preserves its users' privacy by minimizing the disclosed information with respect to an authentication policy. In particular, we provide a language framework for privacy-preserving attribute-based credentials, we describe evaluation algorithms for determining whether credential-based authentication requirements are fulfilled, we define a formal system semantics of our authentication system, and we show how standardized infrastructure can be extended to support our approach. In the following, we outline our research contributions and describe how the individual publications that are included in this thesis fit together.

4.1 Language Framework for Privacy-Preserving Attribute-Based Authentication

Our contributions in the language space can be summarized as abstracting away from all specifics of anonymous credential systems, focusing on the underlying identity concepts rather than the cryptographic mechanisms, casting these concepts into one coherent language framework that is instantiable with different technologies or even combinations thereof, and therefore making them available to application developers without requiring them to know the specifics of the underlying cryptographic mechanisms.

Credential-Based Authentication Requirements Language (CARL)

In Camenisch et al. [31], see page 193, we propose a policy language for expressing authentication requirements in terms of attribute-based credentials. We refer to this *credential-based authentication requirements language* as CARL. The language is specifically targeted at the privacy-preserving functionality supported by anonymous credential systems (see Chapter 2). In particular, CARL is capable of expressing which attribute-based credentials a user has to own (i.e., proof of ownership), which attributes of these credentials have to be revealed (i.e., selective disclosure), which predicate has to hold over the credentials' attributes—no matter if the attributes are revealed or not, which attributes have to be revealed to third parties (i.e., inspection), which statements have to be consented to (i.e., message signatures), and how often the credentials may be used in total (i.e., consumption control). The core properties of CARL are technology independence, privacy preservation, support for multi-credential and cross-credential predicates, and its intuitive natural-language-like syntax. While other languages exist that have some of these properties [31, Section 2], CARL supports of all of them simultaneously.

The language is technology independent in the sense that it hides all specifics of the technology underlying the employed credentials. Thus, it allows for expressing authentication requirements without having to worry about implementation details. Although the main goal of CARL is to support the Idemix and U-Prove credential technologies, it is equally suitable for technologies that were designed without privacy considerations.

The privacy-preserving aspect of CARL is due to its support for the principle of data minimization. As described in Chapter 1, data-minimizing authentication requires a reference for determining minimality. CARL policies provide such reference by stating which information has to be minimally disclosed. This is done by selectively listing the attributes that have to be revealed on the one hand, and by specifying the properties that attributes have to fulfill as logical predicate. Such predicate is a Boolean expression that allows for applying logical, comparison, and arithmetic operators to attributes, constants, and sub-expressions. This includes predicates involving attributes from different credentials. The most important aspect in terms of data minimization and privacy preservation is, however, that predicates may also involve attributes that are not revealed. Thus, a user who convinces a service provider that a policy is fulfilled only discloses the fact *that* the attribute predicate is fulfilled, but not the underlying attributes' values. While the support for attribute predicates enables users to remain anonymous when authenticating, the resulting risk for service providers is acknowledged by the possibility to revoke the anonymity under well defined user-agreed conditions and thus make the authentication transaction accountable. This possibility is ensured by requiring the disclosure of verifiably encrypted strongly identifying attributes to a trusted third party that decrypts the attributes in case of misuse (see Section 2.3.4).

Authentication requirements in CARL are expressed in natural-language-like terms. This makes the language very intuitive and easy to use. For example, the policy for applying for a restrictive premium health insurance could be: *own c of-type BodyMass issued-by Physician or Hospital where 18≤c.BMI and c.BMI≤25 and c.issuanceDate>2012/04/01.* Readers can easily grasp that this policy requires users to have a current body mass index (BMI) between eighteen and twenty-five according to a physician or a hospital. An advantage of the human-readable form is that, at least for simple policies, it can be displayed to users directly or with little post-processing, whereas other languages need more sophisticated post-processing steps to make it suitable for display.

Claim Language

In Bichsel et al. [9], see page 129, we present a *claim language* on the basis of CARL. In general, the language has the same features as CARL, but rather than being used by service providers to express authentication requirements, it is intended for users to make statements about their credentials—which are supported by cryptographic evidence—in order to fulfill given authentication requirements. In particular, with the language a user can express which credentials she owns, which predicate holds over the credentials' attributes, which attributes are revealed to trusted third parties, which statement is signed, and what the cleartext values of selected attributes are. For example, a user could state that her salary is in a certain range with respect to a salary statement of her employer. Stated claims are used in two ways. For users, they serve as technology-independent input to software modules that generate technology-specific cryptographic tokens. For service providers, they serve as basis for verifying a cryptographic token. In addition to defining a generic claim language on the basis of CARL, we also extend the Idemix proof specification language to make it as expressive as our claim language. Although languages such as SAML [72] exist that allow for exchanging the cleartext values of a user's attributes, they lack expressivity for privacy-preserving features such as attribute properties. In Sections 4.2 and 4.3 we describe our contributions regarding the verification of claims that describe cryptographic tokens.

Language Framework

In Camenisch et al. [21], see page 93, we build on our previous results in the language space and expand them to an entire language framework, that is, a set of coherent languages, for privacy-preserving attribute-based authentication systems. On the one hand, we broaden our system view from one that focuses on authentication transactions to a holistic one that covers the full life cycle of attribute-based credentials including issuance and revocation. On the other hand, we enhance the set of supported features with *pseudonyms* and *key binding* such that all privacy-preserving features of anonymous credential systems (see Chapter 2) are covered.

More precisely, we revise the existing authentication and claim languages to support revocation, pseudonyms, and key binding and subsequently call them *presentation policy* and *presentation token* languages. The introduced support for credential revocation, however, mainly affects the system semantics which is described in Section 4.2. The framework's support for pseudonyms enables users to authenticate pseudonymously on the one hand and service providers to require the use of previously established pseudonyms on the other hand. To prevent users from sharing their credentials and thus abusing the system, service providers can express that credentials or pseudonyms must have the same underlying secret key as other credentials or pseudonyms.

Further, we provide technology-independent and cryptography-agnostic policy and token languages that are concerned with the *issuance* process of attribute-based credentials. The issuance policies can express which credentials and pseudonyms have to be presented prior to the issuance of a new credential and what the details of this new credential (that is to be issued) are. These details include, for example, the new credential's type, which attribute values are carried over from existing credentials, to which key the new credential is bound, and which attributes are assigned jointly-generated random values. We also provide specification languages for credential types, issuer parameters, and revocation authority parameters that describe technical details necessary for performing the individual transactions. Summarized, the framework includes issuance and presentation policy languages [31, 21] as well as issuance and presentation token [9, 21] languages. The policy languages are used for describing the requirements on the cryptographic token that has to be presented to a server during an issuance or an authentication transaction, respectively. The token languages are used for describing the content of cryptographic presentation tokens and issuance tokens, respectively.

The language framework acts as *control layer* for anonymous credential systems as the individual languages direct the cryptographic protocols underlying these systems. Such layer is crucial in making data-minimizing authentication usable for application developers who cannot be expected to know the details of the underlying cryptographic mechanisms.

A further contribution is that our language framework is unifying the privacy-preserving cryptographic features and concepts that underlie the two premier anonymous credential system implementations, that is, Idemix and U-Prove, by employing a common terminology and offering common interfaces. Such unification is especially relevant for facilitating the adoption of data-minimizing authentication because this allows all system entities to freely choose or even change their preferred implementation. This obviates a frustration of users caused by incompatibilities between their credentials and their preferred service providers' systems. This shall prevent situations where users have privacy-preserving anonymous credentials, but can only use them with a subset of their preferred privacy-enabled service providers. The adoption of data-minimizing authentication has proved to be difficult in the past and thus must be advocated by joint forces of the system vendors, rather than be hindered by incompatibility issues.

4.2 Formal System Semantics

A language without semantics is just an empty shell of syntax without meaning and can thus not be subject to unambiguous interpretation. To precisely specify the meaning of the languages that we have developed, we provide a formal semantics that mathematically defines the various language features as well as the intended behavior of the overall system. Without such formal semantics, an unambiguous implementation of the system would be impossible and different implementations would most likely be incompatible.

In Camenisch et al. [31], see page 193, we provide a formal semantics and a type inference system for CARL. We describe the semantics by means of a state transition system that assumes transitions for credential issuance, token presentation, credential revocation, and token inspection. However, the initial focus is on authentication transactions, that is, token presentation transitions. In our model, users maintain credential repositories that contain the credentials that were issued to them. Further, credential issuers and token verifiers maintain knowledge states about their communication partners in the form of logical predicates. To have a certain predicate in one's knowledge state expresses that a logical statement over the communication partner's credentials was true at the time of the transition. Rather than describing *how* a transition is performed, we provide the semantics in terms of the conditions under which the transition can be performed as well as the effects that the transition has on the knowledge states of the involved parties. In particular, we specify the conditions under which a user can prove a given presentation token, and we specify the server's knowledge increase when the token is proved. The fact that we formulate the semantics in terms of the effects on the knowledge of involved parties is particularly beneficial for understanding the privacy aspects of our system. This is because regarding privacy protection, the main interest is on disclosing as little information as possible while still achieving one's intended goal, that is, the focus is on the increase of knowledge of a service provider.

In Camenisch et al. [21], see page 93, we enhance the previously introduced semantics in three ways. First, we extend the set of the supported features with pseudonyms and key binding, and we provide a revised version of the token presentation semantics accordingly.

Second, we also formally specify the remaining state transitions, that is, credential issuance, credential revocation, and token inspection. Thus, we cover the complete set of privacy-preserving features and the entire credential life cycle. In particular, we detail how a user's credential portfolio is augmented in issuance transactions for basic and for advanced issuance policies, and we show the corresponding knowledge increase of the issuer. In basic issuance transactions, all attributes of the new credentials are determined by the issuer. In advanced issuance transactions, credentials may be bound to keys underlying existing credentials or pseudonyms, and attribute values may be carried over from attributes of existing credentials or set to jointly generated random values, without the issuer learning the values. Further, we formalize two variants of revocation semantics, where one is driven by credential issuers and the other by revocation authorities. Then, we elaborate on what happens to an involved party's knowledge when decrypting verifiably encrypted attributes during inspection.

Finally, we formalize how to determine whether a presentation policy is fulfilled by a proven presentation token, that is, we formalize policy verification. While the formalized approach considers a single token for the verification, we also elaborate on a knowledge-based verification variant that additionally considers the verifier's accumulated knowledge gained in previous token presentation transactions. The latter allows users to prove just parts of a policy and still authenticate successfully. Note that we give an initial informal textual description of the basic policy verification requirements in Bichsel et al. [9].

We see our formal semantics as consolidated didactical reference for conveying the various privacy-preserving features of anonymous credentials to laymen in cryptography. This specifically includes application developers who have the task to implement a data-minimizing authentication system on the basis of a cryptographic library such as Idemix, and who are not familiar with the specifics of the underlying cryptographic protocols. We are not aware of any other work that discusses these features on an abstract level in such a comprehensive way.

4.3 System Components for Data-Minimizing Authentication

The languages and their well-defined semantics described in the two preceding sections are important components in a data-minimizing authentication system. For implementing a functioning system, however, adequate processing algorithms that generate and verify technology-specific token descriptions and the corresponding cryptographic evidence, as well as a token selection mechanism are required (see Chapter 3). Additionally, a communication protocol specifying the interaction between the components as well as the involved messages is needed.

Architecture and Components

In Camenisch et al. [31], see page 193, we present a system architecture that captures all the components that are necessary to perform a data-minimizing authentication transaction. By means of a sequence diagram we illustrate how the components operate with each other, the messages exchanged between them, and the order in which these messages occur.

In Bichsel et al. [9], see page 129, we close the gap between high-level authentication languages and low-level technology-dependent specification languages that are used for generating and verifying cryptographic evidence.

In particular, we show how to determine the claims—which are also called presentation token descriptions—that a user can make with respect to a given credential repository, so as to fulfill a given presentation policy. We explain that an algorithm that generates token descriptions has to be tailored to the capabilities of the technology underlying the employed credentials, that is, the restrictions of the used technology implementation must be accounted for, and we give such algorithms for Idemix and U-Prove.

Further, we provide detailed instructions on how to transform a token description into a semantically equivalent evidence specification. We do this for *Idemix proof specifications* and *U-Prove token specifications*. Such evidence specifications are crucial for performing data-minimizing authentication in that they act as input to the respective cryptographic libraries both to produce token evidence and to verify it. Thus, on the one hand, with these algorithms we provide the necessary mechanism for generating token evidence that is semantically equivalent to a given token description, that is, evidence that supports the statements made in the token description. On the other hand, we provide the basis for verifying the validity of token evidence, that is, whether a given evidence is indeed semantically equivalent to a given token description. While the Idemix library implementation supports—apart from disjunctive statement—all concepts expressible with our CARL-based claim language, the Idemix proof specification language cannot address some of these concepts. Therefore, we also show how the proof specification language of Idemix can be extended with the missing expressivity.

For a service provider to verify whether a given presentation policy is fulfilled by a given presentation token, in addition to checking the validity of the token—that is, whether the evidence corresponds to the description—it also must be verified whether the statements made in the token description fulfill the policy. Therefore, we also informally specify under which conditions a CARL-based token-description fulfills a given CARL policy. Note that we describe policy verification in a formal manner in Camenisch et al. [21].

Implementation

We show that our ideas and concepts are feasible and efficiently realizable by providing an open-source implementation of all components of our data-minimizing authentication system (see Chapter 3). These components comprise: the CARL policy language and the CARL-based token description language including adequate language parsers, a pre-evaluation function for policies, Idemix token description generation, Idemix token evidence generation, Idemix token verification, and policy verification. Additionally, we provide a policy editor user interface that provides meaningful error messages for authoring correct CARL policies. In particular, we developed a modular *credential-based authentication framework* that provides application developers with a toolbox for building data-minimizing authentication solutions. While the framework is open to be instantiated with any attribute-based credential technology, we provide an Idemix instantiation that supports the following features: proof of ownership, selective disclosure and attribute predicate proofs. On top of the framework, we developed a prototype implementation [11] with a graphical user interface that on the one hand allows for selecting the preferred token that fulfills a given policy, and on the other hand clearly informs a user about which information he is about to send to a service provider. See Section 3.6.2 for details on runtime evaluation of the framework and the prototype.

4.4 Data Minimization with Industry Standards

To facilitate the integration of our data-minimizing concepts with existing infrastructure and consequently drive and accelerate their industry acceptance, we investigated whether existing industry standards such as the Security Assertion Markup Language (SAML) and the eXtensible Access Control Markup Language (XACML) can be extended to support our concepts. SAML is a framework for exchanging certified attribute values, and XACML defines an access control policy language as well as a processing model for evaluating policies with respect to a given access request, which assumes that an enforcement point knows all the requester's attribute values. The results of our efforts are a conference publication [4] as well as an OASIS standardization document [70].

In Ardagna et al. [4], see page 177, we show that authorization decisions can be made in a privacy-preserving manner with above-mentioned standards if one is willing to accept certain language and architecture modifications on the one hand, and an additional communication round-trip involving the disclosure of the authorization policy applicable to the access request on the other hand. Firstly, we describe how SAML, which inherently supports the concept of credentials, can be extended with support for expressing logical predicates over a subject's attributes. Then, we elaborate on the necessary modifications to the XACML language, architecture, and policy evaluation model for support of the credential-based setting and the incoming SAML attribute predicates. In our proposed privacy-preserving setting, SAML acts as presentation token language and the XACML language acts as credential-based presentation policy language. An important aspect of the XACML extensions with respect to standardization is that they are fully transparent, that is, an extended XACML authorization system can handle both standard requests and privacy-preserving requests.

Further, we proposed our privacy-preserving SAML extensions, which are solely based on standard SAML extension points, in the form of a *SAML V2.0 Attribute Predicate Profile* [70] to the OASIS Security Services Technical Committee (SSTC)—the standardization body responsible for SAML. The profile was promoted by the SSTC, to the status of *Committee Specification*, which is the last step before becoming an OASIS standard. Our Attribute Predicate Profile is useful in settings where a user's identity attributes are stored and controlled by the (online) identity provider. In particular, our modifications allow identity providers to confirm or deny that a certain predicate over a user's attributes holds without disclosing the exact values of these attributes. That is, identity providers are now enabled to provide (identity) information in terms of Boolean answers to the relying party's yes-no questions, rather than disclosing full attribute values or even full data sets. This offers advantages for identity providers as well as their customers. Identity providers have a significant business advantage as they retain control over the valuable user attributes they have vetted thoroughly. Customers have a privacy advantage as less personal information is revealed to third parties.

4.5 Language for Controlling Downstream Data Usage

The majority of our publications is concerned with authentication techniques that disclose only the information strictly necessary in a given context. However, even in such data-minimizing authentication scenarios, users eventually have to disclose information about themselves. Accordingly, the field of usage control seeks to protect information *after* it has been disclosed.

In Bussard et al. [18], see page 157, we extended the state of the art in usage control with a two-sided XML-based policy language where users specify their usage control preferences on the one hand, and service providers specify how they intend to treat the requested data on the other hand. We further provide a matching semantics, that is, rules that determine whether a user's preferences are in agreement with a server's policy. The result of such matching is a *sticky policy* that travels with the disclosed data item and governs its use. An important and novel aspect of our language is its capability to express *downstream* usage requirements, that is, restrictions for controlling with whom and under what conditions a data consumer may share the disclosed data with further data consumers and how those data consumers treat the data. In particular, users can specify the exact path their data is allowed to follow as well as the restrictions that apply at each hop, and data consumers can specify with whom they intend to share the data, and how the downstream consumers intend to treat the data. Our usage control language may also be integrated into the data-minimizing authentication system that we propose. In fact, the authentication requirements language that we propose in Camenisch et al. [31], see page 193, provides a placeholder for a policy that governs the intended usage of a requested attribute.

Chapter 5

Conclusion and Open Problems

Having provided a general introduction to data-minimizing authentication systems in the preceding chapters, in this final chapter we first draw conclusions. Afterwards, we elaborate on remaining open problems, which at the same time gives possible directions for future research.

5.1 Conclusion

Authentication as it is performed today causes various problems in terms of security and privacy. This is because it involves the use and disclosure of personal information: on the one hand, this information can be stolen and used for identity theft, and, on the other hand, the disclosure can lead to linkage and monitoring of users' behavior and their identification. In this work, we presented a novel authentication system on the basis of anonymous credentials that was designed to preserve its users' privacy. Our system allows service providers to formulate authentication policies in terms of properties that users' attributes must satisfy, and it allows users to prove the mere fact that their certified attributes fulfill these properties without having to disclose any personal information. Such a *data-minimizing* authentication system provides advantages for both users and service providers: users benefit from significantly increased privacy and service providers profit from reducing the risks associated with holding large sets of sensitive personal information.

To facilitate the deployment of privacy-preserving authentication solutions, we implemented an open-source authentication framework that provides all the necessary components for rendering data-minimizing authentication decisions on the basis of attribute-based credentials in general—and Idemix anonymous credentials in particular. On top of this framework, we developed a full-fledged prototype implementation of a data-minimizing authentication system. Our evaluation results of the prototype show that the computational overhead caused by the framework's generation and verification of authentication token descriptions is low compared to the time needed for generating and verifying cryptographic evidence that supports the claims made in the token descriptions.

We are aware that the use of our technology is not a panacea and can only solve parts of the privacy issues arising from the use of electronic media and computing infrastructure. In particular, identifiers that are implicitly disclosed when using electronic communication networks (Internet protocol (IP) addresses, device fingerprints, for example) are challenges in situations where full anonymity is required. However, they can be addressed to a certain extent with additional anonymization techniques [34] such as proxies or mix networks [55, 48]. Further, the use of data-minimizing authentication is obviously not helpful for services where users voluntarily and purposely disclose sensitive personal information about themselves—such as in electronic social networks. Because there is a multitude of services where privacy and unidentifiability are desired (for example, for obtaining information about health issues), we are convinced that our system is a valuable contribution in the field of privacy protection and a very important step in the right direction.

Having shown that data-minimizing authentication systems are technically feasible and efficiently realizable, the next step must be to drive the adoption of this technology. A voluntary adoption is mainly dependent on the goodwill of the service providers as they are in charge of (re-)formulating their service policies such that only the minimally necessary information for the given context (for example, proofs that involve only attribute properties rather than disclosure of attribute values) is demanded. Such voluntary adoption, however, will not be reachable by simply raising the service providers' awareness of the issues associated with collection of personal data and of the availability of technology that mitigates these issues. Because "the measure of information security and data privacy are dollars" [25], service providers must be convinced with compelling economical arguments that it is more profitable to protect users' privacy than to put it at stake. Given just the high costs associated with data breaches (see Section 1.1.2), we expect that such arguments can be made with further scientific studies.

An alternative adoption scenario includes regulatory frameworks governing that service providers must collect only the data that is minimally necessary in a given context. In fact, recent political developments indicate that governments are taking privacy issues more and more seriously and willing to take the necessary steps to change the status quo on how citizens' privacy is treated by enterprises. For example, in February 2012 the administration of US President Barack Obama unveiled a framework for protecting privacy in the digital age, at which center stands a *Consumer Privacy Bill of Rights* [83]. This bill sets forth seven basic rights for consumers in the commercial Internet. Among these proposed rights, the following are of high relevance for this work: *Individual Control*—consumers have a right to exercise control over what personal data companies collect from them; *Respect for Context*—consumers have a right to expect that companies will collect personal data in ways that are consistent with the context in which consumers provide the data; and *Focused Collection*—consumers have a right to reasonable limits on the personal data that companies collect and retain. According to the presidential administration, its intention is to encourage stakeholders to implement this bill and to put it into law. The passage of such a bill would effectively make the use of privacy-preserving technologies—such as the data-minimizing authentication systems that are proposed in this thesis—mandatory.

5.2 Open Problems

In the following, we elaborate on remaining open issues. First, we discuss general technical challenges. Then, we point out issues that are concerned with the acceptance and deployment of data-minimizing authentication systems. Finally, we describe challenges related to awareness and education.

General Technical Challenges

- The current version of the Idemix [33, 58] library (version 2.3.3) has only partial support for the general feature set that anonymous credential systems offer (see Chapter 2). In particular, the library lacks support for disjunctive attribute property proofs [9] and for credential revocation. For a real-world adoption, however, the availability of these features is crucial.

- Our credential-based authentication framework [9] can merely transform a subset of the CARL claim language to Idemix evidence specifications, that is, attribute disclosure to third parties as well as message signatures are not supported. Given our revised and more comprehensive approach on expressing claims by means of an XML-based presentation token language [21] that also has expressivity for revocation, pseudonyms and key binding, these transformation algorithms must be accordingly adapted to this new language. At the same time, the transformation capabilities should be extended for supporting also the U-Prove credential system as well as certificate technologies such as X.509.

- Although we have mathematically defined the behavior of data-minimizing authentication systems in terms of a formal language semantics [21], it remains open to prove that existing implementations are compliant with this semantics. Such proof would entail that only the information that is specified in the issuance and presentation token description is disclosed in issuance and presentation transactions, respectively, and that the underlying credential system is implemented as intended.

- The work of Wästlund et al. [89] was an important starting point for designing user interfaces (UIs) that match the mental models of users in association with anonymous credentials. However, up to now, they have investigated these models only for proofs of ownership, selective attribute disclosure, and simple property proofs. Accordingly, more research is needed that addresses the entire feature set of anonymous credentials. Additionally, it is still unclear how such UIs have to be designed for mobile devices such as smartphones which have significantly smaller screens than stationary desktop computers.

- To support enterprises in policy authoring and to tap the full potential of data-minimizing authentication, clear guidelines defining what information is minimally necessary for certain contexts must be available. To our knowledge, no such guidelines exist, however. These guidelines should be specified by independent data protection institutions for standard authentication use cases such as online shopping with and without physical good delivery.

- The GUI in our prototype implementation informs a user about which information she is about to disclose to a service provider (see Section 3.4.3). On the one hand, further UI research must be conducted to find ways to also inform users about whether the information they are about to disclose is minimal with respect to the authentication policy, and whether the authentication policy itself is minimal with respect to standardized policies for the use case at hand. On the other hand, is must be studied how such GUIs can be extended to entire identity management solutions that keep track of which data and which pseudonyms were disclosed to whom at which points in time, such that this knowledge can be considered for determining what the most privacy-friendly claim is that a user can make to be granted access to some service.

- Our usage control language for expressing which disclosed personal data can be shared with whom under what conditions [18] is an important starting point in governing downstream usage control. The bigger challenge, however, is the enforcement of usage control policies within and across computing platforms. Although there are first results in this respect from Pretschner et al. [57, 61], usage control enforcement remains a very challenging task. Another issue with respect to usage control is that research has so far focused only on how to control the usage of disclosed attribute values. An interesting open issue, however, is how to treat attribute properties once they have been disclosed.

- For a real-world adoption of data-minimizing authentication solutions, it must be investigated how to build these solutions such that they are secure in various aspects: secure credential storage and management, secure credential processing algorithms, secure display of information that is about to be disclosed (to prevent that information displayed on the computer screen is maliciously altered), and trusted libraries that generate cryptographic evidence (to prevent that the cryptographic modules disclose more data than confirmed by the user).

Acceptance and Deployment Challenges

- Given the current trend towards the use of mobile computing devices, it is crucial to study how users can authenticate with the help of such devices in a data-minimizing manner. An interesting question related to this is how credentials can be securely stored on such devices (current credential storage approaches typically involve smart cards as storage medium) without them being compromisable by smartphone viruses, device loss, or device theft.

- An open issue that is related to the deployment of data-minimizing authentication systems is user acceptance. In our view, reaching user acceptance can only be achieved by solutions that focus on convenience. Given that electronic social network providers (Facebook, for example) act more and more as identity providers in the sense that other service providers can rely on the attributes kept in the social network account (for example, Facebook Connect), it became highly convenient for users to authenticate with the help of their social network provider. Thus, for a successful deployment of data-minimizing authentication solutions with anonymous credentials, ways have to be found for making the new solutions as unobtrusive and convenient as possible such that they can compete with the convenience of existing ones.

- The use of data-minimizing authentication solutions with anonymous credentials inherently involves the need to securely store and access these credentials. The naive approach for enabling this is to host the credentials on a trusted user-controlled device. Users nowadays have a number of devices and also use devices that they do not control themselves, yet users want to authenticate in a convenient manner from all of these devices, it remains unclear how this can be achieved. The use of cloud services in combination with sophisticated cryptographic protocols allowing users to securely access their credentials may be a possible path to explore in this respect.

- To avoid that incompatibilities between different anonymous credential system implementations inhibit the deployment and acceptance of data-minimizing authentication, the availability of unifying cryptographic protocols that allow for the generation of cryptographic evidence that involves credentials from different implementations would be beneficial. Such protocols would allow users to fulfill policies that require multiple credentials with a combination of, for example, Idemix and U-Prove credentials.

- One of the biggest open issues is the question how service providers can be motivated to accept and deploy data-minimizing authentication solutions.

Different scenarios range from voluntary adoption, to forced adoption by regulatory frameworks, to the regulatory power of the free market. As a voluntary adoption will only happen if it entails a clear financial motivation, changes in legislation may provide such motivation in the form of significant fines for data breaches involving sensitive personal data—because currently the risk, that is, the costs, associated with such data breaches are apparently too low.

- To let the free market solve the acceptance issue of privacy-preserving technologies, new innovative business models must be developed that can monetize online privacy. Alternatively, companies may come up with marketing strategies that promote the use of data-minimizing techniques as competitive advantage—similar to how this is currently done with *green* technology—in the hope that users give precedence to service providers who do not spurn the foundational right of their users but employ authentication technologies that let users retain control over their own data. The ultimate (philosophical) question is, what users are willing to pay for their privacy.
- Without a doubt, anonymous credentials are the ideal technology for electronic versions of national identity cards issued by governments. On the one hand, governments (mostly) have a constitutional interest in protecting the privacy of their citizens. On the other hand, identity providers could use such government-issued credentials to bootstrap the issuance of their own credentials, which effectively makes their credentials more trustworthy. However, as governments have not adopted such approach yet, the exact causes of this lack of adoption are yet to be determined and to be tackled.

Awareness and Education Challenges

- Human being's perception of their interactions with information technology is distorted in the sense that they unconsciously apply the rules of social interaction to their interaction with computers. These rules include that other human beings tend to not disclose information that is very personal, that they gradually forget or lose track of information that they did not use for some time, and that they are able to distinguish right from wrong [14]. This is one of the reasons why, for example, it became the norm for teenagers to share sensitive personal information on social network platforms without actively controlling the access to this information [84, 85]. Accordingly, we must develop and integrate awareness features into information and communication technology (ICT) that alert users to the possible security and privacy threats as well as provide early education of users to understand ICT in the appropriate ways [14]. A positive and innovative example of a general effort to create such awareness is provided by a group of privacy enthusiasts who met the subject with a portion of irony: they created an online game [6] where users deal with personal data: user profiles must be collected, linked, and sold on a virtual black market. The more sensitive the handled data is, the more profitable is their fictional business.

- The possibilities opened up by the advanced features of anonymous credential systems are counterintuitive and seemingly contradictory in that users can prove attribute properties without disclosing the attributes' values on the one hand and act anonymous yet accountable on the other hand. Further, these features are incompatible with the way users think about the (non-anonymous) credentials they use today: the plastic cards in their wallets are used in an *all-or-nothing* fashion where users either show these cards with all the information they contain or they do not show them at all. In the physical world, users do not have the opportunity of merely proving logical statements about the information certified on their cards, and it is not trivial to understand that this possibility exists at all, let alone to understand how this can be done technically. Consequently, we must find ways to raise awareness that modern technology has these capabilities and to properly educate individuals about the positive security and privacy implications that the use of this technology has.

Bibliography

[1] ABC4Trust—Attribute-based Credentials for Trust. `https://abc4trust.eu/`, May 2012.

[2] Application Security, Inc. `http://www.appsecinc.com/`, May 2012.

[3] Claudio A. Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. Exploiting cryptography for privacy-enhanced access control. *Journal of Computer Security*, 18(1):123–160, 2010.

[4] 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.

[5] Giuseppe Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 138–146, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.

[6] Ivan Averintsev, Wolfie Christl, Pascale Osterwalder, and Ralf Traunsteiner. Data dealer online game. `http://www.datadealer.net/english`, May 2012.

[7] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

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

[9] 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, Illinois, USA, November 2011. ACM Press.

[10] 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.

[11] Patrik Bichsel and Franz-Stefan Preiss. Demo: A comprehensive framework enabling data-minimizing authentication. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *18th Conference on Computer and Communications Security*, pages 733–736, Chicago, Illinois, USA, October 17–21, 2011. ACM.

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

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

[14] Katrin Borcea-Pfitzmann, Andreas Pfitzmann, and Manuela Berg. Privacy 3.0 := Data Minimization + User Control + Contextual Integrity. *it - Information Technology*, 53(1):34–40, 2011.

[15] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[16] 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, *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415, Townsville, Australia, July 2–4, 2007. Springer, Berlin, Germany.

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

[18] Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Downstream usage control. In *Proc. of the 11th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 22–29, Fairfax, Virginia, USA, July 2010. IEEE Computer Society Press.

[19] Tony Busseri. It's time to take cybersecurity seriously. `http://www.wired.com/threatlevel/2012/03/opinion-busseri-cybersecurity/`, March 2012.

[20] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345, Kyoto, Japan, December 3–7, 2000. Springer, Berlin, Germany.

[21] Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. A language framework for privacy-preserving attribute-based authentication. *In submission.*

[22] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09: 16th Conference on Computer and Communications Security*, pages 131–140, Chicago, Illinois, USA, November 9–13, 2009. ACM Press.

[23] Jan Camenisch, Maria Dubovitskaya, Gregory Neven, and Gregory M. Zaverucha. Oblivious transfer with hidden access control policies. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 192–209, Taormina, Italy, March 6–9, 2011. Springer, Berlin, Germany.

[24] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[25] Jan Camenisch and Günter Karjoth. E-privacy—privacy in the electronic society, February 2011. Lecture Notes.

[26] 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, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443

of *Lecture Notes in Computer Science*, pages 481–500, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

[27] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. Solving revocation with efficient update of anonymous credentials. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 454–471, Amalfi, Italy, September 13–15, 2010. Springer, Berlin, Germany.

[28] 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.

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

[30] 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.

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

[32] 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 17–21, 2003. Springer, Berlin, Germany.

[33] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. IBM Research Report RZ 3419, IBM Research Division, May 2002.

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

[35] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.

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

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

[38] 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.

[39] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Berlin, Germany.

[40] Lidong Chen. Access with pseudonyms. In E. Dawson and J. Golić, editors, *Cryptography: Policy and Algorithms*, volume 1029 of *Lecture Notes in Computer Science*, pages 232–243. Springer, 1995.

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

[42] 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.

[43] 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.

[44] Cybersource. 12th Annual Online Fraud Report: Online Payment Fraud Trends, Merchant Practices and Benchmarks, 2011.

[45] Ivan Bjerre Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In Shafi Goldwasser, editor, *Advances in Cryptology: CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335. Springer, 1990.

[46] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Berlin, Germany.

[47] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[48] 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.

[49] 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'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Germany.

[50] Open Security Foundation. DataLossDB. `http://datalossdb.org/`, May 2012.

[51] The Eclipse Foundation. Rich Ajax Platform (RAP). Enabling modular business apps for desktop, browser and mobile. `http://www.eclipse.org/rap/`, May 2012.

[52] The Eclipse Foundation. Xtext: A framework for development of programming languages and domain specific languages. `http://www.eclipse.org/Xtext/`, May 2012.

[53] PR-COM Gesellschaft für strategische Kommunikation mbH. Projekt Datenschutz. `http://www.projekt-datenschutz.de/`, May 2012.

[54] Gartner. User Survey Analysis: Key Trends Shaping the Future of Data Center Infrastructure Through 2011, October 2010.

[55] 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.

[56] D. Hardt, J. Bufu, and J. Hoyt. OpenID attribute exchange 1.0. `http://openid.net/developers/specs/`, December 2007.

[57] Manuel Hilty, Alexander Pretschner, Christian Schaefer, and Thomas Walter. Duke–distributed usage control enforcement. *Policies for Distributed Systems and Networks, IEEE International Workshop on*, 0:275, 2007.

[58] 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`.

[59] Stanisław Jarecki. *Efficient Threshold Cryptosystems.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 2001.

[60] Stephen T. Kent, Lynette I. Millett, and National Research Council (U.S.). *Who Goes There?: Authentication Through the Lens of Privacy.* National Academies Press, 2003.

[61] Prachi Kumari and Alexander Pretschner. Deriving implementation-level policies for usage control enforcement. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, CODASPY '12, pages 83–94, New York, NY, USA, 2012. ACM.

[62] Meglena Kuneva. Keynote speech at roundtable on online data collection, targeting and profiling, March 2009.

[63] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[64] 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.

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

[66] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999: 6th Annual International Workshop on Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*, pages 184–199, Kingston, Ontario, Canada, August 9–10, 2000. Springer, Berlin, Germany.

[67] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[68] 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, *PKC 2009: 12th International*

Conference on Theory and Practice of Public Key Cryptography, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

[69] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy*, pages 111–125, Oakland, California, USA, May 18–21, 2008. IEEE Computer Society Press.

[70] Gregory Neven and Franz-Stefan Preiss. SAML V2.0 Attribute Predicate Profile Version 1.0. Committee Specification 01, Organization for the Advancement of Structured Information Standards (OASIS), November 2011.

[71] 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 14–18, 2005. Springer, Berlin, Germany.

[72] 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.

[73] Christian Paquin. U-Prove cryptographic specification V1.1. Technical report, Microsoft Corporation, February 2011.

[74] Ponemon Institute. `http://www.ponemon.org`, May 2012.

[75] PrimeLife—Privacy and Identity Management for Life. `http://www.primelife.eu/`, May 2012.

[76] 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 (rump session). In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[77] 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.

[78] 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.

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

[80] Daniel J. Solove. 'I've Got Nothing to Hide' and Other Misunderstandings of Privacy. *San Diego Law Review*, 44(289):745, 2007.

[81] Symantec and Norton. The shocking scale of cybercrime. `http://www.symantec.com/content/en/us/home_homeoffice/html/cybercrimereport/`, September 2011.

[82] The Data Warehousing Institute (TDWI). Taking Data Quality to the Enterprise through Data Governance, April 2006.

[83] The White House. Consumer Data Privacy In A Networked World: A Framework For Protecting Privacy And Promoting Innovation In The Global Digital Economy, February 2012.

[84] Bibi van den Berg and Ronald Leenes. Audience segregation in social network sites. *Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust, 2010 IEEE International Conference on*, 0:1111–1116, 2010.

[85] 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.

[86] Verizon. Data breach investigations reports, 2011 and 2012.

[87] Kristof Verslype. *Improving Privacy in Applications by Managing the Disclosed Personal Properties.* PhD thesis, KU Leuven, Belgium, Leuven, Belgium, March 2011.

[88] Markus Voelter. xtext-typesystem: A type system framework for Xtext. `http://code.google.com/a/eclipselabs.org/p/xtext-typesystem/`, May 2012.

[89] 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.

[90] Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *2010 IEEE Symposium on Security and Privacy*, pages 223–238, Berkeley/Oakland, California, USA, May 16–19, 2010. IEEE Computer Society Press.

Appendix A

Prototype Details

A.1 Example CARL policy

The claim selection GUI displayed in Figure 3.3 was rendered to fulfill the following CARL policy:

```
own mc of-type [MemberShipCard] issued-by [CarRentalCo.com]
own cc of-type [CreditCard] issued-by [Amex.com], [Visa.com]
own dl of-type [DriversLicense] issued-by [DeptMotorVehicles.com]
reveal cc.cardNumber
where dl.vehicleCategoryB==true and
      (mc.status=="gold" or
      mc.status=="silver") and
      cc.expirationDate>2012-05-07 and
      mc.givenName==dl.firstName and
      mc.surname==dl.lastName
```

A.2 Test Data for Time Measurement

In this section we list the test data that was used for measuring the performance of the prototype implementation.

A.2.1 Credential Repository

The following 6 credentials were in the credential repository. Note that the attributes' values are just example data and do not reflect reality.

- Hertz Membership Card:
  Type=MemberShipCard, Issuer=CarRentalCo.com,
  givenName=Franz-Stefan, surname=Preiss, dateOfBirth=1982/05/17, status=gold, expirationDate=2014/03/09

- Private Liability Insurance:
  Type=LiabilityInsurance, Issuer=InsuranceCo.com,
  policyNumber=2009/2M/9876/12345,
  guaranteedUSDAmount=2000000, firstName=Franz-Stefan,
  lastName=Preiss, expirationDate=2015/07/18

- Employer Benefit Insurance:
  Type=LiabilityInsurance, Issuer=EmployerBenefits.com,
  policyNumber=2007/30K/7A57/28,
  guaranteedUSDAmount=30000, firstName=Franz-Stefan, lastName=Preiss,
  expirationDate=2021/07/18

- Private Visa Card:
  Type=CreditCard, Issuer=Visa.com, cardNumber=4711 3702 3910 7228, nameOnCard=PREISS FRANZ-STEFAN, surName=Preiss, cardVerificationValue=845, expirationDate=2013/10/31

- Corporate American Express Card:
  Type=CreditCard, Issuer=Amex.com, cardNumber=5733 030243 24302, nameOnCard=FRANZ-STEFAN PREISS, surName=Preiss, cardVerificationValue=4867, expirationDate=2013/05/31

- Swiss Drivers License:
  Type=DriversLicense, Issuer=DeptMotorVehicles.com, firstName=Franz-Stefan, lastName=Preiss, dateOfBirth=1982/05/17,
  expirationDate=2112/12/08, vehicleCategoryA=false,
  vehicleCategoryAFrom=NA, vehicleCategoryB=true,
  vehicleCategoryBFrom=2002/05/17

A.2.2 Authentication Policy

```
own mc of-type [MemberShipCard] issued-by [CarRentalCo.com]
own cc of-type [CreditCard] issued-by [Amex.com], [Visa.com]
own dl of-type [DriversLicense] issued-by [DeptMotorVehicles.com]
own is of-type [LiabilityInsurance] issued-by [InsuranceCo.com],
                                              [EmployerBenefits.com]
reveal cc.cardNumber, is.policyNumber
where dl.vehicleCategoryB==true and
      is.guaranteedUSDAmount>=30000 and
      (mc.status=="gold" or
      mc.status=="silver") and
      cc.expirationDate>2012-05-07 and
      is.expirationDate>2012-05-07 and
      mc.givenName==dl.firstName and
      mc.surname==dl.lastName and
      dl.firstName==is.firstName and
      dl.lastName==is.lastName
```

A.2.3 Token Description

```
Claim:
i own mc of-type [MemberShipCard] issued-by [CarRentalCo.com]
i own cc of-type [CreditCard] issued-by [Amex.com]
i own dl of-type [DriversLicense] issued-by [DeptMotorVehicles.com]
i own is of-type [LiabilityInsurance] issued-by [EmployerBenefits.com]
where dl.vehicleCategoryB and
      is.guaranteedUSDAmount>=30000 and
      mc.status=="gold" and
      cc.expirationDate>2012-05-07 and
      is.expirationDate>2012-05-07 and
      mc.givenName==dl.firstName and
      mc.surname==dl.lastName and
      dl.firstName==is.firstName and
      dl.lastName==is.lastName
Revealed values:
dl.vehicleCategoryB -> true
cc.cardNumber -> 5733 030243 24302
is.policyNumber -> 2007/30K/7A57/28
mc.status -> gold
```

A.2.4 Idemix Parameters

Idemix System Parameters

```
<?xml version="1.0" encoding="UTF-8"?>
<SystemParameters xmlns="http://www.zurich.ibm.com/security/idemix"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:schemaLocation=
    "http://www.zurich.ibm.com/security/idemix SystemParameters.xsd">
    <Elements>
        <l_e>597</l_e>
        <l_ePrime>120</l_ePrime>
    <l_Gamma>768</l_Gamma>
    <l_H>256</l_H>
    <l_k>160</l_k>
    <l_m>256</l_m>
    <l_n>1024</l_n>
    <l_Phi>80</l_Phi>
    <l_pt>80</l_pt>
    <l_r>80</l_r>
    <l_res>1</l_res>
    <l_rho>256</l_rho>
    <l_v>2048</l_v>
    <l_enc>256</l_enc>
    </Elements>
</SystemParameters>
```

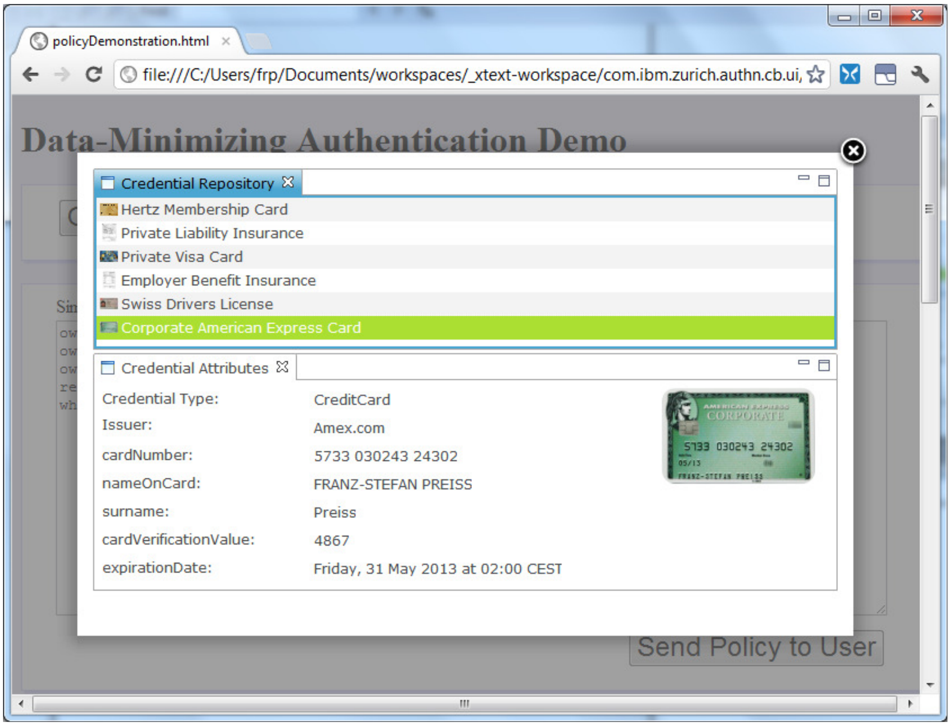
Idemix Group Parameters

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<GroupParameters xmlns="http://www.zurich.ibm.com/security/idemix"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
  "http://www.zurich.ibm.com/security/idemix GroupParameters.xsd">

    <References>
       <SystemParameters>
          http://www.zurich.ibm.com/security/idmx/v2/sp.xml
       </SystemParameters>
    </References>

    <Elements>
        <Gamma>
           9165290133237088109252622858539437514127166392609483973154562
           3682806064687757966976688617296290724878397989803379609106232
           5584064802287728294468642896625540973839483206710301877044439
           99392992864262754084532256387992523359036345695l
        </Gamma>
        <g>
           5449393241448649514864998992383115155669081501945008303794874
           6480571560693553166565029454365555716112029901661254577882483
           6663055752526772831734129015277436123987064573455560044643331
           4243595106518066395157943077569196050298324276 4
        </g>
        <h>
           1489933697667334616115515259305243490263272953907739732035561
           0160369286495793486680379194910037536533127925620182006059541
           5932131561156675272129317873339407328687111761685099261975482
           8830514550763370691462956840758481937392683923 18
        </h>
        <rho>
           7234402063248475505695455931507896457865468817738931859006516
           591926602764845 3
        </rho>
    </Elements>
</GroupParameters>
```

A.3 Screenshots


policyDemonstration.html
file:///C:/Users/frp/Documents/workspaces/_xtext-workspace/com.ibm.zurich.authn.cb.ui,
Data-Minimizing Authentication Demo
Credential Repository
Credential Attributes
Send Policy to User

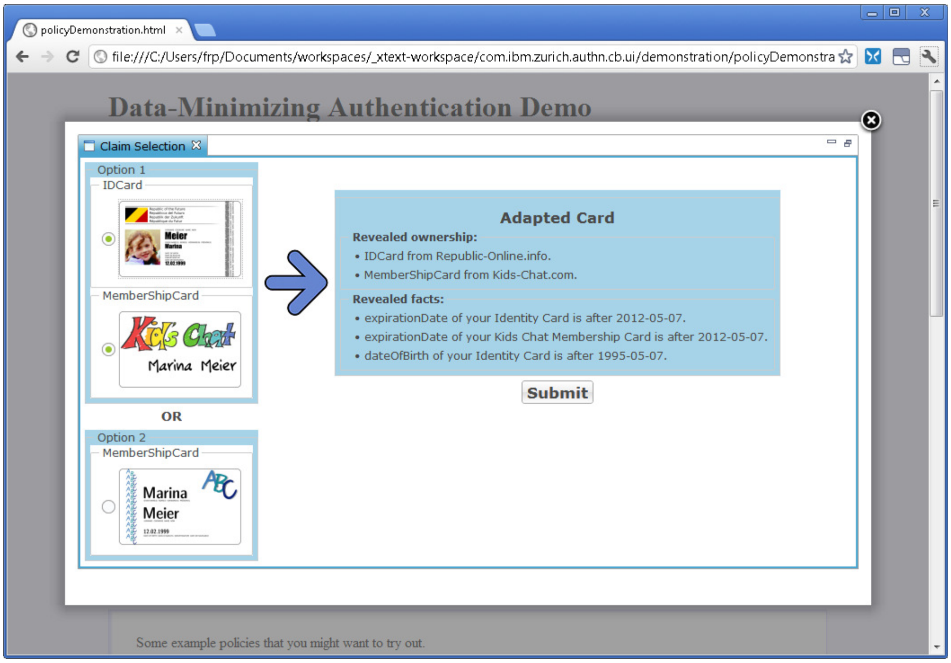
Hertz Membership Card
Private Liability Insurance
Private Visa Card
Employer Benefit Insurance
Swiss Drivers License
Corporate American Express Card

Credential Type:	CreditCard
Issuer:	Amex.com
cardNumber:	5733 030243 24302
nameOnCard:	FRANZ-STEFAN PREISS
surname:	Preiss
cardVerificationValue:	4867
expirationDate:	Friday, 31 May 2013 at 02:00 CEST

Figure A.1: Credential repository browser of prototype application displaying example credit card.

The claim selection GUI shown in Figure A.2 was rendered to fulfill the following disjunctive authentication policy:

```
own id of-type [IDCard] issued-by [Republic-Online.info]
own mc of-type [MemberShipCard] issued-by [Kids-Chat.com]
where id.expirationDate>2012-05-07 and
      mc.expirationDate>2012-05-07 and
      id.dateOfBirth>dateSubtractDuration(2012-05-07,P17Y)
OR
own sc of-type [MemberShipCard] issued-by [School.gov]
where sc.expirationDate>2012-05-07 and
      sc.dateOfBirth>dateSubtractDuration(2012-05-07,P17Y)
```


policyDemonstration.html

file:///C:/Users/frp/Documents/workspaces/_xtext-workspace/com.ibm.zurich.authn.cb.ui/demonstration/policyDemonstra

Data-Minimizing Authentication Demo

Claim Selection

Option 1

IDCard

MemberShipCard

OR

Option 2

MemberShipCard

Adapted Card

Revealed ownership:

- IDCard from Republic-Online.info.
- MemberShipCard from Kids-Chat.com.

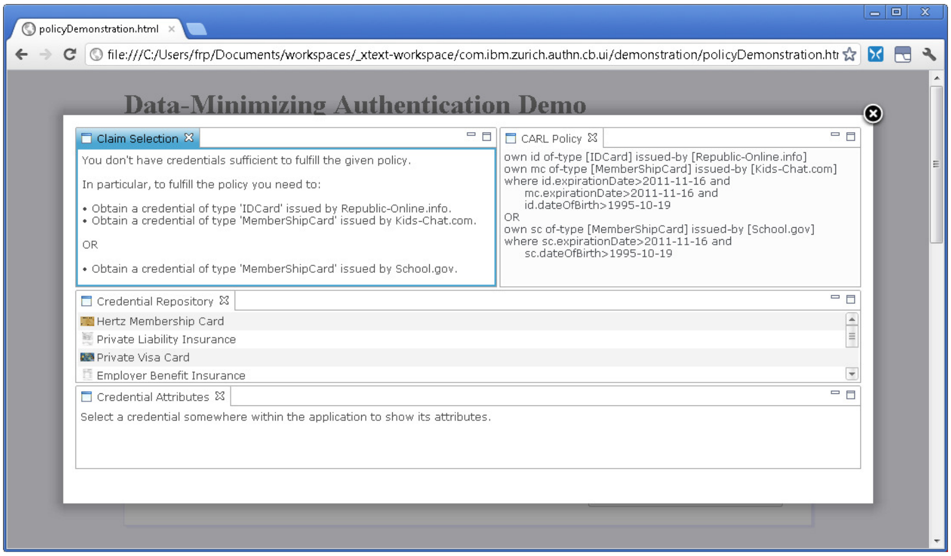
Revealed facts:

- expirationDate of your Identity Card is after 2012-05-07.
- expirationDate of your Kids Chat Membership Card is after 2012-05-07.
- dateOfBirth of your Identity Card is after 1995-05-07.

Submit

Some example policies that you might want to try out.

Figure A.2: Credential selection with two alternative policies.


policyDemonstration.html

file:///C:/Users/frp/Documents/workspaces/_xtext-workspace/com.ibm.zurich.authn.cb.ui/demonstration/policyDemonstration.htı

Data-Minimizing Authentication Demo

Claim Selection

You don't have credentials sufficient to fulfill the given policy.

In particular, to fulfill the policy you need to:

- Obtain a credential of type 'IDCard' issued by Republic-Online.info.
- Obtain a credential of type 'MemberShipCard' issued by Kids-Chat.com.

OR

- Obtain a credential of type 'MemberShipCard' issued by School.gov.

CARL Policy

```
own id of-type [IDCard] issued-by [Republic-Online.info]
own mc of-type [MemberShipCard] issued-by [Kids-Chat.com]
where id.expirationDate>2011-11-16 and
      mc.expirationDate>2011-11-16 and
      id.dateOfBirth>1995-10-19
OR
own sc of-type [MemberShipCard] issued-by [School.gov]
where sc.expirationDate>2011-11-16 and
      sc.dateOfBirth>1995-10-19
```

Credential Repository

- Hertz Membership Card
- Private Liability Insurance
- Private Visa Card
- Employer Benefit Insurance

Credential Attributes

Select a credential somewhere within the application to show its attributes.

Figure A.3: Credential selection where credentials are insufficient to fulfill the policy.

Part II

Publications

List of Publications

Conference Publications

1. Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. A language framework for privacy-preserving attribute-based authentication. *In submission.*

   – See p. 93.

2. 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, Illinois, USA, November 2011. ACM Press.

   – See p. 129.

3. Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Downstream usage control. In *Proc. of the 11th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 22–29, Fairfax, Virginia, USA, July 2010. IEEE Computer Society Press.

   – See p. 157.

4. 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.

   – See p. 177.

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

   – See p. 193.

6. 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.

Conference Demo

1. Patrik Bichsel and Franz-Stefan Preiss. Demo: A comprehensive framework enabling data-minimizing authentication. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *18th Conference on Computer and Communications Security*, pages 733–736, Chicago, Illinois, USA, October 17–21, 2011. ACM.

Book Chapters

1. Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Matching privacy policies and preferences: Access control, obligations, authorisations and downstream usage. In Jan Camenisch, Simone Fischer-Hübner, and Kai Rannenberg, editors, *Privacy and Identity Management for Life*, pages 313–326. Springer, 2011.

2. Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Gregory Neven, Stefano Paraboschi, Eros Pedrini, Franz-Stefan Preiss, Pierangela Samarati, and Mario Verdicchio. Advances in access control policies. In Jan Camenisch, Simone Fischer-Hübner, and Kai Rannenberg, editors, *Privacy and Identity Management for Life*, pages 327–342. Springer, 2011.

3. Jan Camenisch, Benjamin Kellermann, Stefan Köpsell, Stefano Paraboschi, Franz-Stefan Preiss, Stefanie Pötzsch, Dave Raggett, Pierangela Samarati, and Karel Wouters. Open source contributions. In Jan Camenisch, Simone Fischer-Hübner, and Kai Rannenberg, editors, *Privacy and Identity Management for Life*, pages 459–478. Springer, 2011.

Technical Report

1. 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.

Standardization Document

1. Gregory Neven and Franz-Stefan Preiss. SAML V2.0 Attribute Predicate Profile Version 1.0. Committee Specification 01, Organization for the Advancement of Structured Information Standards (OASIS), November 2011.

Patent

1. 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.

Publication

A Language Framework for Privacy-Preserving Attribute-Based Authentication

Publication Data

Jan Camenisch, Maria Dubovitskaya, Anja Lehmann, Gregory Neven, Christian Paquin, and Franz-Stefan Preiss. A language framework for privacy-preserving attribute-based authentication. *In submission.*

Contributions

- Principal author.
- Formal Semantics.

A Language Framework for Privacy-Preserving Attribute-Based Authentication

J. Camenisch[1], M. Dubovitskaya[1], A. Lehmann[1],
G. Neven[1], C. Paquin[2], and F.-S. Preiss[1]

[1] IBM Research – Zurich, Switzerland
[2] Microsoft Research, Redmond, USA

Abstract. Existing cryptographic realizations of privacy-friendly authentication mechanisms such as anonymous credentials, minimal disclosure tokens, self-blindable credentials, and group signatures vary largely in the features they offer and in how these features are realized. Some features such as revocation or de-anonymization even require the combination of several cryptographic protocols. These differences and the complexity of the cryptographic protocols hinder the deployment of these mechanisms for practical applications and also make it almost impossible to switch the underlying cryptographic algorithms once the application has been designed. In this paper, we aim to overcome this issue and simplify both the design and deployment of privacy-friendly authentication mechanisms. We define and unify the concepts and features of privacy-preserving attribute-based credentials (Privacy-ABCs), provide a language framework in XML schema, and give a formal semantics to describe the effects of the transactions in a privacy-friendly authentication system using Privacy-ABCs. Our language framework enables application developers to use Privacy-ABCs with all their features without having to consider the specifics of the underlying cryptographic algorithms—similar to as they do today for digital signatures, where they do not need to worry about the particulars of the RSA and DSA algorithms either.

Keywords: Authentication, privacy, data-minimization, anonymous credentials, digital credentials.

1 Introduction

More and more transactions in our daily life are performed electronically and the security of these transactions is an important concern. Strong authentication and according authorization based on certified attributes of the requester is paramount for protecting critical information and infrastructures online.

Most existing techniques for transferring trusted user attributes cause privacy issues. In systems where an online identity provider creates access tokens on demand, such as SAML, OpenID, or WS-Federation, the identity provider can impersonate its users and can track a user's moves online. Systems with offline token creation, such as X.509 certificates and some WS-Trust profiles, force the user to reveal more attributes than strictly needed (as otherwise the issuer's signature cannot be verified) and make her online transactions linkable across different websites.

These drawbacks can be overcome with privacy-preserving authentication mechanisms based on advanced cryptographic primitives such as anonymous credentials, minimal disclosure tokens, self-blindable credentials, or group signatures [25, 10, 18, 21, 5, 47]. In these schemes, users obtain certified credentials for their attributes from trusted issuers and later derive, without further assistance from any issuer, unlinkable tokens that reveal only the required attribute information yet remain verifiable under the issuer's public key. Well-known examples being Brands' scheme [10] and Camenisch-Lysyanskaya's scheme [18], which have been implemented in Microsoft's U-Prove [26] and IBM's Identity Mixer [24, 32], respectively. Both implementations are freely available and efficient enough for practical use, yet the real-world adoption is slower than one may hope.

One possible reason for the slow adoption of privacy-preserving authentication technologies might be that the various schemes described in the literature have a large set of features where similar features are often called differently or are realized with different cryptographic mechanisms. Many of the features such as credential revocation, efficient attribute encoding, or anonymity lifting even require a combination of separate cryptographic protocols. This makes these technologies hard to understand and compare and, most importantly, very difficult to use.

To overcome this, we provide unified definitions of the concepts and features of the different privacy-preserving authentication mechanisms. We will refer to this unification as *privacy-preserving attribute-based credentials* or *Privacy-ABCs*. Our definitions abstract away from the concrete cryptographic realizations but are carefully crafted so that they can be instantiated with the different cryptographic protocols—or a combination of them. To enable the use and integration of Privacy-ABCs in authentication and authorization systems, we further present cryptography-agnostic definitions of all concepts as well as a language framework with data formats for, e.g., policies and claims. All languages are specified in XML schema and separate the abstract functionality expected from the underlying cryptographic mechanisms from the opaque containers for the cryptographic data itself. Thus, these languages allow application developers to employ Privacy-ABCs without having to think about their cryptographic realization, similarly to how digital signatures or encryption can be used today. The language described in this paper has been implemented in the ABC4Trust project (www.abc4trust.eu) and will be made available as part of a reference implementation of a Privacy-ABC system which will include a number of cryptographic solutions.

Finally, we present a formal semantics that precisely defines the meaning of our comprehensive language and their expressed features. Such a rigorous mathematical description allows to determine, for instance, whether a user can fulfill a given authentication policy with her credential portfolio or whether a derived access token satisfies a policy. As our language covers the entire Privacy-ABC system, we also provide semantics that describe the intended system behaviour, i.e., the effects of state transitions— which are steered by our language—on the different entities and their knowledge states.

2 Related work

Our work is based on the credential-based access control requirements language (CARL) recently proposed by Camenisch et al. [22]. CARL allows a service provider (verifier) to specify which attributes certified by whom a user needs to present in order to get access. Compared to our work, CARL defines only a small part of a Privacy-ABC system, namely the presentation policy, but does not consider how these attributes are transmitted nor how credentials are issued or revoked. Bichsel et al. [6] have extended CARL to cover the transmission of certified attributes. Version 1 of the U-Prove protocols [26] covers credential issuance and presentation but only supports selective attribute disclosure. It does not consider other features such as attribute predicates, inspection, key binding, (cryptographic) pseudonyms, or revocation.

Privacy-ABCs can be used to realize a privacy-respecting form of attribute-based access control. Traditional attribute-based access control [7, 40, 48], however, does not see attributes as grouped together in a credential or token. Thus our framework allows one to realize more specific and more precise access control policies. Also, role-based access control [29, 43] can be seen as a special case of our attribute-based setting by encoding the user's roles as attributes. Recent work [33] extended RBAC with privacy-preserving authentication for the particular case of role and location attributes.

Bonatti and Samarati [7] also propose a language for specifying access control rules based on "credentials". The language focuses on credential ownership and does not allow for more advanced requirements such as for example revealing of attributes, signing statements, or inspection. The same is true for the languages proposed by Ardagna et al. [2] and by Winsborough et al. [49]. However, the latter allows one to impose attribute properties on credentials and its extension by Li et al. [35] supports revealing of attributes. The Auth-SL language [46] focuses on multi-factor authentication and enables the policy author to specify restrictions on the properties of the authentication mechanisms themselves, but not on attributes of individual users.

The language by Ardagna et al. [3] can also be considered as a predecessor to our language in the sense that it focuses on anonymous credential systems and some of the advanced features. However, it considers only the presentation phase

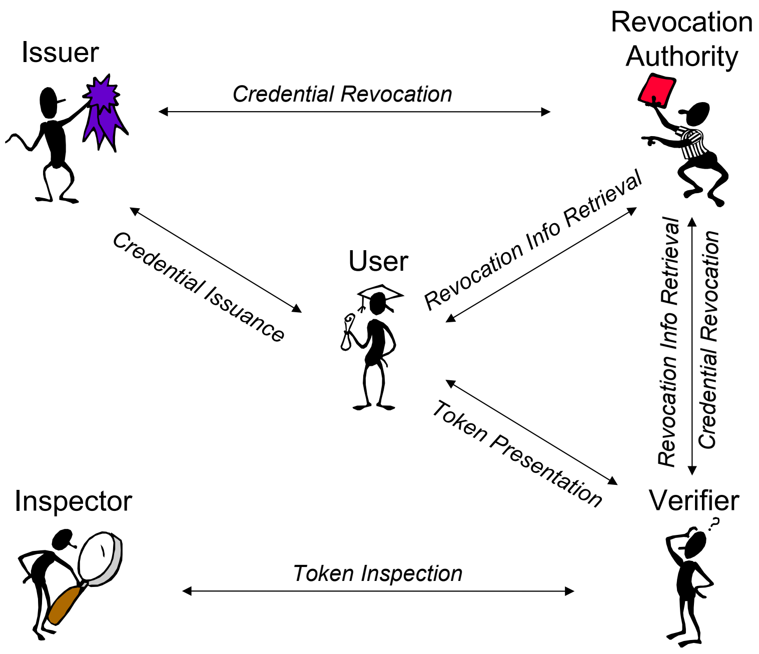


Figure 1: Entities and interactions diagram.

and is less expressive than ours, for instance, it cannot express statements involving attributes from different credentials.

VeryIDX [41] is a system to prevent identity theft by permitting the use of certain identity attributes only in combination with other identity attributes. So-called verification policies specify which attributes have to be presented together. However, these policies are introduced only conceptually without any details on exact expressivity, syntax, or semantics.

Several logic-based and technology-neutral approaches to distributed access control have been proposed [1, 9, 31, 36]. However, none of these have been designed with Privacy-ABCs in mind. In particular, they do not support selective disclosure of attributes, proving predicates over attributes, or attribute inspection.

Summarized, our language framework is the first that covers the whole life-cycle of Privacy-ABCs and also the first one unifying the full spectrum of their features.

3 Concepts and Features

Figure 1 gives an overview of the entities involved in Privacy-ABC systems and the interactions between them. These entities are *users*, *issuers*, *verifiers*, *inspectors* and *revocation authorities*. Each issuer generates a secret issuance key and publishes the *issuer parameters* that include the corresponding public verification key. Similarly, each inspector generates a private decryption key and a corresponding public encryption key, and each revocation authority generates and publishes its revocation parameters. Users get issued credentials by issuers. A credential contains attributes that its issuer vouches for w.r.t. the user. A credential can also specify one or more revocation authorities who are able to revoke the credential if necessary for some reason. Using her credentials, a user can form a presentation token that contains a subset of the certified attributes, provided that the corresponding credentials have not been revoked. Additionally, some of the attributes can be encoded in the presentation token so that they can only be retrieved by an inspector. Receiving a presentation token from a user, a verifier checks whether the presentation token is valid w.r.t. the relevant issuers' public keys and inspector public keys and the latest revocation information. If the verification succeeds, the verifier will be convinced that the attributes contained in the presentation token are vouched for by the corresponding issuers.

Informally, a secure realization of a Privacy-ABC system must guarantee that (1) users can only generate a valid presentation token if they were indeed issued the corresponding credentials that have not been revoked, (2) that attributes encoded in the presentation token for an inspector can indeed be retrieved by that inspector, and (3) that the presentation tokens do not reveal any further information about the users other than the attributes contained in them.

We now provide a brief explanation of the main features supported by Privacy-ABCs, with a focus on the ones that were not modeled so far in existing identity frameworks.

3.1 Pseudonyms

Each user can generate a secret key. Unlike traditional public-key authentication schemes, however, there is no single public key corresponding to the secret key. Rather, the user can generate as many public keys as she wishes. These public keys are called *pseudonyms* in Privacy-ABCs. Pseudonyms are cryptographically unlinkable, meaning that given two different pseudonyms, one cannot tell whether they were generated from the same or from different secret keys. By generating a different pseudonym for every verifier, users can thus be known under different unlinkable pseudonyms to different sites, yet use the same secret key to authenticate to all of them.

While it is sufficient for users to generate a single secret key, they can also have multiple secret keys. A secret key can be generated by a piece of trusted hardware (e.g., a smart card) that stores and uses the key in computations (e.g., to generate pseudonyms), but that never reveals the key. The key is thereby *bound* to the hardware, in the sense that it can only be used in combination with the hardware.

There are situations, however, where the possibility to generate an unlimited number of unlinkable pseudonyms is undesirable. For example, in an online opinion poll, users should not be able to bias the result by entering multiple votes under different pseudonyms. In such situations, the verifier can request a special pseudonym called a *scope-exclusive pseudonym*, which is unique for the user's secret key and a given *scope string*. Scope-exclusive pseudonyms for different scope strings remain unlinkable. By using the URL of the opinion poll as the scope string, for example, the verifier can ensure that each user can only register a single pseudonym to vote.

3.2 Credentials and Key Binding

A *credential* is a certified container of attributes issued by an issuer to a user. Formally, an *attribute* is described by the *attribute type* that determines the semantics of the attribute (e.g., first name) and the *attribute value* that determines its contents (e.g., "John"). By issuing a credential, the issuer vouches for the correctness of the contained attributes with respect to the user. The *credential specification* lists the attribute types that are encoded in a credential. A credential specification can be created by the issuer, or by an external authority so that multiple issuers can issue credentials according to the same specification. The credential specification must be published and distributed over a trusted channel. How exactly this is done goes beyond the scope of our language framework; the specification could for example be digitally signed by its creator.

Optionally, a credential can be *bound* to a user's secret key, i.e., it cannot be used without knowing the secret key. We call this option *key binding*. It is somewhat analogous to traditional public-key certificates, where the certificate contains the CA's signature on the user's public key, but unlike traditional public-key certificates, a Privacy-ABC is not bound to a unique public key: it is only bound to a unique secret key. A user can derive as many pseudonyms as she wishes from this secret key and (optionally) show that they were derived from the same secret key that underlies the credential.

3.3 Presentation

To authenticate to a verifier, the user first obtains the *presentation policy* that describes which credentials the user must present and which information from these credentials she must reveal. If the user possesses the necessary credentials, she can derive from these credentials a *presentation token* that satisfies the presentation policy. The presentation token can be verified using the issuer parameters of all credentials underlying the presentation token.

Presentation tokens derived from Privacy-ABCs only reveal the attributes that were explicitly requested by the presentation policy – all the other attributes contained in the credentials remain hidden. Moreover, presentation tokens are cryptographically unlinkable (meaning no collusion of issuers and verifiers can tell whether two presentation tokens were generated by the same user or by different users) and untraceable (meaning that no such collusion can correlate a presentation token to the issuance of the underlying credentials). Of course, presentation tokens are only as unlinkable as the information they intentionally reveal.

Rather than requesting and revealing full attribute values, presentation policies and tokens can also request and reveal *predicates* over one or more issued attributes. For example, a token could reveal that the name on the user's credit card matches that on her driver's license, without revealing the name. As another example, a token could reveal that the user's birthdate is before January 1st, 1994, without revealing her exact birthdate.

3.4 Issuance

In the simplest setting, an issuer knows all attribute values to be issued and simply embeds them into a credential. Privacy-ABCs also support advanced issuance features where attributes are blindly "carried over" from existing credentials, without the issuer becoming privy to their values. Similarly, the issuer can blindly issue self-claimed attribute values (i.e., not certified by an existing credential), carry over the secret key to which a credential is bound, or assign a uniformly random value to an attribute such that the issuer cannot see it and the user cannot bias it.

Advanced issuance is an interactive protocol between the user and the issuer. In the first move, the issuer provides the user with an *issuance policy* that consists of a presentation policy specifying which pseudonyms and/or existing credentials the user must present, and of a *credential template* specifying which attributes or secret keys of the newly issued credential will be generated at random or carried over from credentials or pseudonyms in the presentation policy. In response, the user sends an *issuance token* containing a presentation token that satisfies the issuance policy. Then the (possibly multi-round) cryptographic issuance protocol ensues, at the end of which the user obtains the new credential.

3.5 Inspection

Absolute user anonymity in online services easily leads to abuses such as spam, harassment, or fraud. Privacy-ABCs provide the option to add accountability for misbehaving users through a feature called *inspection*. Here, a presentation token contains one or more credential attributes that are encrypted under the public key of a trusted *inspector*. The verifier can check that the correct attribute values were encrypted, but cannot see their actual values. The *inspection grounds* describe the circumstances under which the verifier may call upon the inspector to recover the actual attribute values. The inspector is trusted to collaborate only when the inspection grounds have been met; verifiers cannot change the inspection grounds after receiving a presentation token, as the grounds are cryptographically tied to the token.

The presentation policy specifies which attributes from which credentials have to be encrypted, together with the inspector public keys and inspection grounds under which they have to be encrypted.

3.6 Revocation

Credentials may need to be revoked for several reasons: the credential and the related user secrets may have been compromised, the user may have lost her right to carry a credential, or some of her attribute values may have changed. In such cases, credentials need to be revoked globally and we call this *issuer-driven revocation*. Sometimes credentials may be revoked only for specific contexts. For example, a hooligan may see his digital identity card revoked for accessing sport stadiums, but may still use it for all other purposes. We call this *verifier-driven revocation*.

Revocation for Privacy-ABCs is cryptographically more complicated than for classical certificates, but many mechanisms with varying efficiency [34] exist [19, 39, 11, 16, 38]. Bar a few exceptions, all of them can be used for both issuer-driven and verifier-driven revocation.

We describe revocation in a generic mechanism-agnostic way and consider credentials to be revoked by dedicated *revocation authorities*. They are separate entities in general, but may be under the control of the issuer or verifier in particular settings. The revocation authority publishes static *revocation authority parameters* and periodically publishes the most recent *revocation information*. When creating presentation tokens, users prove that their credentials have not been revoked, possibly using *non-revocation evidence* that they fetch and update from the revocation authority. The revocation authority to be used is specified in the issuer parameters for issuer-driven revocation and in the presentation policy for verifier-driven revocation. When a credential is subject to issuer-driven revocation, a presentation token related to this credential must always contain a proof that the presented credential has not been revoked. Issuer-driven revocation is performed based on the *revocation handle*, which is a dedicated unique attribute embedded in a credential. Verifier-driven revocation can be performed based on any combination of attribute values, possibly even from different credentials. This allows the revocation authority for example to exclude certain combinations of first names and last names to be used in a presentation token.

3.7 Cryptographic Realization

The most prominent instantiations of Privacy-ABC systems are IBM's Identity Mixer [24, 32] and Microsoft's U-Prove [26]. Both systems currently support only a subset of the features presented here, but will be extended to support the full feature set as part of the ABC4Trust project. Here, we sketch how the different features can be realized cryptographically and give pointers to relevant related literature; a full security analysis of the combined system is beyond the scope of this paper.

At the core of a Privacy-ABC system is a signature scheme with efficient protocols to prove possession of signatures. Identity Mixer and U-Prove build on the Camenisch-Lysyanskaya [20] and the Brands [10] signature schemes, respectively, but other schemes also exist [21, 4]. An issued credential is a signature (or, for single-show schemes such as [10], a batch of renewable signatures) under such a scheme on the user's attributes. Some schemes inherently support key binding [20, 21, 4], others can be extended by using a randomly chosen attribute as secret key [10].

Ordinary pseudonyms are cryptographic commitments [42, 27] to the secret key; scope-exclusive pseudonyms can be realized as the output of a verifiable random function [28, 14] applied to the secret key as seed and the scope string as input. Inspection can be obtained through verifiable encryption [23] of the inspectable attributes. Revocation of Privacy-ABCs can be done through signed revocation lists [38], through dynamic accumulators [19, 39, 16], or through efficient updates of short-lived credentials [17].

The *glue* binding all primitives together in a presentation token is provided by generalized zero-knowledge proofs of knowledge of discrete logarithms [44, 15] made non-interactive through the Fiat-Shamir transform [30]. These proofs are also used for equality predicates over attributes; inequality predicates can be proved using range proofs [8, 13].

4 Language Framework

Given the multitude of distributed entities involved in a full-fledged Privacy-ABC system, the communication formats that are used between these entities must be specified and standardized.

None of the existing format standards for identity management protocols such as SAML, WS-Trust, or OpenID support all Privacy-ABCs' features. Although most of them can be extended to support a subset of these features, we define for the sake of simplicity and completeness a dedicated language framework which addresses all unique Privacy-ABC features. Our languages can be integrated into existing identity management systems.

In this section we introduce our framework covering the full life-cycle of Privacy-ABCs, including setup, issuance, presentation, revocation, and inspection. As the main purpose of our data artifacts is to be processed and generated by automated policy and credential handling mechanisms, we define all artifacts in XML schema notation, although one could also create a profile using a different encoding such as ASN.1 or JSON.

The XML artifacts formally describe and orchestrate the underlying cryptographic mechanisms and provide opaque containers for carrying the cryptographic data. Whenever appropriate, our formats also support user-friendly textual names or descriptions which allow to show a descriptive version of the XML artifacts to a user and to involve her in the issuance or presentation process if necessary.

For didactic purposes we describe the different artifacts realizing the concepts from Section 3 by means of examples. The full schema is available in [12]. In what follows, we explicitly distinguish between user attributes (as contained in a credential) and XML attributes (as defined by XML schema) whenever they could be confused.

4.1 Credential Specification

Recall that the credential specification describes the common structure and possible features of credentials. For example, suppose the Republic of Utopia issues electronic identity cards to its citizens containing their full name, state, and date of birth. Utopia may issue Privacy-ABCs according to the credential specification shown in Figure 2.

```
 1 <CredentialSpecification KeyBinding="true" Revocable="true">
 2   <SpecificationUID>urn:creds:id</SpecificationUID>
 3   <AttributeDescriptions MaxLength="32">
 4     <AttributeDescription Type="urn:creds:id:name" DataType="xs:string"
 5       Encoding="xenc:sha256">
 6         <FriendlyAttributeName lang="EN"> Full Name
 7         </FriendlyAttributeName>
 8     </AttributeDescription>
 9     <AttributeDescription Type="urn:creds:id:state" DataType="xs:string"
10       Encoding="xenc:sha256"/>
11     <AttributeDescription Type="urn:creds:id:bdate" DataType="xs:date"
12       Encoding="ASN1:GeneralizedTime"/>
13   </AttributeDescriptions>
14 </CredentialSpecification>
```

Figure 2: Credential specification of the identity card.

The XML attribute **KeyBinding** indicates whether credentials adhering to this specification must be bound to a secret key. The XML attribute **Revocable** being set to *"true"* indicates that the credentials will be subject to issuer-driven revocation and hence have a built-in revocation handle. The assigned revocation authority is specified in the issuer parameters.

To encode user attribute values in a Privacy-ABC, they must be mapped to integers of a limited length. The maximal length is indicated by the **MaxLength** XML attribute (Line 3), here 32 bytes. Electronic identity cards contain a person's full name, state, and date of birth. The XML attributes **Type**, **DataType**, and **Encoding** respectively contain the unique identifier for the user attribute type, for the data type, and for the encoding algorithm that specifies how the value is to be mapped to an integer of the correct size (Lines 4, 9, 11). Attributes that may have values longer than **MaxLength** have to be hashed, as is done here for the name using SHA-256. The specification can also define human-readable names for the user attributes in different languages (Line 6–7).

4.2 Issuer and Revocation Parameters

The government of Utopia, that acts as issuer and revocation authority for the identity cards, generates an issuance key pair and publishes the issuer parameters, and generates and publishes the revocation authority parameters.

```
<IssuerParameters>
  <ParametersUID>urn:utopia:id:issuer</ParametersUID>
  <AlgorithmID>urn:com:microsoft:uprove</AlgorithmID>
  <SystemParameters>...</SystemParameters>
  <CredentialSpecUID>urn:creds:id</CredentialSpecUID>
  <HashAlgorithm>xenc:sha256</HashAlgorithm>
  <CryptoParams>...</CryptoParams>
  <KeyBindingInfo>...</KeyBindingInfo>
  <RevocationParametersUID>urn:utopia:id:ra</RevocationParametersUID>
</IssuerParameters>
```

```
<RevocationAuthorityParameters>
  <ParametersUID>urn:utopia:id:ra</ParametersUID>
  <RevocationMechanism>
    urn:abc4trust:accumulators:cl
  </RevocationMechanism>
  <RevocationInfoReference ReferenceType="url">
    https:utopia.gov/id/revauth/revinfo
  </RevocationInfoReference>
  <NonRevocationEvidenceReference ReferenceType="url">
    https:utopia.gov/id/revauth/nrevevidence
  </NonRevocationEvidenceReference>
  <CryptoParams>...</CryptoParams>
</RevocationAuthorityParameters>
```

Figure 3: Issuer and revocation authority parameters.

They are illustrated in Figure 3. The **ParametersUID** element assigns unique identifiers for the issuer and revocation authority parameters. The issuer parameters additionally specify the chosen cryptographic Privacy-ABC and hash algorithm, the credential specification that credentials issued under these issuer parameters will follow, and the parameters identifier of the revocation authority that will manage the issuer-driven revocation. The **SystemParameters**, **CryptoParams**, and **KeyBindingInfo** contain cryptographic algorithm-specific information about the public key.

The revocation authority parameters can be used for both issuer- and verifier-driven revocation. They specify a unique identifier for the parameters, the cryptographic revocation mechanisms, and references to the network endpoints where the most recent revocation information and non-revocation evidence can be fetched.

4.3 Presentation Policy with Basic Features

Suppose that a user already possesses an identity card from the Republic of Utopia issued according to the credential specification depicted in Figure 2. Further suppose that all residents of Utopia can sign up for one free library card using an online issuance service. To get a library card the applicant must present her valid identity card and reveal only the state from it. This results in the presentation policy depicted in Figure 4.

```
 1 <PresentationPolicy PolicyUID="libcard">
 2   <Message>
 3     <Nonce> bkQydHBQWDR4TUZzbXJKYUM= </Nonce>
 4   </Message>
 5   <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true"/>
 6   <Credential Alias="id" SameKeyBindingAs="nym">
 7     <CredentialSpecAlternatives>
 8       <CredentialSpecUID>urn:creds:id</CredentialSpecUID>
 9     </CredentialSpecAlternatives>
10     <IssuerAlternatives>
11       <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
12     </IssuerAlternatives>
13     <DisclosedAttribute AttributeType= "urn:creds:id:state"/>
14   </Credential>
15 </PresentationPolicy>
```

Figure 4: Presentation policy for an identity card.

We now will go through the preceding presentation policy and describe how the different features of Privacy-ABCs can be realized with our language. We will first focus on the basic features and describe extended concepts such as inspection and revocation in our second example.

Signing Messages

A presentation token can optionally sign a message. The message to be signed is specified in the policy (Fig. 4, Lines 2–4). It can include a nonce (to prevent replay attacks and to use for cryptographic evidence generation), any application-specific message, and a human-readable name and/or description of the policy.

Pseudonyms

The optional **Pseudonym** element (Fig. 4, Line 5) indicates that the presentation token must contain a pseudonym. A pseudonym can be presented by itself or in relation with a credential if key binding is used (which we discuss later).

The associated XML attribute **Exclusive** indicates that a scope-exclusive pseudonym must be created, with the scope string given by the XML attribute **Scope**. This ensures that each user can create only a single pseudonym satisfying this policy, so that the registration service can prevent the same user from obtaining multiple library cards. Setting **Exclusive** to *"false"* would allow an ordinary pseudonym to be presented. The **Pseudonym** element has an optional boolean XML attribute **Established**, not illustrated in the example, which, when set to *"true"*, requires the user to re-authenticate under a previously established pseudonym. The presentation policy can request multiple pseudonyms, e.g., in order to verify that different pseudonyms actually belong to the same user.

Selective Disclosure

For each credential that the user is requested to present, the policy contains a **Credential** element (Fig. 4, Lines 6–14). The XML attribute **Alias** assigns an alias by means of which the credential can be referred to from other places in the policy, e.g., from the attribute predicates. For each credential, a list of accepted credential specifications and issuer parameters can be specified, as well as the list of attributes that must be disclosed. The preceding policy only requests the presentation of an identity card and the exposure of the state by the **DisclosedAttribute** element (Fig. 4, Line 13).

Key Binding

If present, the **SameKeyBindingAs** attribute of a **Credential** or **Pseudonym** element (Fig. 4, Line 6), contains an alias referring either to another Pseudonym element within this policy, or to a Credential element for a credential with key binding. This indicates that the current pseudonym or credential and the referred pseudonym or credential have to be bound to the same key. In our preceding example, the policy requests that the identity card and the presented pseudonym must belong to the same secret key.

4.4 Issuance Policy

To support the advanced features described in Section 3, we propose a dedicated *issuance policy.*

A library card contains the applicant's name and is bound to the same secret key as the identity card. So the identity card must not only be presented, but also used as a source to carry over the name and the secret key to the library card, and the library should learn neither of them during the issuance process. Altogether, to issue library cards the state library creates an issuance policy depicted in Figure 5. It contains the presentation policy from Figure 4 and the credential template that we describe in details below.

```
 1 <IssuancePolicy>
 2   <PresentationPolicy PolicyUID="libcard">... </PresentationPolicy>
 3   <CredentialTemplate SameKeyBindingAs="id">
 4     <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
 5     <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
 6     <UnknownAttributes>
 7       <CarriedOverAttribute TargetAttributeType= "urn:utopia:lib:name">
 8         <SourceCredentialInfo Alias="id" AttributeType="urn:creds:id:name"/>
 9       </CarriedOverAttribute>
10     </UnknownAttributes>
11   </CredentialTemplate>
12 </IssuancePolicy>
```

Figure 5: Issuance policy for a library card. The presentation policy on Line 2 is depicted in Figure 4.

Credential Template

A credential template describes the relation of the new credential to the existing credentials that were requested in the presentation policy. The credential template (Fig. 5, Lines 3–11) must first state the unique identifier of the credential specification and issuer parameters of the newly issued credential. The optional XML attribute **SameKeyBindingAs** further specifies that the new credential will be bound to the same secret key as a credential or pseudonym in the presentation policy, in this case the identity card.

Within the **UnknownAttributes** element (Fig. 5, Lines 6–10) it is specified which user attributes of the new credential will be carried over from existing credentials in the presentation token. The **SourceCredentialInfo** element (Fig. 5, Line 8) indicates the credential and the user attribute of which the value will be carried over.

Although this is not illustrated in our example, an attribute value can also be specified to be chosen jointly at random by the issuer and the user. This is achieved by setting the optional XML attribute **JointlyRandom** to *"true"*.

4.5 Presentation and Issuance Token

A *presentation token* consists of the *presentation token description*, containing the mechanism-agnostic description of the revealed information, and the *cryptographic evidence*, containing opaque values from the specific cryptography that "implements" the token description. The presentation token description roughly uses the same syntax as a presentation policy. An *issuance token* is a special presentation token that satisfies the stated presentation policy, but that contains additional cryptographic information required by the credential template.

The main difference with the presentation and issuance policy is that in the returned token a **Pseudonym** (if requested in the policy) now also contains a concrete **PseudonymValue** (Fig. 6, Line 6). Similarly, the **DisclosedAttribute** elements (Fig. 6, Lines 10–12) in a token now also contain the actual user attribute values. Finally, all data from the cryptographic implementation of the presentation token and the advanced issuance features are grouped together in the **CryptoEvidence** element (Fig. 6, Line 17).

```
 1 <IssuanceToken>
 2   <IssuanceTokenDescription>
 3     <PresentationTokenDescription PolicyUID ="libcard" >
 4       <Message>...</Message>
 5       <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true" />
 6         <PseudonymValue> MER2VXISHI=</PseudonymValue>
 7       </Pseudonym>
 8       <Credential Alias="id" SameKeyBindingAs="nym" >
 9           ...
10         <DisclosedAttribute AttributeType="urn:creds:id:state" >
11           <AttributeValue> Nirvana </AttributeValue>
12         </DisclosedAttribute>
13       </Credential>
14     </PresentationTokenDescription>
15     <CredentialTemplate SameKeyBindingAs="id" >...</CredentialTemplate>
16   </IssuanceTokenDescription>
17   <CryptoEvidence>...</CryptoEvidence>
18 </IssuanceToken>
```

Figure 6: Issuance token for obtaining the library card.

4.6 Presentation Policy with Extended Features

Suppose that the state library has a privacy-friendly online interface for borrowing digital and paper books. Books can be browsed and borrowed anonymously using the digital library cards based on Privacy-ABCs. Paper books can be delivered in anonymous numbered mailboxes at the post office. However, when books are returned late or damaged, the library must be able to identify the reader to impose an appropriate fine. Repeated negligence may even lead to exclusion from borrowing further paper books while borrowing digital books remains possible.

Moreover, assume that the library offers special conditions for young readers that can be used by anyone below the age of twenty-six years. As library cards do not contain a date of birth, a user must prove to be below that age by combining her library card with her identity card. Altogether, for borrowing books under the "young-reader"-conditions, users have to satisfy the presentation policy depicted in Figure 7.

A presentation policy that is used for plain presentation (i.e., not within an issuance policy) can consist of multiple policy alternatives, each wrapped in a separate **PresentationPolicy** element (Fig. 7, Lines 2–37). The returned presentation token must satisfy (at least) one of the specified policies.

```
 1 <PresentationPolicyAlternatives>
 2   <PresentationPolicy PolicyUID= "young−reader" >
 3     <Message>...</Message>
 4     <Credential Alias="libcard" SameKeyBindingAs="id" >
 5       <CredentialSpecAlternatives>
 6         <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
 7       </CredentialSpecAlternatives>
 8       <IssuerAlternatives>
 9         <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
10       </IssuerAlternatives>
11       <DisclosedAttribute AttributeType= "urn:utopia:lib:name" >
12         <InspectorAlternatives>
13           <InspectorPublicKeyUID> urn:lib:arbitrator
14           </InspectorPublicKeyUID>
15         </InspectorAlternatives>
16         <InspectionGrounds> Late return or damage. </InspectionGrounds>
17       </DisclosedAttribute>
18     </Credential>
19     <Credential Alias="id" >
20       <CredentialSpecAlternatives>
21         <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
22       </CredentialSpecAlternatives>
23       <IssuerAlternatives>
24         <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
25       </IssuerAlternatives>
26     </Credential>
27     <VerifierDrivenRevocation>
28       <RevocationParametersUID> urn:lib:blacklist
29       </RevocationParametersUID>
30       <Attribute CredentialAlias ="libcard"
31         AttributeType="urn:utopia:lib:name" />
32     </VerifierDrivenRevocation>
33     <AttributePredicate Function= "...:date−greater−than" >
34       <Attribute CredentialAlias ="id" AttributeType= "urn:creds:id:bdate" />
35       <ConstantValue> 1986−07−16 </ConstantValue>
36     </AttributePredicate>
37   </PresentationPolicy>
38 </PresentationPolicyAlternatives>
```

Figure 7: Presentation policy for borrowing books.

The example presentation policy contains two **Credential** elements, for the library and for the identity card, which must belong to the same secret key as indicated by the XML attribute **SameKeyBindingAs**.

Attribute Predicates

No user attributes of the identity card have to be revealed, but the **AttributePredicate** element (Fig. 7, Lines 33–36) specifies that the date of birth must be after July 16th, 1986, i.e., that the reader is younger than twenty-six. Supported predicate functions include equality, inequality, greater-than and less-than tests for most basic data types, as well as membership of a list of values. The arguments of the predicate function may be credential attributes (referred to by the credential alias and the attribute type) or constant values.

Inspection

To be able to nevertheless reveal the name of an anonymous borrower and to impose a fine when a book is returned late or damaged, the library can make use of inspection. The **DisclosedAttribute** element for the user attribute *"...:name"* contains **InspectorPublicKeyUID** and **InspectionGrounds** child elements, indicating that the attribute value must not be disclosed to the verifier, but to the specified inspector with the specified inspection grounds. The former child element specifies the inspector's public key under which the value must be encrypted, in this case belonging to a designated arbiter within the library. The latter element specifies the circumstances under which the attribute value may be revealed by the arbiter. Our language also provides a data artifact for inspection public keys, which we omit here for space reasons.

Issuer-Driven Revocation

When the presentation policy requests a credential that is subject to issuer-driven revocation, the credential must be proved to be valid with respect to the most recent revocation information. However, a policy can also specify a particular version of the revocation information to use. In the latter case, the element **IssuerParametersUID** has an extra XML attribute **RevocationInformationUID** containing the identifier of the specific revocation information. The specification of the referenced **RevocationInformation** is given in [12].

Verifier-Driven Revocation

If customers return borrowed books late or damaged, they are excluded from borrowing further paper books, but they are still allowed to use the library's online services. In our example, this is handled by a **VerifierDrivenRevocation** element (Fig. 7, Lines 26–29), which specifies that the user attribute *"...:name"* of the library card must be checked against the most recent revocation information from the revocation authority *"urn:lib:blacklist"*. Revocation can also be based on a combination of user attributes from different credentials, in which case there will be multiple **Attribute** child elements per **VerifierDrivenRevocation**. The presentation policy can also contain multiple **VerifierDrivenRevocation** elements for one or several credentials, the returned presentation token must then prove its non-revoked status for *all* of them.

5 Semantics

To precisely specify the meaning of the XML language that we propose, we provide a formal semantics which mathematically defines the various language features and thus the intended system behavior. The semantics that we propose is based on the one shown by Camenisch et al. [22] for the CARL language. Similar to their approach, we also define our semantics in the context of a state transition system, but apart from token presentation we additionally cover state transitions for credential issuance, credential revocation, and token inspection as well as pseudonyms and key binding.

In our model, users maintain credential portfolios containing the credentials that were issued to them. Issuers and verifiers maintain knowledge states about their communication partners in the form of logical predicates. These predicates express that a certain logical expression over the communication partner's credentials has been successfully verified at the time of the transition. Similar to [22], the semantics is given in terms of the effects that the transitions have on the knowledge states and portfolios of the parties involved in the transition.

5.1 Dedicated Functions and Attributes

We abstract away from the XML notation of our language and list all the information contained in a presentation token in a formal manner. We first introduce some notation, then we specify the semantics of attribute predicates, and afterwards we describe a set of dedicated functions and attributes that are needed to specify the semantics associated with pseudonyms, inspection, revocation, and attribute predicates.

We assume that an ontology $\mathfrak{T}$ defines *data types*, *attributes*, *credential types*, and functions on the data types, which correspond to the data types, attribute types, credential specifications and predicate functions from the XML notation. All parties in the system are assumed to use the same ontology $\mathfrak{T}$. A credential type defines which attributes the credentials of this type contain. An issued credential is represented as a function of its attributes to values of the correct data types. The ontology contains the equality function for all data types and greater-than and less-than functions on integers and dates.

An **AttributePredicate** element represents a logical predicate $\phi = f(\ldots)$, where f is a boolean-valued function over *attribute variables* and constants, which are expressed by **Attribute** and **ConstantValue** XML elements, respectively. We denote attribute variables with $C.a$ for a credential alias C and an attribute a. If an interpretation $\mathcal{I}$ assigns the value *true* to a predicate ϕ, it is called a model of that predicate, which we denote as $\mathcal{I} \models \phi$. Multiple predicates $\phi_1, \ldots, \phi_n$ within the same **PresentationPolicy** element are conjunctively combined to $\phi = \phi_1 \wedge \ldots \wedge \phi_n$. We define the meaning of attribute predicates with respect to a given ontology $\mathfrak{T}$ and an interpretation $\mathcal{I}$. An interpretation assigns values to the symbols in a formula so that the formula can be evaluated. In particular, for a function symbol f, we denote by $f^{\mathfrak{I}}$ the ontology-defined meaning of f, and for terms $t_1, \ldots, t_n$, where a term is an attribute variable or a constant, we define $f(t_1, \ldots, t_n)^{\mathcal{I}} = f^{\mathfrak{I}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$ and $(C.a)^{\mathcal{I}} = (C^{\mathcal{I}})(a)$.

We assume a non-deterministic function $k \leftarrow \mathit{genKey}$, which generates a new unique secret key k every time it is called. We denote the set of keys that user U generated with $\mathfrak{S}_U$ and assume all users' key sets are disjoint. Further, we assume a non-deterministic function $p \leftarrow \mathit{nym}(k)$, which derives a new unique *pseudonym* p from secret key k every time it is called, and a boolean function $\mathit{verifyNym}(p, k)$ that is *true* iff p was derived from k. Pseudonyms derived from the same secret key are in principle unlinkable, i.e., for two pseudonyms $p_1 = \mathit{nym}(k)$ and $p_2 = \mathit{nym}(k')$, it cannot be determined whether $k = k'$, unless the user explicitly creates a presentation token showing that p and p' are associated to the same key, denoted by the boolean function $\mathit{verifyNyms}(p, p')$.

Also, we assume a function $d \leftarrow \mathit{seNym}(k, w)$, which derives a unique scope-exclusive pseudonym d from a secret key k for scope w, as well as a boolean function $\mathit{verifySeNym}(d, w, k)$ that is *true* iff d was derived from k for w. Scope-exclusive pseudonyms are also unlinkable, i.e. for two pseudonyms $d_1 \leftarrow \mathit{seNym}(k, w)$ and $d_2 \leftarrow \mathit{seNym}(k', w')$ with $w \neq w'$, it cannot be determined whether $k = k'$. We denote with $\mathfrak{N}_U$ and $\mathfrak{D}_U$ the pseudonyms and scope-exclusive pseudonyms that were derived from a key $k \in \mathfrak{S}_U$.

For inspection, we further assume a verifiable encryption function $e \leftarrow \mathit{vEncrypt}(S, g, v_1, v_2, \ldots)$, which encrypts a concatenation of values $v_1, v_2, \ldots$ under the public key of party S and inspection grounds g, as well as a decryption function $\langle v_1, v_2, \ldots\rangle \leftarrow \mathit{vDecrypt}(e, sk, g')$, which decrypts e with the secret key sk on grounds g' to the originally encrypted concatenation $\langle v_1, v_2, \ldots\rangle$ iff sk is the secret key of S and $g = g'$. Also, we assume a boolean function $\mathit{verifyEnc}(e, S, g, v_1, v_2, \ldots)$, which evaluates to *true* iff $e = \mathit{vEncrypt}(S, g, v_1, v_2, \ldots)$.

All credentials have the dedicated attributes *type* and *issuer*. Credentials with key binding, i.e., whose credential specification has XML attribute **KeyBinding**= "*true*", additionally have an attribute *uKey* of type *String* containing the secret key of its owner. Revocable credentials have a dedicated attribute *rHandle* representing the revocation handle.

5.2 Credential Issuance

We first consider simple issuance policies whose credential template merely specifies a credential type τ and an issuer I. After introducing the semantics for token presentation, we extend the issuance semantics for unrestricted issuance policies.

An issuance transition between a user U and an issuer I has effects on the user's credential portfolio $\mathfrak{P}_U$ as well as the issuer's knowledge state $\mathfrak{K}_I(X)$, where X is some issuer-chosen identifier under which U is currently known to I. Note that the same user U may be known to the issuer under multiple different identifiers. After the transition, the augmented credential portfolio $\mathfrak{P}'_U = \mathfrak{P}_U \cup \{N\}$ contains a new credential N of type τ, of issuer I, and with $N(a_i) = A_i :: \beta_i$ for $a_i :: \beta_i \in A_\tau$, where $a :: \beta$ denotes that attribute a has data type β, where $A_\tau = \{a_1 :: \beta_1, \ldots\}$ is the set of attributes that credentials of credential type τ have, and where the A_i are the attribute values that I certifies for U. If τ is revocable, then $N(\mathit{rHandle}) = r$, where r is a globally unique issuer-chosen revocation handle. The new knowledge state $\mathfrak{K}'_I(X)$ of the issuer increases such that $\mathfrak{K}'_I(X) = \mathfrak{K}_I(X) \wedge N.\mathit{type} == \tau \wedge N.\mathit{issuer} == I \wedge \bigwedge_{i=1}^{|A_\tau|} N.a_i == A_i$. If τ is revocable, then $\mathfrak{K}'_I(X)$ has the additional conjunct $N.\mathit{rHandle} == r$. This models that for simple issuance policies the issuer learns the values of *all* the attributes that are certified in the newly issued credential N.

5.3 Credential Revocation

In order to define the semantics of revocation, we introduce revocation epochs and revocation logs. Note that these concepts are only part of our ideal world that is modeled, i.e., typically they do not have counterparts in a real implementation. We assume that revocation authorities maintain revocation epochs in the form of sequence numbers, where a function $h \leftarrow \mathit{currEpoch}(Y)$ returns the current epoch $h \in \mathbb{Z}$ for revocation authority Y. Revocation authorities advance their current epoch at their own discretion, e.g., at regular time intervals or whenever a new credential is revoked, from h to $h' > h$.

For issuer-driven revocation, the revocation authority I maintains a local *revocation log file* $\mathfrak{R}_I$ that is a set of pairs (h, r) of an epoch $h \in \mathbb{Z}$ and a revocation handle r, representing that the credential with handle r was revoked in epoch h. An issuer adding an entry to his revocation log models an issuer-revocation transition in our transition system. A *credential C is* not issuer-revoked *for epoch h*, denoted $\mathit{notIssRevoked}(I, h, r)$, iff $\nexists \big(h', r\big) \in \mathfrak{R}_{C(issuer)} \cdot h' \leq h$. Similarly, for verifier-driven revocation, each revocation authority V maintains a local revocation log file $\mathfrak{L}_V$ containing tuples $(h, v_1, v_2, \ldots)$ of an epoch $h \in \mathbb{Z}$ and a list of values $v_1, v_2, \ldots$. The predicate $\mathit{notVerRevoked}(V, h, v_1, v_2, \ldots)$ is *true* iff $\nexists (h', v_1, v_2 \ldots) \in \mathfrak{L}_V \cdot h' \leq h$.

5.4 Token Presentation

In this section we define the semantics of a token presentation transition, which models that a user U proves to a verifier S that she fulfills the statements made in a given presentation token. We first formalize the requirements on the user's portfolio such that it can be used to prove a token, and then we specify the knowledge increase of the verifier to whom the token is proved.

Let T be a presentation token that, from an abstract point of view, i.e. without considering its form in XML notation, contains the following information:[3]

[3]We $\underline{\text{underline}}$ variables to express that they are associated with a presentation token.

- $\underline{k}$ pseudonym values $\underline{p}_1, \dots, \underline{p}_{\underline{k}}$
- $\underline{\ell}$ scope-excl. pseudonym values $\underline{d}_1, \dots, \underline{d}_{\underline{\ell}}$ for scopes $\underline{w}_{d_1}, \dots, \underline{w}_{d_\ell}$
- $\underline{n}$ credentials $\underline{C}_1 :: \underline{\tau}_1, \dots, \underline{C}_{\underline{n}} :: \underline{\tau}_{\underline{n}}$ of issuers $\underline{I}_{C_1}, \dots, \underline{I}_{C_n}$ with reference revocation epochs $\underline{h}_{C_1}, \dots, \underline{h}_{C_n}$
- Attribute predicate $\underline{\phi}$
- Disclosed values $\underline{v}_{(i,S)}$ for the attributes $\underline{C.a}_{(i,S)}$, for $1 \le i \le \underline{n}_S$ where $\underline{n}_S$ is the number of attributes disclosed to S
- Attributes $\underline{C.a}_{(i,S_j)}$ that are verifiably encrypted for inspector $\underline{S}_j$ in ciphertext $\underline{e}_{S_j}$ on grounds $\underline{g}_{S_j}$ for $1 \le i \le \underline{n}_{S_j}$ and $1 \le j \le \underline{q}$, where $\underline{n}_{S_j}$ is the number of attributes disclosed to $\underline{S}_j$ and $\underline{q}$ is the number of different inspectors
- Non-revoked attributes $\underline{C.a}_{(i,V_j)}$ at revocation authority $\underline{V}_j$ with reference revocation epochs $\underline{h}_{V_j}$ for $1 \le i \le \underline{n}_{V_j}$ and $1 \le j \le \underline{o}$ where $\underline{n}_{V_j}$ is the number of revocation attributes of $\underline{V}_j$ and $\underline{o}$ is the number of revocation authorities
- Message $\underline{m}$
- Symmetric key binding relation $\underline{K} \subseteq \underline{R} \times \underline{R}$, with $\underline{R} = \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\} \cup \{\underline{p}_1, \dots, \underline{p}_{\underline{k}}\} \cup \{\underline{d}_1, \dots, \underline{d}_{\underline{\ell}}\}$, where $x \mapsto y \in \underline{K}$ states that x is key-bound to y.

As a credential's type and issuer are treated as dedicated attributes, for each credential $C :: \tau \in \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\}$ issued by I we let the predicate $\underline{\phi}$ contain an additional conjunct $C.type == \tau \wedge C.issuer == I$. Further, for all pairs of credentials $C, C' \in \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\}$ that are bound to the same key, i.e., $C \mapsto C' \in \underline{K}$, we let the predicate $\underline{\phi}$ contains a conjunct $C.uKey == C'.uKey$. Note that there is only one inspection ground per inspector in our model, while in the XML notation there is one ground per attribute to disclose. This is because in our model the attributes disclosed to one inspector are *packaged* into one ciphertext to capture their semantic correlation. Without preserving this correlation, dishonest verifiers might mix attributes disclosed within different ciphertexts or across different token presentations.

We say that U *can prove the presentation token* T iff there exists a credential assignment b and an interpretation $\mathcal{I}$ such that

1. b is a total mapping from $\{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\}$ to $\mathfrak{P}_U$,
2. $\mathcal{I} \models \underline{\phi}$, with $\underline{C}_i^{\mathcal{I}} = b(\underline{C}_i)$ for $1 \le i \le n$,
3. $\forall C \in \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\} \cdot \mathit{notIssRevoked}\big(\underline{I}_C, \underline{h}_C, (C.rHandle)^{\mathcal{I}}\big)$,
4. $\forall V \in \{\underline{V}_1, \dots, \underline{V}_{\underline{o}}\} \cdot \mathit{notVerRevoked}(V, \underline{h}_V, (\underline{C.a}_{(1,V)})^{\mathcal{I}}, \dots, (\underline{C.a}_{(\underline{n}_V, V)})^{\mathcal{I}})$,
5. $\forall i \in \{1, \dots, \underline{n}_S\} \cdot (\underline{C.a}_{(i,S)})^{\mathcal{I}} = \underline{v}_{(i,S)}$, i.e., the sent values are correct w.r.t. the credential assignment b,
6. $\forall x \in \{\underline{S}_1, \dots, \underline{S}_{\underline{q}}\} \cdot \mathit{verifyEnc}\big(\underline{e}_x, x, \underline{g}_x, (\underline{C.a}_{(1,x)})^{\mathcal{I}}, \dots, (\underline{C.a}_{(\underline{n}_x, x)})^{\mathcal{I}}\big)$,
7. $\forall i \in \{1, \dots, \underline{k}\} \cdot \underline{p}_i \in \mathfrak{N}_U \wedge \big(e(\underline{p}_i) \rightarrow \mathfrak{K}_S(\underline{p}_i) \neq \epsilon\big)^4 \wedge \big(\forall C \in \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\} \cdot \underline{p}_i \mapsto C \in$

[4] If pseudonym $\underline{p}_i$ is claimed to be established, then $\underline{p}_i$ must have been used in a previous token presentation transition with verifier S

$\underline{K} \to \mathit{verifyNym}(\underline{p}_i, (C.uKey)^{\mathcal{I}})\big) \wedge \big(\forall j \in \{1, \ldots, i-1, i+1, \ldots, \underline{k}\} \cdot \underline{p}_i \mapsto \underline{p}_j \in \underline{K} \to \mathit{verifyNyms}(\underline{p}_i, \underline{p}_j)\big) \wedge \big(\forall j \in \{1, \ldots, \underline{\ell}\} \cdot \underline{p}_i \mapsto \underline{d}_j \in \underline{K} \to \mathit{verifyNyms}(\underline{p}_i, \underline{d}_j)\big)$, where the predicate $e(x)$ denotes that pseudonym x is claimed to be established and ϵ denotes an empty predicate, i.e., U knows the secret keys of the pseudonyms, the pseudonyms are correctly established and all stated key-bindings are correct, and

8. $\forall i \in \{1, \ldots, \underline{\ell}\} \cdot \underline{d}_i \in \mathfrak{D}_U \wedge \big(e(\underline{d}_i) \to \mathfrak{K}_S(\underline{d}_i) \neq \epsilon\big) \wedge \big(\forall C \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \cdot \underline{d}_i \mapsto C \in \underline{K} \to \mathit{verifySeNym}(\underline{d}_i, \underline{w}_{d_i}, (C.uKey)^{\mathcal{I}})\big) \wedge \big(\forall j \in \{1, \ldots, i-1, i+1, \ldots, \underline{\ell}\} \cdot \underline{d}_i \mapsto \underline{d}_j \in \underline{K} \to \mathit{verifyNyms}(\underline{d}_i, \underline{d}_j)\big) \wedge \big(\forall j \in \{1, \ldots, \underline{k}\} \cdot \underline{d}_i \mapsto \underline{p}_j \in \underline{K} \to \mathit{verifyNyms}(\underline{d}_i, \underline{p}_j)\big)$.

When a user *can* prove a token and convinces the verifier S of this fact by engaging in a presentation proof protocol, the effect of the token presentation transition is twofold. First, there is an increase of knowledge of S about the token's pseudonyms in the form of a logical predicate. This models that S learns that the corresponding predicate was *true* at the time of the transition. Second, the transcript of the proof protocol acts as transferable *signature* σ_T on message $\underline{m}$, i.e., S may subsequently use σ_T to convince third parties about the fact that the user who proved token T (i.e., who fulfilled all the properties stated in T) consented to the statement $\underline{m}$.

Concerning the knowledge increase, we require that the credential variables in the verifier's knowledge state are disjoint with the credential variables $\underline{C}_1, \ldots, \underline{C}_n$ of T. If this is not the case, the variables of the verifier's state have to be renamed. Let $\underline{K}^C_x = \{y \in \{\underline{C}_1, \ldots, \underline{C}_{\underline{n}}\} \mid x \mapsto y \in \underline{K}\}$ be the set containing all credentials that x is key-bound to, and let $\underline{K}^{\mathfrak{N}}_x$ and $\underline{K}^{\mathfrak{D}}_x$ be analog sets for pseudonyms and scope-exclusive pseudonyms, respectively. The following specifies the knowledge of S after the token presentation transition about the $1 \leq i \leq \underline{k}$ pseudonyms:

$$\begin{aligned}
\mathfrak{K}'_S(\underline{p}_i) = \mathfrak{K}_S(\underline{p}_i) \wedge \underline{\phi} \wedge \textstyle\bigwedge_{j=1}^{\underline{n}_S} \underline{C.a}_{(j,S)} == \underline{v}_{(j,S)} \wedge \\
\textstyle\bigwedge_{j=1}^{\underline{n}} \mathit{notIssRevoked}(\underline{I}_{C_j}, \underline{h}_{C_j}, \underline{C}_j.rHandle) \wedge \\
\textstyle\bigwedge_{j=1}^{\underline{o}} \mathit{notVerRevoked}(\underline{V}_j, \underline{h}_{V_j}, \underline{C.a}_{(1,V_j)}, \ldots, \underline{C.a}_{(\underline{n}_{V_j},V_j)}) \wedge \\
\textstyle\bigwedge_{j=1}^{\underline{q}} \mathit{verifyEnc}(\underline{e}_{S_j}, \underline{S}_j, \underline{g}_{S_j}, \underline{C.a}_{(1,S_j)}, \ldots, \underline{C.a}_{(\underline{n}_{S_j},S_j)}) \wedge \\
\textstyle\bigwedge_{x \in \underline{K}^C_{\underline{p}_i}} \mathit{verifyNym}(\underline{p}_i, x.uKey) \wedge \\
\textstyle\bigwedge_{x \in \underline{K}^{\mathfrak{N}}_{\underline{p}_i} \cup \underline{K}^{\mathfrak{D}}_{\underline{p}_i}} \mathit{verifyNyms}(\underline{p}_i, x) \wedge \\
\textstyle\bigwedge_{j=1}^{i-1} \overrightarrow{\mathfrak{K}'_S(\underline{p}_j)} \wedge \bigwedge_{j=i+1}^{\underline{k}} \overrightarrow{\mathfrak{K}'_S(\underline{p}_j)} \wedge \bigwedge_{j=1}^{\underline{\ell}} \overrightarrow{\mathfrak{K}'_S(\underline{d}_j)}
\end{aligned}$$

The analog we require for knowledge states $\mathfrak{K}'_S(\underline{d}_i)$ for $1 \leq i \leq \underline{\ell}$, where the last three lines of the formula are adapted accordingly. The above formula models that S learns the values that U discloses, and that the predicate $\underline{\phi}$ over the contained attribute variables holds, however, S does not learn the values of these attributes.

Further, S learns that the credentials used to prove the token are neither issuer-revoked nor verifier-revoked, however, S neither learns the revocation handle nor the values of the non-revoked attributes, respectively. Also, S learns that the ciphertext $\underline{e}_x$ for inspector x contains the values that correspond to the attributes $\underline{C.a}_{(1,x)}, \ldots, \underline{C.a}_{(1,\underline{n}_x)}$, however, he does not know what the values are. The verifier also learns which credentials and pseudonyms have the same underlying key, however, he does not learn the key itself. The last line of above formula models that the knowledge states of all pseudonyms used within the presentation token are merged. We use the notation $\overrightarrow{\cdot}$ to express that a predicate is not used by value, but rather by reference. This models that a verifier can transitively link all pseudonyms that were jointly contained in presentation tokens proved to him. For example, consider two consecutive token presentations where in the first one the predicate ψ_1 for pseudonyms p_1 and p_2 is proved, and in the second one ψ_2 with p_1. With the 'by reference' semantics, $\mathfrak{K}(p_2)$ is then $\psi_1 \wedge \psi_2$, rather than just ψ_1 with the 'by value' semantics. For the knowledge states that are merged, we require that the states' variables are disjoint, as otherwise new (and possibly wrong) knowledge about credentials is created by the merge operation itself. In case a token contains no pseudonyms at all, i.e., $\underline{k}+\underline{\ell} = 0$, a new unique identifier X is used to refer to the new knowledge $\mathfrak{K}'_S(X) = \underline{\phi} \wedge \cdots$ (same as above, except the last three formula lines as they contain references to pseudonyms). Unlinkability of credentials is achieved due to the renaming of the verifier's variables that was mentioned above, i.e., a verifier does not learn whether a user used the same or different credentials to prove a particular token.

5.5 Policy Verification

For a server to make an access control decision after a presentation token has been proved, he verifies that token w.r.t. a presentation policy. In the following, we describe how such verification is performed in detail. Let P be the presentation policy of S that U needs to satisfy. Each alternative contains from an abstract point of view, i.e. without considering its form in XML notation, the following information:

– k pseudonym aliases $p_1, \ldots, p_k$, optionally established
– ℓ scope-excl. pseudonym aliases $d_1, \ldots, d_\ell$ for scopes $w_{d_1}, \ldots, w_{d_\ell}$, optionally established
– Message m
– n credentials $C_1, \ldots, C_n$ with type- and issuer-alternatives, and revocation epochs $h_{I_{(i,C_j)}}$ for each issuer alternative $I_{(i,C_j)}$ for $1 \leq i \leq n_{C_j}$ and $1 \leq j \leq n$, where n_{C_j} is the number of issuer alternatives for C_j
– Attribute predicate ϕ
– Attributes $C.a_{(i,S)}$ to disclose for $1 \leq i \leq n_S$ where n_S is the number of attributes to disclose to S

– Attributes $C.a_{(i,S_j)}$ to disclose to inspector S_j on grounds g_{S_j} for $1 \leq i \leq n_{S_j}$ and $1 \leq j \leq q$ where n_{S_j} is the number of attributes to disclose to S_j and q is the number of different inspectors
– Non-revoked attributes $C.a_{(i,V_j)}$ at revocation authority V_j with reference revocation epochs h_{V_j} for $1 \leq i \leq n_{V_j}$ and $1 \leq j \leq \underline{o}$ where n_{V_j} is the number of revocation attributes of V_j and o is the number of revocation authorities
– Symmetric key binding relation $K \subseteq R \times R$, with $R = \{C_1, \dots, C_n\} \cup \{p_1, \dots, p_k\} \cup \{d_1, \dots, d_\ell\}$, where $x \mapsto y \in K$ states that x is key-bound to y.

For simplicity, we do not consider inspector alternatives. Again, a credential's type and issuer are treated as dedicated attributes, and thus are specified as part of the predicate ϕ. In particular, specifying alternatives $\tau_1, \dots, \tau_n$ for a credential C is an abbreviation for specifying $C.\mathit{type}{==}\ \tau_1 \ \vee \dots \vee C.\mathit{type}{==}\ \tau_n$ as additional conjunct of ϕ. Issuer alternatives are treated analog. We further assume that for all pairs of credentials $C, C' \in \{C_1, \dots, C_n\}$ that are bound to the same key, i.e., $C \mapsto C' \in K$, ϕ contains a conjunct $C.uKey == C'.uKey$. Note that the revocation epochs are optional in the XML notation. For our model we assume $h_x = \mathit{currEpoch}(x)$ for revocation authority x in case h_x is not given.

We say that *the presentation token T fulfills the presentation policy P* iff:

1. $\underline{k} \geq k$ and $\forall x \in \{1, \dots, k\} \cdot e(x) \to \big(\mathfrak{K}_S(\underline{p}_x) \neq \epsilon\big)$, where the predicate $e(x)$ denotes that x-th pseudonym in the policy is required to be established and ϵ denotes an empty predicate, i.e., at least k pseudonyms are provided and established, respectively,
2. $\underline{\ell} \geq \ell$ and $\forall x \in \{1, \dots, \ell\} \cdot \underline{w}_{d_x} {=} w_{d_x} \wedge \big(e(x) \to (\mathfrak{K}_S(\underline{d}_x) \neq \epsilon)\big)$, i.e., at least ℓ scope-exclusive pseudonyms for the requested scopes are provided and established, respectively,
3. $m = \underline{m}$, i.e., the correct message is signed,
4. $\{C_1, \dots, C_n\} \subseteq \{\underline{C}_1, \dots, \underline{C}_{\underline{n}}\}$, i.e., the same credential variables are used in the policy and the token,
5. $\underline{\phi}$ implies ϕ, i.e., the proved attribute predicate implies the one requested, which incorporates verifying the permitted type- and issuer-alternatives,
6. $\forall x \in \{C_1, \dots, C_n\} \cdot \underline{h}_x \geq h_{\underline{I}_x}$, i.e., all requested credentials were valid w.r.t. the corresponding epoch required by the policy or a later one,
7. $\{C.a_{(1,S)}, \dots, C.a_{(n_S,S)}\} \subseteq \{\underline{C.a}_{(1,S)}, \dots, \underline{C.a}_{(\underline{n}_S,S)}\}$, i.e., at least the requested attributes to disclose are disclosed,
8. $\forall x \in \{S_1, \dots, S_q\} \cdot g_x = \underline{g}_x \wedge \{C.a_{(1,x)}, \dots, C.a_{(n_x,x)}\} \subseteq \{\underline{C.a}_{(1,x)}, \dots, \underline{C.a}_{(\underline{n}_x,x)}\}$, i.e., at least the requested attributes to disclose to the respective inspectors are disclosed,
9. $\forall x \in \{V_1, \dots, V_n\} \cdot \underline{h}_x \geq h_x \wedge \forall i \in \{1, \dots, n_x\} \cdot C.a_{(i,x)} = \underline{C.a}_{(i,x)}$, i.e., the exact requested combination of attribute values is not revoked at the respective requested revocation authorities, and
10. there exists a total injective mapping r from R to $\underline{R}$ that maps (a) the policy's

credential aliases to token aliases and (b) the policy's pseudonym aliases to token pseudonym values, such that $\forall x \mapsto y \in K \cdot \exists x' \mapsto y' \in \underline{K} \cdot x' = r(x) \wedge y' = r(y)$, i.e., for all key-bindings required by the policy there exists a corresponding key-binding in the token.

A token T fulfills a presentation policy $P = \{P_1, \ldots, P_{n_P}\}$ with alternatives $P_1, \ldots, P_{n_P}$ iff $\exists p \in P$ such that T fulfills p.

5.6 Token Inspection

During a token presentation transition, a server S may obtain verifiably encrypted values for attributes $\underline{C.a}_{(i,I)}$ for $1 \leq i \leq \underline{n}_I$ in the form of a ciphertext $\underline{e}_I$ that is only decryptable by the designated inspector I on grounds $\underline{g}_I$. To obtain the plaintext values, S provides I with the ciphertext, the decryption grounds, and evidence for these grounds (e.g., a damaged book). I checks by means of the evidence whether the grounds are indeed fulfilled and (only) then decrypts $\underline{e}_I$ with his secret key sk to the originally encrypted values $\langle v_1, \ldots, v_{\underline{n}_I} \rangle \leftarrow \mathit{vDecrypt}(\underline{e}_I, sk, \underline{g}_I)$. Assuming that I shares these values with S and that the user who initially provided the cipertext was known under identifier X, the knowledge state of S after this transition is: $\mathfrak{K}'_S(X) = \mathfrak{K}_S(X) \wedge \underline{C.a}_{(1,I)} == v_1 \wedge \cdots \wedge \underline{C.a}_{(\underline{n}_I,I)} == v_{\underline{n}_I}$. This models that S learns the plaintext values of the verifiably encrypted attributes.

5.7 Advanced Credential Issuance

Now we look at issuance transaction between a user and an issuer that are based on more advanced issuance policies than the one described in Section 5.2. In particular, we now consider credential templates that may also contain key-binding information and carried-over attributes.

Let $Q = \langle P, M \rangle$ be an issuance policy that consists of a presentation policy P and credential template M, where P contains the information as specified in Section 5.5, and M contains, from an abstract point of view, i.e. without considering its form in XML notation, the following information:

- Type τ to be issued
- Issuer I
- Attributes q_i to carry over from $C.a_{(i,I)}$ for $1 \leq i \leq n_I$ where n_I is the number of attributes to carry over
- Attributes $r_1, \ldots, r_{n_r}$ to establish with joint random values
- Optional key-binding either (1) to credential B, or (2) to the i-th pseudonym for $1 \leq i \leq k$, or (3) to the i-th scope-exclusive pseudonym for $1 \leq i \leq \ell$

Further, let T be a presentation token containing the information as specified in Section 5.4, that can be fulfilled by U under interpretation $\mathcal{I}$. An advanced issuance transition between a user U and an issuer I is performed when U proves the presentation token T to I under interpretation $\mathcal{I}$ such that the proved T fulfills the presentation policy P that is contained in Q. As for basic issuance, the transition has effects on the user's credential portfolio as well as the issuer's knowledge state.

The set of credentials that U owns is augmented in the same way as this is the case for basic issuance, where the newly issued credential N has the following *additional* properties:

- $N(q_i) = \big(C.a_{(i,I)}\big)^{\mathcal{I}}$ for $1 \leq i \leq n_I$
- $N(r_i)$ is a jointly generated random number for $1 \leq i \leq n_r$
- in case of a key-binding with a credential:
  $N(uKey) = (B.uKey)^{\mathcal{I}}$
- in case of a key-binding with a (scope-exclusive) pseudonym:
  $N(uKey) = s$, where s is the user key that underlies $\underline{p}_i$ or $\underline{d}_i$ from the token T, respectively

The issuer's knowledge increase of the advanced issuance transaction is a conjunction of:

1. the knowledge increase that we specified for token presentation transitions in Section 5.4,
2. the knowledge increase that we specified for basic issuance in Section 5.2,
3. the predicate $\bigwedge_{i=1}^{n_I} N.q_i \;==\; C.a_{(i,I)}$, which models that I knows that the carried-over attributes are equal to its source attributes, however, I does not learn the values of those attributes,
4. the predicate $\bigwedge_{i=1}^{n_r} \textit{jointlyRandom}(N.r_i)$, which models that I knows which attributes have jointly random generated values (expressed by a dedicated boolean function *jointlyRandom*$(C.a)$ for attribute variable $C.a$), but without I learning these values,
5. in case of a key-binding with a credential: the predicate
   $N.uKey == B.uKey$, which models that I learns that the user keys of the new and the source credentials are equal, however, I does not learn the key itself,
6. in case of a key-binding with a pseudonym: the predicate
   verifyNym$(\underline{p}_i, N.uKey)$, which models that I learns that the new credential is bound to the same user key that underlies $\underline{p}_i$, however, I does not learn the key itself,
7. in case of a key-binding with a scope-exclusive pseudonym: the predicate *verifySeNym*$(\underline{d}_i, \underline{w}_{d_i}, N.uKey)$, which models that I learns that the new credential is bound to the same user key that underlies $\underline{d}_i$ for scope $\underline{w}_{d_i}$, however, I does not learn the key itself.

5.8 Knowledge-Based Policy Verification

In Section 5.5 we show how a verifier determines whether a given presentation policy is fulfilled by the information proved in a given presentation token. Rather than considering merely the information provided in one single token for verifying the policy, a verifier S may also consider all the accumulated knowledge $\mathfrak{K}_S$ about all known pseudonyms for the verification. This flexible approach allows a user to prove just parts of a policy while still being granted access due to prior token presentations. As knowledge states are indirectly *time stamped* with epochs, verifiers can choose to disregard—or forget—knowledge before a certain epoch. The presentation policy language can accordingly be extended with a *switch* stating whether all of the verifier's accumulated knowledge is considered for policy verification or only the information provided in the current token. However, we leave the formalization of such sophisticated policy verification approach for future work.

6 Security Discussion

For our framework to be useful, the various parties need to be given security guarantees: a verifier wants to be sure that if a presentation token verifies then all the statements made in it are indeed supported by the issuer and have not been revoked. Furthermore, the verifier wants to be sure that inspection will succeed when needed. The issuer wants to have similar guarantees w.r.t. issuance tokens. The user wants assurance that her privacy is maintained, i.e., that no more information is leaked than what she willingly released in a presentation token.

Formally proving that such guarantees are fulfilled, provided the underlying cryptography is sound, is far beyond the scope of this paper and is the topic of future work. For the individual cryptographic building blocks, security proofs are given in the literature and Camenisch and Lysyanskaya give a proof for their credential system [18]. However, already proving the security for the different combination of the building blocks has not been done, although there is no reason to believe that such combinations would not be secure. Thus, the first step in proving security of our language framework is to provide precise security notions that capture the high-level security properties stated above and to prove that the composition of the cryptographic building blocks as imposed by our languages achieves those. Next, one would have to show that the mapping of our languages to the concrete crypto realizations is sound. Only then one could attempt to formally prove that the security guarantees as stated above are fulfilled.

7 Conclusion

We presented a language framework enabling a unified deployment of Privacy-ABC technologies, in particular, of U-Prove and Identity Mixer. Our framework improves upon the state of the art [37, 22] by covering the entire life-cycle of Privacy-ABCs, including issuance, presentation, inspection, and revocation, and by supporting advanced features such as pseudonyms and key binding. The framework offers a set of abstract concepts that make it possible for application developers to set up a Privacy-ABC infrastructure and to author policies without having to deal with the intricacies of the underlying cryptography. We demonstrate the soundness of our languages by providing a formal semantics that specifies the effects of issuing, presenting, verifying, inspecting, and revoking credentials on the user's credential portfolio and on the knowledge states of the involved parties.

The proposed language framework has been implemented as part of the ABC4Trust project, where it will be rolled out in two pilot projects. Preliminary tests indicate that our language framework adds a noticeable but reasonable overhead to the cryptographic routines, comparable to the overhead incurred by, for example, XML Signature [45] with respect to the underlying signing algorithm.

Our language framework supports a number of different authentication mechanisms including the mentioned privacy-preserving ones but also standard mechanisms such as X.509. However, most of them will not support the full set of features but we are currently working on a protocol framework that allows the combination of different cryptographic mechanisms to address this.

Acknowledgements

The authors thank Bart De Decker for his invaluable comments on the formalization of the language semantics, Lan Nguyen for his comments on the design of the revocation framework, and Robert Enderlein for reviewing the XML schemas and for suggesting changes to simplify the implementation of the framework. The research leading to these results was supported by the European Commission under the Grant for the ABC4Trust project.

References

[1] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 52–62, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.

[2] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security*, 16(4):369–397, 2008.

[3] Claudio A. Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. Exploiting cryptography for privacy-enhanced access control. *Journal of Computer Security*, 18(1):123–160, 2010.

[4] Man Ho Au, Willy Susilo, and Yi Mu. Constant-size dynamic k-TAA. In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125, Maiori, Italy, September 6–8, 2006. Springer, Berlin, Germany.

[5] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[6] 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, Illinois, USA, November 2011. ACM Press.

[7] P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[8] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444, Bruges, Belgium, May 14–18, 2000. Springer, Berlin, Germany.

[9] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael K. Reiter. Consumable credentials in linear-logic-based access-control systems. In *ISOC Network and Distributed System Security Symposium – NDSS 2007*, San Diego, California, USA, February 28 – March 2, 2007. The Internet Society.

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

[11] 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, *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 400–415, Townsville, Australia, July 2–4, 2007. Springer, Berlin, Germany.

[12] J. Camenisch, I. Krontiris, A. Lehmann, G. Neven, C. Paquin, K. Rannenberg, and H. Zwingelberg. Architecture for attribute-based credential technologies.

Technical Report D2.1, ABC4Trust project deliverable, 2011. Available from `https://abc4trust.eu`.

[13] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252, Melbourne, Australia, December 7–11, 2008. Springer, Berlin, Germany.

[14] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash (extended abstract). In Roberto De Prisco and Moti Yung, editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 141–155, Maiori, Italy, September 6–8, 2006. Springer, Berlin, Germany.

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

[16] 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, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

[17] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. Solving revocation with efficient update of anonymous credentials. In Juan A. Garay and Roberto De Prisco, editors, *SCN 10: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 454–471, Amalfi, Italy, September 13–15, 2010. Springer, Berlin, Germany.

[18] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.

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

[20] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, September 12–13, 2002. Springer, Berlin, Germany.

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

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

[23] 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 17–21, 2003. Springer, Berlin, Germany.

[24] Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. IBM Research Report RZ 3419, IBM Research Division, May 2002.

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

[26] Credentica. U-Prove SDK overview: A Credentica white paper. `http://www.credentica.com/files/U-ProveSDKWhitepaper.pdf`, 2007.

[27] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, December 1–5, 2002. Springer, Berlin, Germany.

[28] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431, Les Diablerets, Switzerland, January 23–26, 2005. Springer, Berlin, Germany.

[29] David Ferraiolo and Richard Kuhn. Role-based access control. In *Proc. of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[30] 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'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Germany.

[31] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of authorization and knowledge. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS 2006: 11th European Symposium on Research in Computer Security*, volume 4189 of *Lecture Notes in Computer Science*, pages 297–312, Hamburg, Germany, September 18–20, 2006. Springer, Berlin, Germany.

[32] 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`.

[33] M. Kirkpatrick, G. Ghinita, and E. Bertino. Privacy-preserving enforcement of spatially aware rbac. *IEEE Transactions on Dependable and Secure Computing*, PP(99):1, 2011.

[34] 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.

[35] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 46–57, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press.

[36] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, February 2003.

[37] Microsoft Corp. Microsoft U-Prove Community Technology Preview R2. `https://connect.microsoft.com/site1188`, August 2011. (accessed: 10 January 2012).

[38] 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, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 463–480, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

[39] 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 14–18, 2005. Springer, Berlin, Germany.

[40] 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.

[41] Federica Paci, Ning Shang, Kevin Steuer Jr., Ruchith Fernando, and Elisa Bertino. VeryIDX - A privacy preserving digital identity management system for mobile devices. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, MDM '09, pages 367–368, Washington, DC, USA, 2009. IEEE Computer Society.

[42] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Berlin, Germany.

[43] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[44] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[45] Satoshi Shirasuna, Aleksander Slominski, Liang Fang, and Dennis Gannon. Performance comparison of security mechanisms for grid services. In *Proc. of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 360–364, Washington, DC, USA, 2004. IEEE Computer Society.

[46] Anna Cinzia Squicciarini, Abhilasha Bhargav-Spantzel, Elisa Bertino, and Alexei B. Czeksis. Auth-SL - a system for the specification and enforcement of quality-based authentication policies. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 07: 9th International Conference on Information and Communication Security*, volume 4861 of *Lecture Notes in Computer Science*, pages 386–397, Zhengzhou, China, December 12–15, 2007. Springer, Berlin, Germany.

[47] Eric R. Verheul. Self-blindable credential certificates from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 533–551, Gold Coast, Australia, December 9–13, 2001. Springer, Berlin, Germany.

[48] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *ACM Workshop on Formal Methods in Security Engineering (FMSE)*, pages 45–55. ACM, 2004.

[49] W.H. Winsborough, K.E. Seamons, and V.E. Jones. Automated trust negotiation. In *Proc. of the DARPA Information Survivability Conference and Exposition (DISCEX)*, volume 1, pages 88 –102 vol.1, 2000.

Publication

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, Illinois, USA, November 2011. ACM Press.

Contributions

- Principal author.
- Claim and evidence handling, claim language, claim building blocks, implementation.

Copyright

A Comprehensive Framework for Data-Minimizing Authentication

P. Bichsel, J. Camenisch, and F.-S. Preiss

IBM Research – Zurich, Switzerland

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 w.r.t. 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.

Keywords: 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, e.g., 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, i.e., 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, e.g., 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 w.r.t. 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, e.g., in an database. If the user wants to release any of her attribute values, she relates the relying party to her issuer, i.e., 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, i.e., 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 w.r.t. 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.

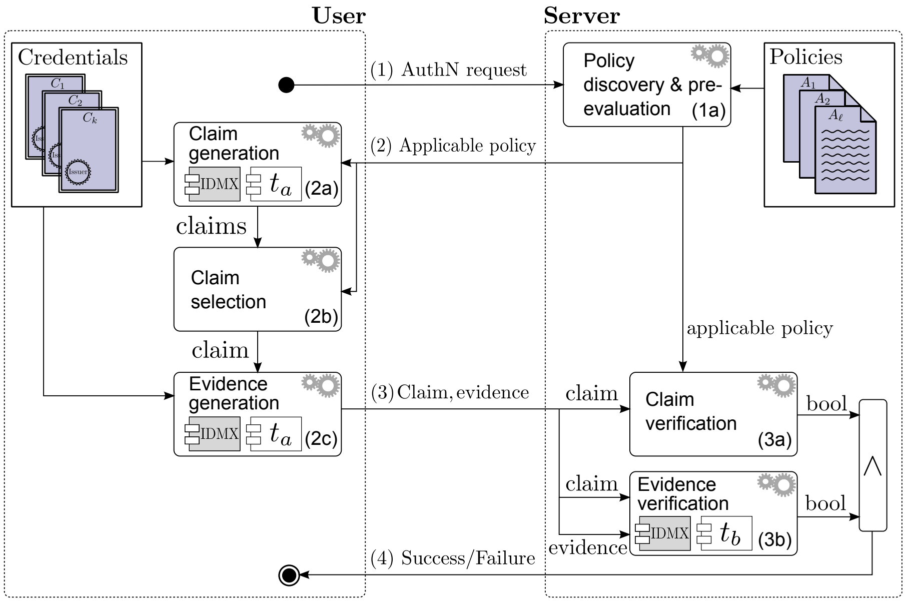


Figure 1: Data-Minimizing Authentication

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. Having received the policy, the user's system determines which *claims*, i.e., 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 w.r.t. 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:

```
own dl::DriversLicense issued-by DeptMotorVehicles
own mc::MemberShipCard issued-by CarRentalCo
own cc::CreditCard issued-by Amex, Visa
own li::LiabilityInsurance issued-by InsuranceCo
reveal cc.number
reveal li.pNo to EscrowAgent under 'in case of damage'
where dl.issueDate ≤ dateSubtrDuration(today(), P3Y) ∧
      li.guaranteedAmoutUSD ≥ 30.000 ∧
      (mc.status == 'gold' ∨ mc.status == 'silver') ∧
      dl.name == li.name
sign '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 (3) 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 w.r.t. 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, i.e., it provides unlinkability when using a credential multiple times *only* w.r.t. 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 w.r.t. to the issuer that an online credential can never provide. However, when it comes to the privacy w.r.t. 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, i.e., 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	✓		$\checkmark_{(+)}$	✓
Disclosure to Third-Parties	✓		$\checkmark_{(+)}$	✓
Limited Spending	✓		$\checkmark_{(+)}$	$\checkmark_{(+)}$
Cross-Credential Proofs	(✓)		$\checkmark_{(+)}$	✓
Unlinkable Multi-Use	(✓)			✓

Table 1: Certification technology feature comparison. ✓: Supported. (✓): Limited support. $\checkmark_{(+)}$: 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, e.g., 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, i.e., 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, i.e., 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 w.r.t. 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:

i own $dl::DriversLicense$ issued-by DeptMotorVehicles
i own $mc::MemberShipCard$ issued-by CarRentalCo
i own $cc::CreditCard$ issued-by Visa
i own $li::LiabilityInsurance$ issued-by InsuranceCo
where $dl.issueDate \leq dateSubtrDuration(today(), P3Y) \wedge$
$\quad li.guaranteedAmoutUSD \geq 30.000 \wedge$
$\quad mc.status ==$ ‘silver’ $\wedge$ $dl.name == li.name \wedge$
$\quad cc.number ==$ ‘1234 5678 9012 3456’
i reveal $li.pNo$ to EscrowAgent under ‘in case of damage’
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, e.g., 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), it has to be determined which claims can be made w.r.t. her credential portfolio $\mathfrak{P}_u$. To do so, the possible options of assigning credentials from $\mathfrak{P}_u$ to all of the credential variables occurring in the policy are calculated.

```
    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

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 of 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 w.r.t. 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), i.e., $\bigvee_i \bigwedge_j \ell_{ij}$, 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 $c{::}\mathit{CreditCard}$ 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, i.e., 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, i.e., misuse the verifiable encryption. While the claim language allows for disclosing a list of terms, i.e., 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, i.e., $a.b - c.d \neq 0, a.b - \mathsf{ConstExp} \neq 0$. 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 w.r.t. 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, i.e., $a.b - c.d$. 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 ($\mathsf{NonConstExp}$) 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, e.g., $a.b - 2c.d$, 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, i.e., 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 James Joshi and Barbara Carminati, editors, *Proc. of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 119–128, Pittsburgh, Pennsylvania, 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.

Publication

Downstream Usage Control

Publication Data

Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Downstream usage control. In *Proc. of the 11th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 22–29, Fairfax, Virginia, USA, July 2010. IEEE Computer Society Press.

Contributions

- Principal author.
- Language model, matching semantics.

Copyright

Downstream Usage Control

L. Bussard[1], G. Neven[2], and F.-S. Preiss[2]

[1] European Microsoft Innovation Center, Germany
[2] IBM Research – Zurich, Switzerland

Abstract. Whereas access control describes the conditions that have to be fulfilled *before* data is released, usage control describes how the data has to be treated *after* it is released. Usage control can be applied to digital rights management, where the data are usually copyright-protected media, as well as in privacy, in which case the data are privacy-sensitive personal information. An important aspect of usage control for privacy, especially in light of the current trend towards composed web services (so-called *mash-ups*), is *downstream usage*, i.e., with whom and under which usage control restrictions data can be shared. In this work, we present a two-sided XML-based policy language: on the one hand, it allows users to express in their preferences in a fine-grained way the exact paths that their data is allowed to follow, and the usage restrictions that apply at each hop in the path. On the other hand, it allows data consumers to express in their policies how they intend to treat the data, with whom they intend to share it, and how the downstream consumers intend to treat the data.

Keywords: Privacy, policy languages, usage control, data sharing.

1 Introduction

Many web services today are so-called service *mash-ups*. A mash-up is a service that acts as a front-end for a composition of multiple subservices that are offered by different companies. For example, a travel booking mash-up may offer an integrated interface to book flights, hotels, and rental cars. In the background, however, it invokes the web service APIs of different specialized airline, hotel, and car rental subsidiaries to collect offers. The best offers are presented to the user, who selects an offer, enters her booking and payment information, and confirms to let the mash-up make all the bookings for her through the subsidiaries' APIs.

Service mash-ups are important for leveraging online service APIs to create new functionality, but pose significant privacy risks for their users. The users cannot keep track of who stores what information about them, and often they do not even know the subsidiaries their data is shared with.

To overcome this problem, service providers publish their privacy policies to inform the users about how the gathered data is used. Human-readable privacy policies, e.g., have the disadvantage of mostly being written in complex language influenced by the legal profession and thus being ignored by users. Even if clear privacy policies are presented, they often remain vague about the sharing of information with third parties. For example, the privacy policy of Expedia.com[3], a popular online travel service, mentions the following with regard to sharing personal information with suppliers:

> *We do not place limitations on our suppliers' use or disclosure of your other personal information* [i.e., other than the email address]*. Therefore, we encourage you to review the privacy policies of any travel supplier whose products you purchase through this site.*

Machine-interpretable privacy policy languages such as EPAL [2] and P3P [17] is a promising approach, in particular when used in combination with a privacy preference language such as APPEL [8]. In the latter language, users can express how they expect their data to be treated, so that an automated or semi-automated matching procedure can decide on the acceptability of a proposed P3P policy.

Unfortunately, EPAL and P3P are both rather constrained in expressivity regarding sharing personal information with third parties, or *downstream usage* as we call it here. EPAL leaves the definition of specific actions and obligations up to enterprise-defined vocabularies, and is hence silent about downstream usage. Support in P3P is limited to specifying which of six classes of third-party recipients the information will be shared with; it is up to the server to classify his recipients into one or more of the classes.

In this work, we investigate how to structure a policy language suitable for downstream usage control. The difficulty here is that downstream usage control involves a mixture of what is typically considered access control (who is allowed to receive the data, e.g., by roles or owned credentials) and usage control (how is the recipient supposed to treat the data, e.g., usage purposes or retention period). We consider the most general setting here, where the user states in her privacy preferences, for each hop in a chain of downstream recipients, how they have to treat her data and to whom they can further forward it. At the same time, each recipient specifies in his privacy policy how he intends to treat the data and to whom he intends to forward it. We propose XML-based languages to express both the user's preferences and the servers' policies, and describe an automated matching algorithm to determine whether the proposed policies are allowed by the specified preferences. The user and the servers can specify downstream usage restrictions up to any number of hops (not necessarily the same number). Optionally, they can either specify that the last restrictions in the chain are valid for all subsequent hops, or that after that hop no further downstream usage is allowed. Moreover, for situations where the server does not

[3]cf. `http://www.expedia.com/daily/service/privacy.asp`

know at the time of data collection to whom he may forward the data, we propose an alternative matching algorithm at which a server declares his willingness to impose any restrictions on further downstream usage that the user may specify.

This work relies on the trust model of P3P and EPAL: each service is willing to enforce its privacy policy. Authors assume that service providers are appropriately enforcing their policies to protect their reputation and/or that services are regularly audited and certified by trusted third parties. Proving the correctness of policy enforcement, for instance using a trusted stack (certified TPM, trusted OS, and trusted application), is out of the scope of this paper.

2 Related Work and Contributions

Our work is closely related to rights expression languages (RELs), privacy policy languages, and usage control. We give a brief overview of each of these lines of work and how they relate to the language we propose.

Rights Expression Languages Even if scenarios and trust models are different, there are clear similarities between digital right management (DRM), enterprise right management (ERM), and privacy policies as described in this paper. Indeed, in each case, a data provider attaches constraints, in the form of a license or a sticky policy, to data sent to a data consumer. The domain-specific vocabularies for privacy policies and RELs may be rather different, but the same overall language structure can be used for both.

There are three main differences between our approach and the state of the art ERM and DRM languages MPEG-21 REL [18], XrML [7], and ODRL [13]. First, in RELs it is the data provider who unilaterally creates the license and imposes it onto the consumer, without any kind of matching procedure. In privacy settings, the provider (i.e., the user) usually does not have such power. Second, RELs focus mainly on rights (e.g., play, print) and constraints on these rights (e.g., temporal, fees, device), but obligations remain underspecified. Third, the constraints on downstream data sharing (e.g., sell, lease, share) are not nearly as expressive as in our model.

Privacy Policy Languages In the introduction we already discussed the shortcomings of the privacy policy languages EPAL [2], P3P [17], and APPEL [8] when it comes to expressing restrictions on downstream usage. SecPAL for Privacy (S4P) [4] is a logic-based language to specify privacy policies and preferences. In S4P, matching is about evaluating queries with a set of assertions, while in the work presented in this document, matching is done by comparing statements. Ardagna et al. [1] describe a data handling policy language that allows for specifying data recipients, usage purposes and obligations. The data recipient can specify global rules on who may receive the data, but transitive downstream usage restrictions cannot be expressed.

Usage Control The subject of usage control [14, 9] is concerned with how data is treated *after* its release. The Obligation Specification Language (OSL) [10] supports a wide range of usage control requirements related to time, cardinality, purpose, and events. OSL is logic-based, so that a sequence of events can automatically be checked for compliance with the specified policy. The OSL extensions for policy evolution [15] target a use case similar to ours as they allow for expressing which consumers (indicated by their roles) have to adhere to which rights and duties. However, in OSL the data provider unilaterally creates the policy to be adhered to, whereas in our language it is the result of matching a consumer's policy against the provider's preferences. Moreover, in our language one can describe the full downstream path that the data is allowed to follow, including who can share it with whom, and how many hops it can take. For example, in our language a patient could impose one usage control policy when a health insurance company obtains her medical record through the hospital, and another policy when it obtains the record from the patient directly. In OSL both cases are treated the same.

Provenance Access Control As highlighted in [11], we consider that metadata associated with personal data may also be subject to usage control. For instance, the obligation of notifying the user may contain an e-mail address.

3 Description of Solution

3.1 Example Scenario

Alice is a privacy-aware user who regularly shops online, but who is concerned about what happens to the data that she provides about herself. For example, she's willing to provide her postal address to online shops so that the goods can be delivered, but she realizes that most shops rely on external shipping companies. She wants to impose a detailed set of usage control preferences though, where the restrictions on the shipping company depend on who the front-end service is from which it obtained the address. Namely, when obtained through a book shop, she is fine with the shipping company using her address for statistics, but when obtained through any other store, which may include liquor or lingerie stores, she is not. More precisely, she wants to enforce the following preferences:

- Book shops can collect her address for the purpose of statistics, contact, and account administration. They must delete it after two years and are allowed to forward to shipping companies who can use it for shipping and statistics, have to delete it after two weeks, and can further forward it to shipping companies under the same restrictions.
- Any shop can collect her address for the purpose of account administration provided they delete it within one year. They can further forward it to

shipping companies who can use it for shipping only, have to delete it after two weeks, and are not allowed to share the data with anyone.

Alice regularly buys books at the online book shop bookshop.com because its privacy policy matches her preferences. Namely, bookshop.com states that it will

- use her address for statistics and account administration, that it will delete the address after one year, and that it will forward the address only to shipping.com.

The policy of shipping.com states that

- the address will be used for shipping and statistical purposes, and will be deleted after one week.

When buying a book at bookshop.com, Alice can safely give her address away since bookshop.com's and shipping.com's privacy policies match her own preferences. However, when buying a bottle of wine at liquor.com, who also use shipping.com for shipping, the transaction will be refused, as Alice's preferences in this case do not allow shipping.com to use her address for statistical analysis.

3.2 Language Model

The abstract scenario we consider is one where two parties, typically a user and a server, engage in an interaction where one of the parties, typically the server, requests some personally identifiable information (PII) from the other party. We will from now on call the party that provides the data the *data provider* and the party that requests the data the *data consumer*. Moreover, we consider a scenario where at a later point in time, the data consumer may want to forward the PII to a third party, called the *downstream data consumer*.

Both the data provider and the data consumer have their own policies expressing the required and proposed treatment of the PII, respectively. These policies contain access control and usage control policies. A piece of PII is only sent to a data consumer after (1) the access control requirements have been met, and (2) a suitable usage control policy has been agreed upon.

We distinguish three kinds of policies:

- **Preferences:** In his *preferences* the data provider describes, for specific pieces of PII, which access control requirements a data consumer has to satisfy in order to obtain the PII, as well as the usage control requirements according to which the PII has to be treated after transmission. These requirements may include *downstream usage requirements*, meaning the requirements that a downstream data consumer has to fulfill in order to obtain the PII from the (primary) data consumer.

- **Policy:** The *policy* is the data consumer's counterpart of the data controller's preferences. In a policy the data consumer contains, for specific pieces of PII to be obtained, his certified properties (roles, certificates, etc.) that can be used to fulfill access control requirements, and a usage control policy describing how he intends to use the PII.

- **Sticky policy:** The *sticky policy* describes the mutual agreement concerning the usage of a transmitted piece of PII. This agreement is the result of a matching process between a data providers's preferences and a data consumer's policy. Technically a sticky policy is quite similar to preferences as described above, but it describes a mutual agreement between the data provider and the data consumer that cannot be changed. After receiving the PII, the data consumer is responsible for storing and enforcing the sticky policy.

To illustrate our ideas we employ a simple XML-based language to express preferences, policies, and sticky policies. In the following we first introduce our language and then focus on how to express downstream usage requirements. Note that we provide the full language schemas in an extended version of this paper [6].

Preferences model

Figure 1 shows Alice's preferences for book shops expressed in our policy language. The `Preferences` root element contains multiple `Preference` elements, each describing to which PII it applies and what the respective access and usage controls are. An attribute `sticky` indicates whether these preferences are in fact a sticky policy for the PII (cf. 3.2), acting as an explicit reminder that they cannot be changed. Alternatively, one could keep all sticky policies separately in a read-only policy store. A `Preference` can refer to the applicable PII by their data type, meaning that the `Preference` applies to all PII of this type, or by a unique identifier pointing to a single instance of PII. We assume that a typing mechanism and unique naming scheme for PII are in place. A complete language would probably offer more powerful mechanisms to specify applicability, allowing for example attribute expressions or temporal constraints.

Our language is strictly limited to positive statements, in the sense that it explicitly lists the permitted information exchanges, and assumes that all other exchanges are forbidden. Apart from this being a safe privacy-conservative choice, it also simplifies the matching procedure. However, it means that one cannot express conditions of the form "do not forward to X" or "do not use for purpose Y".

If multiple `Preference` elements apply to a single piece of PII, then satisfying the conditions in either of them results in a match. In other words, `Preference`s are combined according to "or" semantics. This makes it possible to define more permissive exceptions to general preferences.

Within a `Preference`, access and usage control requirements occur in a pair enclosed in an `ACUC` element. A pair (AC, UC) means that any data consumer satisfying access requirements AC can obtain the PII when adhering to usage

```
<Preferences>
 <Preference>
  <Applicability>
   <DataType> Address </DataType>
  </Applicability>
  <ACUC>
   <AccessControl>
    <Rule>CertifiedAsBy{bookshop, CAx}</Rule>
   </AccessControl>
   <UsageControl>
    <Rights>
     <UseDownstream allowLazy="false">
      <ACUC id="ACUCshipping@alice">
       <AccessControl>
        <Rule>CertifiedAsBy{shipping, CAy}</Rule>
       </AccessControl>
       <UsageControl>
        <Rights>
         <UseDownstream allowLazy="false">
          <ACUC reference="ACUCshipping@alice"/>
         </UseDownstream>
         <UseForPurpose>statistics</UseForPurpose>
         <UseForPurpose>shipping</UseForPurpose>
        </Rights>
        <Obligations>
         <DeleteWithin>P14D</DeleteWithin>
        </Obligations>
       </UsageControl>
      </ACUC>
     </UseDownstream>
     <UseForPurpose>statistics</UseForPurpose>
     <UseForPurpose>contact</UseForPurpose>
     <UseForPurpose>accountadmin</UseForPurpose>
    </Rights>
    <Obligations>
     <DeleteWithin>P2Y</DeleteWithin>
    </Obligations>
   </UsageControl>
  </ACUC>
 </Preference>
 ...
</Preferences>
```

Figure 1: Excerpt from Alice's preferences.

requirements UC. To allow multiple AC/UC combinations for a single piece of PII, one can use multiple `Preference` elements with the same `Applicability`.

An optional attribute `id` assigns a unique identifier to an `ACUC` pair. This identifier can be referred to from another `ACUC` element via a `reference` attribute. The referring element is then interpreted as if it was substituted with the referred element.

Access control requirements are expressed in terms of `Rule`s where each rule specifies a *property* that a data consumer must have in order to be granted access.

Properties are stated in terms of attributes as certified by some certification authority (CA). Empty access control requirements mean that anybody who commits to fulfilling the usage control requirements is granted access. The simple access control language that we use here could in a real system, e.g., be substituted with a complete role-based [16], attribute-based [5, 12], or logic-based [3] access control language.

Usage control requirements are expressed by distinguishing between `Rights` and `Obligations`. A right states an action that the data consumer is allowed to perform on the data, but doesn't have to perform to comply with the policy. An obligation states an action that a data consumer is obliged to perform. We model two types of rights and two types of obligations here:

- `UseDownstream`: The right to forward the PII under given conditions to further data consumers. This is a crucial element in our policy language; we come back to its exact structure and meaning later.
- `UseForPurpose`: The right to use the PII for a specific purpose.
- `DeleteWithin`: The obligation to delete the PII within a given amount of time.
- `NotifyOnAccess`: The obligation to notify the user when the PII is accessed.

The complete language supports more rights and obligations, however, those four suffice to illustrate the ideas of this work. Multiple rights and obligations within a `UsageControl` element are combined by "and" semantics, meaning that the data consumer obtains all the specified rights and has to adhere to all of the specified obligations.

Policy model

A data consumer's policy states which usage control requirements he is willing to adhere to when requesting a specific resource from a data provider. In addition the policy states the properties the data consumer is willing to disclose for fulfilling the data provider's access control rules for that resource.

The language schema of the server policy resembles the schema of the preferences, exhibits the following differences though. The top-level elements are `Policies` and `Policy`, where the `Policy` has no `sticky` attribute; the `AccessControl` element contains the properties in `Property` elements rather than access requirement rules; and the `ACUC` element in the `UseDownstream` right has different cardinality as explained later. In the following we provide example policies for the book shop and the shipping company.

The shop's policy stated in Figure 2 expresses that for collecting addresses, it is willing to authenticate as a shop or a book shop certified by CAx, and as bookshop.com certified by CAz. The address will be deleted after one year and

```
<Policies id="Policies@Shop">
 <Policy>
  <Applicability>
   <DataType> Address </DataType> </Applicability>
  <ACUC id="ACUCaddress@Shop">
   <AccessControl>
    <Property>CertifiedAsBy{bookshop, CAx}</Property>
    <Property>CertifiedAsBy{shop, CAx}</Property>
    <Property>CertifiedAsBy{bookshop.com, CAz}</Property>
   </AccessControl>
   <UsageControl>
    <Rights>
     <UseDownstream allowLazy="false">
      <ACUC reference="ACUCaddress@Shipping"/>
     </UseDownstream>
     <UseForPurpose>statistics</UseForPurpose>
     <UseForPurpose>accountadmin</UseForPurpose>
    </Rights>
    <Obligations>
     <DeleteWithin>P1Y</DeleteWithin>
    </Obligations>
   </UsageControl>
  </ACUC>
 </Policy>
 ...
</Policies>
```

Figure 2: Excerpt from bookshop.com's policies.

used for statistics and account administration. Further, the shop wants to be able to forward it under the policy of the shipping company (that is specified below).

Figure 3 depicts shipping.com's relevant policies. When collecting addresses, it is willing to authenticate as a shipping company certified by CAy and as shipping.com certified by CAz. It intends to use the address for shipping and statistical purposes and will delete it within one week.

Sticky policy model

Sticky policies follow the same schema as preferences, but have the `sticky` attribute in the `Preference` element set to true. This is to indicate that this sticky policy originates from another party and must thus not be modified. Note that a data consumer may, in addition to the sticky policy, also have own preferences for forwarding previously received PII. Those preferences can, however, be changed and are therefore not sticky.

3.3 Downstream usage

The crucial aspect of our policy language is that it allows both the data provider and the data consumer to express to whom and under what conditions PII can or

```
<Policies id="Policies@Shipping">
 <Policy>
  <Applicability>
   <DataType> Address </DataType> </Applicability>
  <ACUC id="ACUCaddress@Shipping">
   <AccessControl>
    <Property>CertifiedAsBy{shipping,CAy}</Property>
    <Property>CertifiedAsBy{shipping.com,CAz}</Property>
   </AccessControl>
   <UsageControl>
    <Rights>
     <UseForPurpose>shipping</UseForPurpose>
     <UseForPurpose>statistics</UseForPurpose>
    </Rights>
    <Obligations>
     <DeleteWithin>P7D</DeleteWithin>
    </Obligations>
   </UsageControl>
  </ACUC>
 </Policy>
 ...
</Policies>
```

Figure 3: Excerpt from shipping.com's policies.

will be forwarded. These conditions are expressed in `UseDownstream` elements.

We first focus on `UseDownstream` elements occurring in the data provider's preferences. Each `UseDownstream` element contains exactly one `ACUC` child element. This `ACUC` element either contains a fully specified pair of access and usage control requirements, or another `ACUC` element is referenced with the `reference` attribute.

In the former case, the access control requirements specify to which downstream data consumers the PII can be forwarded, while the usage control requirements specify how these downstream consumers are supposed to treat it. Usage requirements can on their turn also contain `UseDownstream` elements that specify to whom and under what conditions the downstream consumer can further forward the PII, which on their turn can contain `UseDownstream` elements as well, etc. This mechanism enables the data provider to restrict the forwarding of his PII up to an arbitrary number of "hops". We refer to this approach as *nested* downstream usage control.

In the case where an `ACUC` element references the content of another `ACUC` element, for the sake of simplicity we insist that it can only refer to its closest ancestor `ACUC` element, i.e., the ancestor four levels higher in the XML tree. (We impose this restriction since it simplifies the matching procedure and there seem to be no convincing use cases for "cyclically recursive" policies with cycle length greater than one.) This means that the data consumer can then forward the PII under the same restrictions that were imposed on himself. We therefore call this approach *recursive* usage control. Note that our policy language allows to combine nested and recursive usage control by defining a chain of nested usage controls for the first number of hops and a final recursive usage control for any further hops.

In a data consumer's policy, each `UseDownstream` element contains *at most* one `ACUC` element. If present, it contains a set of properties describing to whom he plans to forward the PII, and a usage control policy describing how that downstream consumer will treat the data. This usage control policy could contain further `UseDownstream` elements, describing the next hops up to an arbitrary nesting degree. Alternatively, the `reference` attribute can be used to point to another `ACUC` element. This element may even be hosted directly by a downstream consumer (where we assume the `reference` to act as a URL).

In many situations, the downstream consumer or his policy are not be known at the time the PII is transmitted to the primary consumer. Rather than specifying all intended hops in full detail, the data consumer can indicate his willingness to enforce any restrictions imposed by the data provider by setting the `allowLazy` attribute of the `UseDownstream` element to `true` and omitting the `ACUC` element. The matching between the data provider's preferences and the downstream consumer's policy is then done by the primary consumer at the time the PII is forwarded to the downstream consumer. We refer to the next section for more details on lazy matching. When the `allowLazy` attribute occurs in `Preferences`, it indicates whether the data provider allows the restrictions expressed in the child `ACUC` element to be matched lazily.

Finally, in the `UseDownstream` element a `maxDepth` attribute can be set to an integer or to `unbounded` to indicate how often a piece of PII can be forwarded at most. The intended behavior concerning this limit can be explained with a counter contained in sticky policies. The counter in a sticky policy that is attached to forwarded PII is decreased by one w.r.t. the previous sticky policy or w.r.t. `maxDepth` in case the PII is forwarded by the primary data provider.

4 Matching

Given a data provider's preferences and a consumer's policies, matching aims at automating the process of deciding whether the provider can safely transmit a piece of personal data. We introduce a '*more or equally permissive than*' operator to match preferences with policies. We say there is a *match* when the preferences are more or equally permissive than the policy.

4.1 Matching Privacy Preferences and Policies

To explain the matching procedure, we use a set-based representation of the XML structure described in the previous section. A `Preferences` element is represented by a set *Prefs* containing an element $Pref \in Prefs$ for each of its `Preference` child elements in the XML structure. $Pref.App$ represents a set containing all the PII owned by the user that is covered by the `Applicability` element, and

Pref.ACUC designates the contained `ACUC` child elements. The set *ACUC.AC* is the set of access control rules (e.g., `CertAsBy`) contained in the `Rule` elements of the embedded `AccessControl`, while *ACUC.UC* is the set of usage controls in terms of rights (*UC.Rights*) and obligations (*UC.Obls*), specified by the `Rights` and `Obligations` elements, respectively. We use an analogous notation for the consumers' policies.

Intuitively, preferences *Prefs* are more or equally permissive than policies *Pols*, denoted $Prefs \trianglerighteq Pols$, if the access control properties in *Pols* satisfy the rules in *Prefs* and if *Pols* asks for less rights and promises to adhere to stricter obligations than specified in *Prefs*. We "overload" the notation of the $\trianglerighteq$ operator to compare not only preferences with policies, but also to compare rights, obligations, access control policies as well as usage control policies.

To determine if there is a match between preferences *Prefs* and policies[4] *Pols*, it is verified if for each ACUC pair in a policy the user has a corresponding piece of PII with a more or equally permissive ACUC pair:

$$\begin{aligned} Prefs \trianglerighteq Pols \Leftrightarrow \forall Pol \in Pols \cdot \exists Pref \in Prefs \ \cdot \ \exists PII \in PIIs \ \cdot \\ (PII \in (Pol.App \cap Pref.App) \wedge Pref.ACUC \trianglerighteq Pol.ACUC) \end{aligned} \tag{1}$$

Above, *PIIs* is the set of all pieces of personal information that the user possesses, and *PII* can be any specific piece of PII in that set. In the following, we use the notations $*_{Pref}$ and $*_{Pol}$ to denote elements within preferences and policies respectively.

Pairs of access control and usage control policies are matched as follows:

$$\begin{aligned} ACUC_{Pref} \trianglerighteq ACUC_{Pol} \ \Leftrightarrow \ & (ACUC_{Pref}.AC \trianglerighteq ACUC_{Pol}.AC) \ \wedge \\ & (ACUC_{Pref}.UC \trianglerighteq ACUC_{Pol}.UC) \end{aligned} \tag{2}$$

Note that (2) is evaluated multiple times during the evaluation of (1). For example, $ACUC_{Pol}$ is instantiated subsequently with $Pol_i.ACUC$ for all Pol_i in *Pols*. The access control mechanism we employ is based on certified properties such as roles or IDs. To match access control requirements, it is verified if for each `Rule` in the preferences there is a corresponding `Property` in the policy:

$$AC_{Pref} \trianglerighteq AC_{Pol} \ \Leftrightarrow \ \forall r \in AC_{Pref} \cdot \exists r' \in AC_{Pol} \cdot r = r' \tag{3}$$

This could be extended to cover more sophisticated access control mechanisms such as claim-based access control or hierarchical roles. However, the matching becomes more complex when doing so (e.g., as environment attributes such as time of day cannot be pre-evaluated). Usage control requirements are matched as follows:

$$\begin{aligned} & UC_{Pref} \trianglerighteq UC_{Pol} \ \Leftrightarrow \\ & \quad (\forall R \in UC_{Pol}.Rights \cdot \exists R' \in UC_{Pref}.Rights \cdot R' \trianglerighteq R) \ \wedge \\ & \quad (\forall O \in UC_{Pref}.Obls \cdot \exists O' \in UC_{Pol}.Obls \cdot O \trianglerighteq O') \end{aligned} \tag{4}$$

[4]Note that policies *Pols* are not complete privacy policies of a website but rather policies of specific inputs in a Web form or policies of credential attributes required to authenticate.

The matching of rights and obligations is specified for the different types of rights and obligations individually. In the following we give example specifications for the ones introduced in Section 3.2. If obligations R and R' specify that the user must be notified when her PII is accessed, $R' \trianglerighteq R$ is evaluated with the appropriate $\trianglerighteq$ operator, i.e., (6) in this case. Letting obligation `DeleteWithin` with duration t be denoted as $\mathit{DelWithin}(t)$, matching is done as follows:

$$\mathit{DelWithin}(t) \trianglerighteq \mathit{DelWithin}(t') \;\Leftrightarrow\; t \geq t' \;. \tag{5}$$

Letting obligation `NotifyOnAccess` with contact information c be denoted as $\mathit{NotifyOnAcc}(c)$, matching is done as follows (where $*$ represents *any* string):

$$\mathit{NotifyOnAcc}(c) \trianglerighteq \mathit{NotifyOnAcc}(c') \;\Leftrightarrow\; c' = * \;\vee\; c = c' \;. \tag{6}$$

Letting right `UseForPurpose` with purpose p be denoted as $\mathit{UseForPurp}(p)$, matching is done as follows:

$$\mathit{UseForPurp}(p) \trianglerighteq \mathit{UseForPurp}(p') \;\Leftrightarrow\; p = p' \tag{7}$$

For the sake of readability, support for hierarchical purposes is not described here.

As downstream usage is the focus of this paper, the following two subsections explain the details of the matching procedure for downstream usage rights.

4.2 Proactive Matching of Downstream Rights

This section provides the intuition behind proactively matching a downstream structure, i.e., matching structures where both the preferences and the full chain of downstream usage policies are known at the time of matching.

For a given pair $ACUC$, let $|ACUC|$ be the "local" ACUC, meaning containing only those restrictions and obligations that do not affect downstream usage. Using this notation, we can represent the structure of an ACUC policy with downstream usage as a directed graph where each node represents a hop in the downstream usage. Each node is labeled with the local ACUC policy describing how the data are to be treated locally. Each edge represents the permission (in case of a provider's preferences) or intention (in case of a consumer's policy) to forward the data under the restrictions specified by the ACUC of the endpoint of the edge. By the restrictions that we imposed on the connection among ACUC pairs, the structure of the graph is similar to that of a tree where the leaf nodes can optionally have a loop, representing recursion in the downstream usage policy. Intuitively, matching two ACUC pairs $ACUC_{\mathit{Pref}}$ taken from a provider's preferences and $ACUC_{\mathit{Pol}}$ specified in a consumer's policy is done by simultaneously going over the nodes in the two tree representations of $ACUC_{\mathit{Pref}}$ and $ACUC_{\mathit{Pol}}$ and verifying that it is possible to cover each branch of the policy-side tree with a more or equally permissive branch on the preference side.

Letting $\mathit{UseDS}(ACUC)$ denote a `UseDownstream` element with an `ACUC` child element represented by $ACUC$, the matching for downstream usage rights works according to the rule:

$$\mathit{UseDS}(ACUC) \trianglerighteq \mathit{UseDS}(ACUC') \;\Leftrightarrow\; ACUC \trianglerighteq ACUC' \tag{8}$$

We assume that an `ACUC` specifying recursive downstream sharing right with depth d is “folded out” into a graph of $d+1$ nested `ACUC`. Formula (8) is thus sufficient to handle nested as well as recursive access and usage control. Obviously, to improve performance when matching recursive `ACUC` and to avoid infinite loops in case of unlimited recursion depth, the concrete implementation of the matching algorithm must keep track of which combinations of `ACUC` elements have already been matched against one another and must take depth into account. We refer to the extended version [6] for a complete example of a proactive matching process.

4.3 Lazy Matching

In the previous section we focused on *proactive matching*, i.e., matching where all downstream policies are known beforehand. It is, however, not always possible to collect all policies during matching. For this reason, we also introduce *lazy matching*, which only takes into account the properties and policies of the data consumer, but not those of any downstream data consumers. Rather, the data consumer expresses that he is willing to impose whatever usage restrictions on downstream consumers that the data provider may specify.

Both types of matching imply that the sticky policy that the data consumer associates to the data must at least enforce the preferences of the data provider. On one hand, proactive matching allows to minimize the rights and maximize the obligations that are transferred as the matching procedure can already take the downstream consumers and their policies into account. On the other hand, lazy matching offers more flexibility and is the only option in dynamic settings, where either the downstream consumers or their policies are not known yet at the moment of matching, or where the access control policy depends on environment variables that will only be known when the data is actually forwarded.

To support lazy matching, we redefine formula (8) to take the `allowLazy` attribute into account, which is represented by a boolean value *lazy* here. Matching for downstream usage then follows the rule:

$$\begin{aligned} &\mathit{UseDS}(\mathit{lazy}, \mathit{ACUC}) \trianglerighteq \mathit{UseDS}(\mathit{lazy}', \mathit{ACUC}') \;\Leftrightarrow \\ &(\mathit{lazy} \wedge \mathit{lazy}') \;\vee\; (\mathit{ACUC} \trianglerighteq \mathit{ACUC}') \end{aligned} \tag{9}$$

4.4 Creating Sticky Policies

A sticky policy specifies the commitment of the data consumer towards the data provider w.r.t. treatment of her shared data. We envision a sticky policy SP to be produced in case a matching procedure is successful. This sticky policy then never violates the preferences and is compliant with the policy, i.e., $\mathit{Prefs} \trianglerighteq \mathit{SP} \trianglerighteq \mathit{Pols}$ always holds. (Note that we assume that matching preferences with preferences is analog to the matching of preferences with policies.)

A privacy-conservative matching algorithm will choose a sticky policy that is as close as possible to the policy proposed by the consumer. Note that a sticky policy includes only those preferences that contain rights the data consumer gets or obligations he has to adhere to w.r.t. a given piece of PII.

Clearly, when a primary data consumer forwards the data to a downstream consumer, he must also attach a sticky policy. The data consumer may have its own preferences Prefs' regarding data sharing. On top of his own preferences, the data consumer must enforce the sticky policy SP associated to this piece of data. In other words, the primary data consumer has to find a downstream sticky policy SP' so that $\mathit{Prefs}' \trianglerighteq \mathit{SP}' \trianglerighteq \mathit{Pols}'$ and, informally, $\mathit{SP} \trianglerighteq \mathit{SP}' \trianglerighteq \mathit{Pols}'$, where Pols' is the policy of the downstream data consumer. More precisely, the latter evaluation is done by first extracting downstream preferences (Prefs_{DS}) from the sticky policy (SP) as depicted below, where $A \leftarrow B$ denotes the assignment to A of the value of B:

$$\begin{aligned} &\forall \mathit{Pref} \in \mathit{SP} \cdot (\forall R \cdot (R \in \mathit{Pref.ACUC.UC.Rights} \wedge R = \mathit{UseDS}(\cdot,\cdot)) \cdot \\ &\qquad (\mathit{Pref}_{DS}.\mathit{App} \leftarrow \mathit{Pref.App}, \\ &\qquad \mathit{Pref}_{DS}.\mathit{ACUC} \leftarrow \mathit{R.ACUC}, \\ &\qquad \mathit{Prefs}_{DS} \leftarrow \mathit{Prefs}_{DS} \cup \mathit{Pref}_{DS})) \end{aligned} \tag{10}$$

Next Prefs_{DS} is used for matching the downstream policy (i.e., $\mathit{Prefs}_{DS} \trianglerighteq \mathit{Pols}'$) and to create the downstream sticky policy SP' so that $\mathit{Prefs}_{DS} \trianglerighteq \mathit{SP}' \trianglerighteq \mathit{Pols}'$.

5 Conclusion and Future Work

This paper presents a simple yet expressive language to specify privacy policies and user preferences which may express downstream usage requirements. Our paper also describes a matching algorithm that, given a server's privacy policy, helps a user to decide whether her personal data can be shared with the server according to her preferences.

In our language we employ disjunctive semantics when combining multiple preference elements. We are currently investigating if deviating from this semantics allows policies and preferences to be expressed in a more compact way. Moreover, we also study how the matching algorithms have to be adapted when employing more advanced access control mechanisms than the one described in this paper.

Finally, we are developing tools to facilitate the enforcement of policies. Those tools only help honest services to fulfill their commitments. Proving correct enforcement remains out of our scope.

The main ideas of our language have been integrated into the PrimeLife Policy Language (PPL). A prototype engine implementation is currently being developed by the PrimeLife project.

Acknowledgment

The authors would like to thank Ulrich Pinsdorf and Mario Verdicchio for their constructive feedback. 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.

References

[1] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security*, 16(4):369–397, 2008.

[2] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.2). `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/`, November 2003.

[3] Moritz Y. Becker, Cedric Fournet, and Andrew D. Gordon. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security*, 18(4), 2010.

[4] Moritz Y. Becker, Alexander Malkis, and Laurent Bussard. A framework for privacy preferences and data-handling policies. Technical Report MSR-TR-2009-128, Microsoft Research, September 2009.

[5] P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[6] Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Downstream usage control. IBM Technical Report RZ 3777, IBM Research – Zurich, 2010.

[7] ContentGuard. XrML 2.0 Technical Overview. `http://www.xrml.org/Reference/XrMLTechnicalOverviewV1.pdf`, March 2002.

[8] Lorrie Cranor, Marc Langheinrich, and Massimo Marchiori. A P3P preference exchange language 1.0 (APPEL1.0). `http://www.w3.org/TR/P3P-preferences/`, April 2002.

[9] Manuel Hilty, David A. Basin, and Alexander Pretschner. On obligations. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 98–117, Milan, Italy, September 12–14, 2005. Springer, Berlin, Germany.

[10] Manuel Hilty, Alexander Pretschner, David A. Basin, Christian Schaefer, and Thomas Walter. A policy language for distributed usage control. In Joachim Biskup and Javier López, editors, *ESORICS 2007: 12th European Symposium on Research in Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 531–546, Dresden, Germany, September 24–26, 2007. Springer, Berlin, Germany.

[11] Qun Ni, Shouhuai Xu, Elisa Bertino, Ravi Sandhu, and Weili Han. An access control language for a general provenance model. In *Proc. of the 6th VLDB Workshop on Secure Data Management (SDM)*, pages 68–88, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] 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.

[13] ODRL. Open Digital Rights Language (ODRL), version 1.1. `http://www.odrl.net/1.1/ODRL-11.pdf`, March 2002.

[14] Jaehong Park and Ravi Sandhu. The UCONABC usage control model. *ACM Transactions on Information and System Security*, 7(1):128–174, February 2004.

[15] A. Pretschner, F. Schütz, C. Schaefer, and T. Walter. Policy evolution in distributed usage control. In *4th Intl. Workshop on Security and Trust Management*. Elsevier, June 2008.

[16] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[17] W3C. The platform for privacy preferences 1.1 (P3P1.1) specification. `http://www.w3.org/TR/P3P11/`, November 2006.

[18] Xin Wang. Mpeg-21 rights expression language: Enabling interoperable digital rights management. *IEEE MultiMedia*, 11(4):84–87, 2004.

Publication

Enabling Privacy-Preserving Credential-Based Access Control with XACML & SAML

Publication Data

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.

Contributions

- Principal author.
- Concepts, credential-based XACML, SAML as claims language, architectoral extensions.

Copyright

Enabling Privacy-Preserving Credential-Based Access Control with XACML and SAML

C. A. Ardagna[1], S. De Capitani di Vimercati[1], G. Neven[2], S. Paraboschi[3], F.-S. Preiss[2], P. Samarati[1], and M. Verdicchio[3]

[1] Università degli Studi di Milano, Italy
[2] IBM Research – Zurich, Switzerland
[3] Università degli Studi di Bergamo, Italy

Abstract. In this paper we describe extensions to the access control industry standards XACML and SAML to enable privacy-preserving and credential-based access control. Rather than assuming that an enforcement point knows all the requester's attributes, our extensions allow the requester to learn which attributes have to be revealed and which conditions must be satisfied, thereby enabling to leverage the advantages of privacy-preserving technologies such as anonymous credentials. Moreover, our extensions follow a credential-based approach, i.e., attributes are regarded as being bundled together in credentials, and the policy can refer to attributes within specific credentials. In addition to defining language extensions, we also show how the XACML architecture and model of evaluating policies can be adapted to the credential-based setting, and we discuss the problems that such extensions entail.

Keywords: Access control, privacy, anonymous credentials, XACML.

1 Introduction

An appropriate access protection of valuable online services and resources is fundamental to support the ongoing digitalization of businesses, institutions, and governments. In particular, enterprises are moving away from an assumption of an *a priori* knowledge of all authorized users. Rather, they are increasingly opening up their services to — possibly new and unknown — users in possession of credentials issued by trusted third-party identity providers.

In such a scenario, classic access control approaches are not adequate, and alternative approaches such as attribute-based access control (ABAC) [3] are more and more deployed, allowing for the expression of access control restrictions in terms of conditions over the attributes of the requester and of the protected resource. Moreover, standards such as the eXtensible Access Control Markup Language (XACML) and the Security Assertion Markup Language (SAML) were proposed to specify ABAC policies and to exchange authenticated attribute values, respectively.

A number of recent works [11, 2, 1, 6] have proposed a shift from ABAC to *credential-based access control* (CBAC, also called "card-based access control"), in which the attributes, which are collectively attached by a requester in ABAC, are instead grouped in credentials (or "cards") owned by such requester. The issuer of a credential vouches for the correctness of the attribute values with respect to the credential owner.

Not only does this abstract view on credentials intuitively mirror the real-world authentication cards found in every citizen's wallet today, but it also acts as an excellent model to unify authentication technologies as diverse as SAML, OpenID, X.509 certificates, trusted LDAP servers, and anonymous credentials [8, 4, 5]. In particular, we see anonymous credentials as particularly interesting, because of the strong privacy advantages they offer. More specifically, they enable the requester to selectively reveal subsets of attributes from a credential, and even to merely prove that the attributes contained in a credential satisfy a certain condition without revealing the exact attribute values, and all of this while preserving unlinkability between different uses of the same credential.

The goal of this work is to bring privacy-preserving credential-based access control to the real world by leveraging the status of XACML as *de facto* standard in access control languages. To do so, however, a number of issues need to be addressed. First, XACML does not manage attributes bundled in credentials, and, thus, does not allow to distinguish whether two attributes are contained in the same credential or in different ones. This feature is needed to avoid abuses like, for instance, the possibility for the owner of two university diploma credentials to use the course of study of one diploma and the grades of the other. Also, the type of the containing credential may be important, e.g., to distinguish the name as it appears on a passport and on a driver's license, even if both are issued by the state. To achieve such a result, it is fundamental to have a description of credentials and their attributes and that such description is shared among all the actors involved in the system, i.e., there is a need for an ontology of credential types.

Second, XACML prescribes that the requester communicates all of her attributes to the server for the evaluation of the access control policy, which is problematic from a privacy perspective. Some technologies such as SAML, OpenID, and anonymous credentials, offer the possibility to reveal only a subset of the attributes contained in a credential. Such features can be exploited by first communicating the policy to the requester, so that she can disclose only the information necessary for the access.

Third, XACML and SAML merely allow requesters to reveal concrete attribute values, rather than allowing them to prove that certain conditions over the attributes hold. This further privacy-preserving feature can be obtained by leveraging the cryptographic power of anonymous credentials.

This work tackles above issues and is organized as follows: Sections 2 and 3 provide an overview of the literature and the scenario we refer to, respectively;

Section 4 presents the basics of XACML, which are then extended in Section 5 to deal with credentials; Section 6 illustrates how we leverage existing Semantic Web standards to organize credentials and their attributes into hierarchies; Section 7 provides the details of the extension of SAML to enable users to prove conditions over their attributes, and Section 8 shows the necessary modifications of the current XACML architecture to support the proposed extensions; finally, Section 9 summarizes our efforts and discusses our future work.

2 Related Work

The problem of performing access control without prior knowledge of the access-requesters is typically overcome by basing the access control decisions on, possibly certified, properties the requester has. Most of the proposed solutions, e.g., [3, 10, 14], are based on different forms of logic. Although such approaches are highly expressive and powerful, they are difficult to apply in practice where simplicity and easy of use are required. In addition, all such proposals lack support for anonymous credentials.

The eXtensible Access Control Markup Language (XACML) [9] is the de facto standard for expressing access control policies and permits to express access control rules on the basis of a requester's properties. Although XACML enjoys large adoption in industry due to its simplicity and its powerful extension mechanism, it has several limitations. In particular, there is no support for certified credentials nor does it allow for dealing with unknown requesters. Rather, it is assumed that requesters provide all their properties together with the access request. However, this poses significant privacy risks for the requesters.

Some recent proposals have investigated credential-based access control with anonymous credentials. Ardagna et al. [2] introduce a language that is explicitly targeted to anonymous credentials. However, unlike the solution presented in this paper, their proposal does not extend to other technologies. Camenisch et al. [6] propose a language for technology-independent credential-based access control, but the language follows a proprietary syntax, which makes deployment in real-world access control scenarios hard. The recent work by Ardagna et al. [1] defines credential-based access control extensions for XACML, but cannot express the advanced functionalities of anonymous credentials. Neither of the above languages specifies wire formats necessary for the server to communicate the applicable access control requirements to the requester, and for the requester to send a description of her claimed credential properties back to the server. In this work we solve this issue by extending SAML with the necessary expressivity.

3 Scenario

The setting that we envisage is the following. The requester owns a set of credentials obtained from various issuers, possibly implemented in different credential technologies. A credential is a list of attribute-value pairs with technology-specific information, called *pre-evidence*, that the requester will need to substantiate the claims that she will make about the credential. A credential is always of a certain *type* that defines which attributes are contained in the credential. We assume that every credential has the dedicated attributes *type* and *issuer*.

Servers host resources and protect them with policies expressed in an extended version of XACML. Users requesting access to a resource receive the relevant policy, which describes the requirements on the requester's credentials in order to be granted access. The policy may include requirements on multiple credentials at the same time, meaning that multiple credentials have to be presented in order to obtain access, and may include *provisional actions*, i.e., actions that the requester needs to fulfill prior to being granted access.

Subsequently, the requester inspects the policy and, if she has the necessary credentials to satisfy it, she creates a *claim* over a suitable subset of her credentials, which can describe (1) values of attributes contained in these credentials, (2) conditions over non-disclosed attributes, and (3) the fulfilled provisional actions. From the pre-evidence contained in the credentials, the requester derives (technology-specific) *evidence* for the claim to convince the server of its correctness, of the integrity of the attribute values, of her ownership of the credentials, and, possibly, of the freshness of the claim.

Afterwards, the requester makes a new request for the resource, but this time she includes the created claim and evidence. The server verifies the validity of the evidence w.r.t. the claim and evaluates whether the policy is fulfilled by the claim. Acces is granted or denied accordingly.

Note that there is a clear distinction between a credential and a claim: while the former is a typically long-lived bundle of attributes, the latter is a short-lived description of properties that these attributes enjoy. The requester keeps her credentials and the associated pre-evidence until they expire or are revoked; a claim and the corresponding evidence are usually only relevant within a single session with a server.

4 Basics of XACML

XACML defines an XML-based access control policy language as well as a processing model for evaluating the policies on the basis of a given XACML access request. Such request specifies by means of *attributes* which *subject* (i.e., who) wants to perform which *action* (i.e., do what) on which *resource* (i.e., on what).

An XACML policy has a *PolicySet* root element that further contains *Policy* or *PolicySet* elements. A *Policy* contains a set of *Rule*s that define positive or negative authorizations (*Permit* or *Deny* rules). The *Rule*, *Policy* and *PolicySet* elements may contain a *Target* that determines their applicability, i.e., to which access requests the respective elements and their children apply. The *Target* is expressed

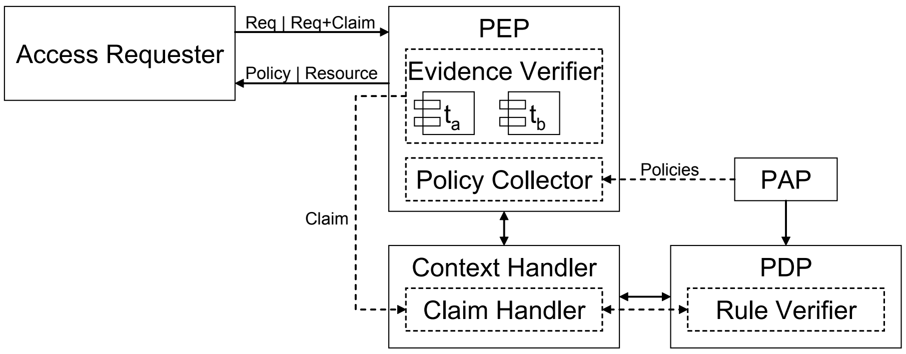


Figure 1: XACML architecture with extensions. Standard XACML components are depicted with *solid* lines. Extensions are depicted with *dotted* lines.

in terms of simple combinations of attributes describing applicable subjects, actions and resources. The applicability of a *Rule* is further determined by a boolean *Condition* that allows for more complex restrictions by means of functions over such attributes. Functions are stated by means of *Apply* elements that specify a respective *FunctionID* XML-attribute (e.g., 'string-equal') and that contain child elements representing the appropriate function parameters. Such parameters may be: (1) *AttributeDesignator* elements referring to attributes given in the request, (2) concrete attribute values, or (3) further *Apply* elements. Each *PolicySet* and *Policy* element also specifies a *combining algorithm* defining how to combine the different outcomes of the contained child elements (i.e., *Policy*/*PolicySet* and *Rule* elements, respectively) when a request is evaluated w.r.t. an XACML policy.

An XACML system consists at least of a policy enforcement point (PEP), a policy decision point (PDP), a policy administration point (PAP) and a context handler (cf. Figure 1). Access requesters issue their requests to the PEP who is responsible for enforcing the access control decisions that are rendered by the PDP on the basis of the request. The PDP makes decisions w.r.t. policies that are created and maintained by the PAP. The context handler is an intermediate component between the PEP and the PDP that buffers the attributes that were given to the PEP in the request and provides them to the PDP on demand.

5 Credential-based XACML

In the following we use the namespace prefix `xacml` to refer to the XACML 3.0 [9] namespace. Our extensions are defined in namespace `http://www.primelife.eu`, denoted by prefix `pl` or without prefix.

The language extensions that we propose to XACML go beyond the standard extension points. All proposed extensions are in line with the semantics of existing XACML language constructs though, i.e., we do not alter the semantics of existing elements or attributes. To facilitate the adoption of our approach in existing XACML code bases, we designed our extensions for minimal impact on the language and the XACML evaluation mechanism.

XACML rules that contain credential requirements can only have effect `Permit`. Rules with effect `Deny` are pointless as they essentially require that the requester *does not* have a certain credential. Assuming that the requester's goal is to obtain access, she can always pretend not to have the specified credentials.

Our extensions enable policy authors to express conditions on the credentials that a requester must own and the actions that she must perform to be granted access. To this end we augment the `<xacml:Rule>` element with optional `<CredentialRequirements>` and `<ProvisionalActions>` child elements. The former describes the credentials that the requester needs to own and the conditions these credentials have to satisfy. The latter describes the actions that she has to perform. We now discuss both elements in more detail.

5.1 Credential Requirements

To express credential-based access control policies, the language needs a way to refer to the credentials that bundle several attributes together. For example, it must be possible to refer to the requester's name as it appears on her passport, not on her credit card. Cross-credential conditions are another important use case: for example, the policy language must allow to express that the names on a credit card and on a passport must match, or that the expiration date of an entry visa is before the expiration date of a passport.

To this end, `<CredentialRequirements>` contains a `<Credential>` child element for each credential involved in the rule, which is assigned a (rule-wide unique) identifier `CredId` as an attribute. The `<Credential>` can contain `<AttributeMatchAnyOf>` child elements that allow to compare an attribute of that credential to a list of values. The `<CredentialRequirements>` also contain a `<Condition>` where conditions on the credentials' attributes can be expressed. Inside a condition, one can refer to an attribute `AttrId` within a particular credential by means of a `<CredAttributeDesignator>` element which takes both `CredId` and `AttrId` as attributes. We purposely did not add an optional `CredId` attribute to `<xacml:AttributeDesignator>`, as the `Issuer` attribute of the latter would conflict with the credential attribute `pl:Issuer`. An example rule is given in Figure 2.

Conditions on credential attributes are expressed using the same schema as the `<xacml:Condition>` element (including the mentioned extension with the `<CredAttributeDesignator>` element), but are contained in a separate `<Condition>` child element of the `<CredentialRequirements>` element. The reason for this separation is that whenever no adequate claim is attached to a resource request, the applicable policy must be returned. Deciding applicability of a policy entails evaluating the `<xacml:Condition>`, which is impossible when it involves credential attributes that have not been revealed yet. To avoid this issue,

```
<Rule Effect="Permit" RuleId="rule2">
 <xacml:Condition>
       <!-- XACML condition relevant for the rule's applicability -->
 </xacml:Condition>
 <CredentialRequirements>
  <Credential CredentialId='pp'>
   <AttributeMatchAnyOf AttributeId="pl:CredentialType">
    <MatchValue MatchId="pl:subtype-of">un:PhotoID</MatchValue>
   </AttributeMatchAnyOf>
   <AttributeMatchAnyOf AttributeId="pl:Issuer">
    <MatchValue MatchId="xacml:anyURI-equal">http://www.usa.gov</MatchValue>
   </AttributeMatchAnyOf>
  </Credential>
  <Condition>
   <xacml:Apply FunctionId='xacml:date-less-than-or-equal'>
    <CredentialAttributeDesignator CredId="pp" AttributeId="un:DateOfBirth"/>
    <xacml:Apply FunctionId="xacml:date-subtract-yearMonthDuration">
     <xacml:EnvironmentAttributeDesignator AttributeId="xacml:current-date"/>
     <xacml:AttributeValue DataType="xs:duration">P21Y</xacml:AttributeValue>
    </xacml:Apply>
   </xacml:Apply>
  </Condition>
 </CredentialRequirements>
 <ProvisionalActions>
  <ProvisionalAction ActionId="pl:Reveal">
   <xacml:AttributeValue DataType="xs:anyURI">un:Sex</xacml:AttributeValue>
   <xacml:AttributeValue DataType="xs:anyURI">pp</xacml:AttributeValue>
  </ProvisionalAction>
 </ProvisionalActions>
</Rule>
```

Figure 2: Example rule stating that access is granted to users who are at least twenty-one years old according to a piece of PhotoID issued by the US government, but only after revealing the gender mentioned on the same piece of PhotoID. Namespace prefix `xacml` refers to the XACML 3.0 namespace, `xs` to XML Schema, `pl` to `http://www.primelife.eu`, and `un` to `http://www.un.org`.

we keep the credential requirements separate from elements relevant to the rule's applicability.

Conditions can contain any combination of restrictions on any credential attributes, including the issuer and the type of the credential. For matching credential types, we introduce a new function `pl:subtype-of` that checks whether the presented credential is of a subtype of a specified credential type as per the ontology that we will discuss in Section 6.

5.2 Provisional Actions

The `<ProvisionalActions>` element contains the actions that have to be performed by a requester prior to being granted access. The types of actions that we model are:

– *Consent:* The requester has to explicitly consent to a given statement, e.g., the terms of service. How consent is given could depend on the underlying technology: it could for example involve a cryptographic signature, or simply a click on a button in the user interface.

– *Attribute disclosure:* Rather than assuming that all attributes of a credential are revealed by default, the policy explicitly lists which attributes of which credential need to be revealed. Moreover, in order to fully leverage the power of privacy-enhancing technologies such as anonymous credentials that allow the requester to prove conditions without revealing the attribute values, this list does *not* (necessarily) include the attributes that occur in the conditions. Apart from the attribute and credential identifiers, the requirement to reveal an attribute can optionally specify a data handling policy describing how the attribute value will be treated.

In some cases, it is not the PEP who needs the attribute value, but some third party. For instance, an online bookshop may require the requester to reveal her address to a shipping company, not to the bookstore itself. When using anonymous credentials, such requirements can be efficiently fulfilled by means of verifiable encryption [7]. We model this with an optional fourth argument describing the entity to whom the attribute must be disclosed.

– *Consumption control:* Consumption control allows the policy author to impose limitations on how often the same credential can be used to obtain access. For example, one could impose that each ID card can only be used once to vote in an online opinion poll. A consumption control statement has to specify the credential to be consumed, the number of units to spend, the limit of units that can be spent in total, and a "consumption scope". We refer to [6] for details on their exact semantics.

Rather than restricting our extensions to the above action types, we leave open the possibility to add new types of provisional actions later. We do so by a mechanism similar to the one used for defining functions in XACML. Namely, each provisional action is contained in a `<ProvisionalAction>` element that includes an action identifier as an attribute `ActionId`. We define action identifiers for the action types above, but allow users to add more identifiers later.

Provisional actions can be parameterized with arguments, e.g., the statement to be consented to in a `Consent` action. We use an approach similar to how in XACML arguments are passed to functions in the `<xacml:Apply>` element. Namely, a `<ProvisionalAction>` can take any number of `<xacml:Expression>` child elements, in which the action arguments are encoded. For example, the policy in Figure 2 contains the requirement to reveal the gender as specified on the identity card.

This is, however, slightly problematic for actions with optional arguments: since arguments are passed as an unnamed sequence of `<xacml:Expression>` elements, parsing them becomes ambiguous if more than one argument is optional.

For example, the requirement to reveal an attribute takes one mandatory argument (the attribute to be revealed) and three optional arguments (the recipient, the data handling policy, and the credential identifier). We solve this problem by introducing separate action identifiers for each possible combination of specified attributes. For example, for attribute disclosure we define the action identifiers `pl:Reveal`, `pl:RevealTo`, `pl:RevealUnderDHP`, and `pl:RevealToUnderDHP`.

6 Credential Type Ontologies

For an effective extension of XACML that bundles attributes into credentials, such organization of information, which we call a *credential type ontology*, must be shared by all involved parties. The best way to achieve this result is to express the structures of the credentials in a widespread standard language. Moreover, this language should allow for the extension of existing types, so that credential issuers can introduce new types within the context of the ontologies already in use. We rely on the Web Ontology Language (OWL) [13], which is XML-based, thus allowing for ontologies that are easily extended and exchanged by applications, regardless of the platform.

In our model, each credential belongs to a specific type, which defines the list of its attributes. The type of a credential is itself an attribute. Types and attributes are defined in ontologies that can be referred to in the policy language by means of URIs, possibly mapped onto prefixes (e.g. `http://www.un.org/` is mapped onto `un:`). For example, the United Nations could design an ontology that defines the attribute `un:nationality` as specifying the nationality of a credential bearer, encoded as a two-character ISO 3166 country code, and the credential type `un:Passport` as a digital template for real-world passports, comprised of a list of attributes including `un:nationality`. Attributes defined in an ontology can be reused in other ontologies: a government may define a credential type for national ID cards, including the `un:nationality` attribute. Credential types are organized in a hierarchy with supertypes and subtypes. If credential type A has a subtype B, then B contains all the attributes of A with possible restrictions on their values, and it may include additional attributes. Multiple inheritance is allowed, meaning that a subtype contains all the attributes of all its supertypes.

Our use of OWL for credential ontologies is based on the following idea: a credential type is modeled as an OWL class. In particular, the URI of the OWL class can be used as the URI of the credential type, and we are enabled to exploit the OWL class definition mechanism to define which attributes a certain credential type includes. Moreover, we can make use of OWL's subclassing mechanism (`rdfs:subClassOf`) to model inheritance among credential types in a very natural way, with OWL subclasses corresponding to credential subtypes.

We define a root credential type with URI `pl:Credential` (cf. Figure 3), which is the supertype of all other credential types and contains the credential's issuer (`pl:issuer`) as an attribute.

```
<owl:Class rdf:about="Credential">
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="issuer"/>
   <owl:qualifiedCardinality
       rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:qualifiedCardinality>
   <owl:onDataRange rdf:resource="&xsd;anyURI"/>
  </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

Figure 3: An OWL class for credentials

All other credentials can be defined as a subclass of the `pl:Credential` class. For instance, the United Nations' `un:Passport` credential type can be defined as a subclass of `pl:Credential` by extending its attribute list with attributes `un:firstName`, `un:lastName`, `un:dateOfBirth`, `un:nationality`, and `un:photo`.

In our model, when a policy requires the presentation of a credential of a certain type, a credential of any subtype is accepted, as it contains all the information included in the supertype, and some more. Function `pl:subtype-of` is introduced to be applied to two URIs to check whether the first argument refers to a subclass of the class referred to by the second argument. Let us suppose a credential of type `un:Passport` is presented by an access requester for the evaluation of a policy that requires an instance of `pl:Credential`. By exploiting the capabilities of ontology reasoners, function `pl:subtype-of` states that `un:Passport` is indeed a subclass of `pl:Credential`, so that the requester can be granted access.

7 SAML as Claims Language

Here we describe how we extend SAML for transporting the claims defined in Section 3. SAML is a standard allowing for the exchange of certified attributes bundled together into *assertions*, which are similarly structured as credentials. The standard, however, allows only for the exchange of attribute values but not conditions on such values nor notifications of provisional action fulfillment. To address these issues, we use the standard's extension points to embed our `<Condition>` and `<ProvisionalAction>` elements into SAML assertions. This allows for the expression of conditions on attributes from one or more credentials as well as action fulfillments, together with the necessary evidence.

8 Architectural Extensions

In the following we sketch how we adapt the XACML architecture such that (1) the credential-based XACML policy applicable to a request is communicated to the requester, and (2) the policy can be evaluated on the basis of the provided SAML claim. The modified architecture maintains all standard XACML functionality, i.e., the modifications are *extensions* that do not substitute existing functionality and that are usable in combination with standard features.

We adapt the XACML communication model for allowing the following two-round pattern. In the first round, the requester specifies a resource and obtains the relevant policy from the PEP; in the second round the requester sends the same request with an additional SAML claim. Resending the request is necessary because the XACML architecture is stateless, meaning that the individual components do not maintain information across multiple rounds. A PEP's response in the first round is embedded in an *XACMLPolicy Assertion* element (cf. SAML profile of XACML [12]), to which the requester is supposed to reply with an appropriate SAML claim. The PEP grants or denies access depending on the claim's validity and the decision of the PDP.

We need to modify the PEP such that it obtains all policies applicable to a user's request and it sends them in a pre-evaluated version to the user. The pre-evaluation substitutes known attributes, e.g., environment attributes such as time and date, with concrete values. One way to obtain all applicable policies is to exploit the *ReturnPolicyIdList* feature of XACML 3.0, which enables a PEP to learn from a PDP all the relevant policy identifiers. The PEP can then obtain the policies directly from the policy administration point (PAP). Alternatively, the PDP could be modified to return not only a list of `PolicyIds` to the PEP, but rather the policies themselves.

When the PEP receives a request with an attached SAML claim, it has to verify the validity of the claim and make it available to the PDP. To verify the validity of the claim evidence, we extend the PEP with an *evidence verifier* component (cf. Figure 1). For every supported credential technology t, this component has a plug-in that can verify evidence specific to this technology. To make the claim available to the PDP, we introduce a *claim handler* component within XACML's context handler. If the claim is valid, the PEP forwards it to the claim handler, which buffers it so that it can be retrieved by the PDP. The PEP then forwards the request (without attached claim) to the PDP.

A PDP evaluates a request from the PEP as usual w.r.t. the rules in the policy. However, rules with credential requirements or provisional actions are treated specially. For such rules to yield a `Permit` decision, not only its applicability (specified by its *target* and *condition*) is relevant, but also the fulfillment of the credential-requirements and provisional actions if any are specified. If so, the PDP fetches the claim from the claim handler. We extend the PDP with a *rule verifier* component that, for given credential requirements, given provisional actions, and a given claim, decides whether the claim implies the requirements and fulfills all the provisional actions. If so, then the rule evaluates to `Permit`, otherwise it evaluates to `Indeterminate`.

9 Conclusion

In this paper, we proposed extensions to the industry standards XACML and SAML to add support for credential-based access control that fully leverage the power of privacy-preserving technologies such as anonymous credentials. A prototype engine for the language and architecture extensions that we defined is currently being developed under the auspices of the European Commission's PrimeLife project.

Our current architecture requires a new access control process with new claims to take place for each resource request with associated credential requirements. In future work, we will investigate to which extent this can be avoided by comparing the relevant policy to previously made claims.

Acknowledgements

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.

References

[1] C.A. Ardagna, S. De Capitani di Vimercati, S. Paraboschi, E. Pedrini, P. Samarati, and M. Verdicchio. Expressive and deployable access control in open web service applications. *IEEE Transactions on Services Computing*, 4(2):96–109, April-June 2011.

[2] Claudio A. Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. Exploiting cryptography for privacy-enhanced access control. *Journal of Computer Security*, 18(1):123–160, 2010.

[3] P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[4] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[5] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.

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

[7] 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 17–21, 2003. Springer, Berlin, Germany.

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

[9] eXtensible Access Control Markup Language (XACML) Version 3.0. Committee Draft 01, Organization for the Advancement of Structured Information Standards (OASIS), April 2009.

[10] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26:214–260, June 2001.

[11] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 46–57, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press.

[12] SAML 2.0 Profile of XACML v2.0. OASIS Standard, Organization for the Advancement of Structured Information Standards (OASIS), February 2005. OASIS Standard.

[13] W3C. OWL 2 web ontology language. `http://www.w3.org/TR/owl2-overview/`, October 2007.

[14] Marianne Winslett, Neil Ching, Vicki Jones, and Igor Slepchin. Assuring security and privacy for digital library transactions on the web: client and server security policies. In *Proc. of the IEEE International Forum on Research and Technology Advances in Digital Libraries (ADL)*, pages 140–151, Washington, DC, USA, 1997. IEEE Computer Society.

Publication

A Card Requirements Language Enabling Privacy-Preserving Access Control

Publication Data

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

Contributions

- Principal author.
- Card and system model, requirements language and grammar.

Copyright

A Card Requirements Language Enabling Privacy-Preserving Access Control

J. Camenisch, S. Mödersheim, G. Neven, F.-S. Preiss, and D. Sommer

IBM Research – Zurich, Switzerland

Abstract. We address the problem of privacy-preserving access control in distributed systems. Users commonly reveal more personal data than strictly necessary to be granted access to online resources, even though existing technologies, such as anonymous credential systems, offer functionalities that would allow for privacy-friendly authorization. An important reason for this lack of technology adoption is, as we believe, the absence of a suitable authorization language offering adequate expressiveness to address the privacy-friendly functionalities. To overcome this problem, we propose an authorization language that allows for expressing access control requirements in a privacy-preserving way. Our language is independent from concrete technology, thus it allows for specifying requirements regardless of implementation details while it is also applicable for technologies designed without privacy considerations. We see our proposal as an important step towards making access control systems privacy-preserving.

Keywords: Access control, policy languages, privacy, anonymous credentials, digital credentials.

1 Introduction

Current industry trends such as software as a service and cloud computing drive businesses to open up their software infrastructures to a wider online audience. Enterprises that used to populate their internal user directories by doing their own identity vetting are now moving away from their closed-world assumption and start relying more and more on external identity providers instead. On the other hand, as users' personal data move outside the enterprise boundaries, privacy protection becomes an even bigger concern than before. Clearly, companies have to strike a trade-off between the protection of their resources and the privacy of their users. A crucial tool in implementing this trade-off is an adequate language to express access control requirements in a way that does not force users to reveal more personal information than strictly necessary.

A plethora of technologies exist to authenticate users and bind attributes to them, including X.509 certificates, SAML, OpenID, trusted LDAP directories, and anonymous credentials [20, 12, 16]. The latter offer a number of appealing privacy features, such as proving predicates over attributes, disclosing attributes to third parties, accountable anonymity, and privacy-friendly consumption control. In this paper, we abstract all of these technologies into a generic model of digital "cards" that captures the advanced features they offer, but makes abstraction of the technical details. The idea of using an abstraction is not new. In particular, the Identity Metasystem [21] advocates an abstraction called "identities" which is similar to our card model (cf. Section 2 for details).

We model a *card* as an authenticated list of attribute-value pairs issued by an issuer to a card owner. To obtain access to a server's resource, the card owner creates a *claim*, i.e., a statement about some of the attributes of one or more of her cards. She also generates (technology-dependent) *evidence* for the claim that is basis for the server (also called the *relying party*) to verify the authenticity of the claim, the freshness of the evidence, and fulfillment of the policy protecting the resource.

In this paper, we propose a card-based access control requirements language (CARL) that allows the server to express the requirements that a user's cards have to satisfy in order to gain access to a resource, without having to worry about the specifics of the underlying technology. Moreover, our language is privacy-preserving in the sense that it expresses the *minimal* claim that a user has to present. It does so in terms of which cards have to be involved in the claim, which attributes of those cards have to be revealed, and which conditions have to hold over the attributes (whether these were revealed or not). This approach allows the user to minimize the amount of data that she reveals to the server, which is important as cards often contain sensitive personal information. Moreover, some of the supported technologies (including SAML, OpenID, and in particular anonymous credentials) allow to derive claims that do not necessarily reveal all of the attributes of a card, but only a subset of them, or even just the fact that their values satisfy a certain boolean condition. Our language leverages these technologies to their full potential, without making compromises on compatibility with older technologies though, since the user can always choose to reveal more information than required.

We summarize the core features of our language below. While some of the individual features may also be supported in previous work, we see as important contribution that the CARL supports all these features *simultaneously*. See the next section for a detailed comparison with related work.

Privacy preservation. The CARL is privacy-preserving in the sense that it supports the principle of minimal information disclosure. Rather than assuming that all attributes in a card are revealed by default, it explicitly specifies which attributes must be revealed, and clearly distinguishes between the requirement to reveal the value of an attribute (e.g., the date of birth) and the requirement that an attribute has to satisfy a certain condition (e.g., age greater than 18). Our language also supports *accountability*, so that anonymity can be revoked by a third party in case of abuse.

Technology independence. Our language is independent of the technology underlying the cards, so that different technologies or even a mix of technologies can be used without modifying the policy specifications. Also, its concepts are detailed enough to leverage the advanced features of available technologies (in particular those of anonymous credentials) such as consumption control and attribute disclosure to third parties.

Multi-card claims. Our policy language can express requirements involving multiple cards at the same time and has a way to refer to individual cards and the attributes they contain. It can thereby impose "cross-card" conditions, e.g., a car rental service can request to see an identity card, a driver's license, and a credit card, which are all registered to the same name. The *card type* determines the attributes a card contains, and the policy can specify a card to be of a certain type. For example, the car rental service may want to see the address as stated on the credit card, not on the driver's license. Being able to reference individual cards is also important when a user has multiple cards of the same type. For example, when a user has two credit cards, the policy should be unambiguous about whether it wants to see the credit card number and security code of the *same* card or of *different* cards. (We refer to this issue as the *card mixing problem.*)

Formal semantics. We provide a formal semantics for our language that mathematically defines an ideal system that determines what a particular realization must implement. This ideal system defines the proof goal for any realization with a concrete technology. We do not define a particular kind of realization in terms of a concrete set of actions (such as an exchange of particular messages), because this would be technology dependent.[1] Defining a formal semantics has helped us to surface subtle problems that might otherwise go unnoticed, such as the card mixing problem mentioned above.

In this paper, we focus only on the language that is used to describe card requirements. When designing a comprehensive access control language, various additional aspects have to be taken into account, such as how to specify the resources and requested actions to which the policy applies, how to express trust relationships and trust delegation, how to state data handling policies, how to combine multiple applicable policies, etc. In Appendix D, we sketch how our language can be integrated into XACML [34] to profit from the mechanisms for defining the applicable resources and actions, and how it can be integrated into existing authorization languages like SecPal [7] and DKAL [28] to express delegation and other complex trust structures.

[1]An example of such a realization was done by Mödersheim and Sommer [17].

2 Related Work

Card-based access control can be seen as a generalization of a variety of access control models. Role-based access control [25, 35] can be seen as a special case of our card-based setting by encoding a user's roles as attributes in a card. Attribute-based access control [10, 34, 37] comes closer to our concept of card-based access control, but it does not see attributes as grouped together in cards.

The idea of technology independence for authentication mechanisms is not new. In particular, the Identity Metasystem [21] advocates a very similar abstraction, but focuses solely on use cases where only a single card (called "identity" in their framework) is used at each authentication session. The associated WS-* suite of policy languages and the CardSpace implementation support selective attribute disclosure and even a limited set of predicates over attributes, but again only for a single card per authentication. We think that multi-card authentication is an important use case, even though it necessarily adds to the complexity of the policy language; hence the need for our language. Advanced features such as disclosure to third parties, signing statements, and consumption control are also not supported in the Identity Metasystem.

Bonatti and Samarati [10] propose a language for specifying access control rules based on "credentials", the equivalent of our cards. The language focuses on card ownership and does not allow for more advanced requirements such as revealing of attributes or signing statements. The same is true for the language proposed in [2].

Winsborough et al. [38] present a credential-based access control language that allows one to impose attribute properties on credentials, but does not support advanced features such as revealing of attributes, signing statements, or consumption control. The extension by Li et al. [30] supports revealing of attributes, but not the other features.

The PolicyMaker [8] and KeyNote [9] trust management system uses 'credentials' to bind 'assertions' to keys. In PolicyMaker, assertions can be described in any safe language, while KeyNote fixes a language. Both could be used to encode attribute-value pairs in the assertion to implement our card concept, but selective attribute disclosure would not be possible, unless new credentials are re-issued at each login.

P3P [36] defines a format for websites to express their privacy practices. EPAL [4] is a framework that allows enterprises to manage personal data they have collected on the basis of privacy statements. However, no support for credentials is offered.

The languages mentioned above are not targeted to anonymous transactions and thus lack language constructs that allow for obtaining accountability in anonymous transactions, which we do achieve through a combination of signing statements and disclosure to third parties. The first paper towards third-party disclosure is due to Backes et al. [6]. Gevers and De Decker [27] extend P3P to describe credentials and necessary access control requirements, including verifiable encryption (which we generalized to disclosure to third parties), but no precise syntax or semantics are specified.

The recent language by Ardagna et al. [3] features a card typing mechanism and advanced features such as consumption control and signing statements. It focuses only on anonymous credential systems though, hence cannot be used in combination with other technologies. It also lacks a formal semantics and suffers from the card mixing problem.

Several logic-based approaches for authentication and distributed access control have been devised to achieve similar goals as this work: a technology-neutral, declarative specification of distributed access control [1, 11, 26, 31]. The fundamental idea is to describe access control requirements by formulae in authentication logic and to provide a calculus for proving such formulae. To gain access, the requester constructs a valid proof for the formula based on the credentials she owns and sends it, together with the relevant credentials, to the server. However, none of these approaches is fully privacy-preserving: they do not support selective disclosure of attributes (as opposed to transmitting entire credentials/cards), proving predicates over attributes (as opposed to disclosing these attributes), disclosing attributes to third parties, and signing statements with respect to the proved attribute properties.

In contrast, CARL specifies the minimal amount of information each involved party has to learn for access to be granted. The formal semantics is specified not as a calculus but as conditions on the cards/credentials that the requester holds and the precise amount of information the involved parties gain. It is then the duty of an implementation to show that users can only prove statements that hold on their credentials and that involved parties do not learn more information than specified. We briefly discuss how to combine CARL with existing logic-based approaches in Appendix D.1; we leave an in-depth investigation of such combinations for future work.

3 Background

In the following we explain in more detail our abstract model of a "card". We also discuss some of the technologies that can be used to instantiate this model and the functionalities that they offer, because they have strongly influenced the design of our language. Finally, we sketch how our language fits into the bigger picture of a complete privacy-enhanced card-based access control solution.

3.1 Our Card Model

The language that we present is geared towards enabling user-centric and privacy-preserving access control based on certified cards. While it leverages the advanced features offered by anonymous credential systems, it is technology-agnostic in the sense that it makes abstraction of the particular technology that is used to certify the cards.

To better understand our model, we will at each step illustrate the concepts by describing how they are instantiated by X.509 v3 certificates [23]. These allow a certification authority (CA) to bind arbitrary community-specific attributes to a user's public key by creating a signature (under the CA's public key) of the user's attributes and her public key. In the next subsection, we sketch how a number of other technologies can also be seen to implement the same concepts.

A *card* is issued by a card *issuer* to a card *owner*. The issuer vouches for the correctness of the information on the card with respect to the intended owner. The information is meant to affirm qualification, typically in the form of *identity* or *authority*. However, the meaning of the information has no technical relevance and is subject to interpretation of the party relying on it. In an X.509 certificate, the CA acts as card issuer. While we are mainly interested in cards that can be presented to third parties, some cards may be intended for internal use only and be verifiable only by the issuer; think for example of a company-internal LDAP directory.

A card consists of a list of attribute-value pairs and of technology-specific auxiliary data called the *pre-evidence*. The pre-evidence can contain meta-data (cryptographic or other) that the owner needs when presenting the card to a relying party. In an X.509 certificate, e.g., the pre-evidence contains the CA's public key pk_{CA}, the user's public key pk_{U}, the user's secret key sk_{U}, and the CA's signature σ_{CA} on the list of attributes and pk_{U}. (For simplicity, we assume that the CA is a root authority; otherwise, the pre-evidence also contains the certificate chain of pk_{CA} to a root authority.) The issuing process may be carried out on-line, e.g., by visiting the issuer's website, as well as off-line, e.g., at the local town hall.

Cards are always of a certain *card type* that specifies the list of attributes that the card contains. For example, a national ID card may contain the first name, last name, date of birth, and address of the owner, while a movie ticket contains the time and date of the showing and a seat number. We consider a hierarchical ontology of card types so that types can inherit from other types; see Section 4.1 for details. A card serves as a means for proving qualification, i.e., it typically serves to prove identity, authority, or both. For example, a national ID card can be used to prove identity, a movie ticket to prove authority to attend a particular movie showing from a particular seat, and a driver's license to prove identity as well as the authorization to drive motor vehicles of a certain category.

For gaining access to a resource protected by a policy, the server has to be convinced that the policy is fulfilled. To do so, in our system model the card owner makes a *claim* about the cards she owns and about the attributes they contain. Claims are made independent from concrete technology, and are authenticated by accompanying *evidence*. The evidence is, however, specific to the technology underlying the cards. In the ideal case of privacy-preserving technologies, this claim reflects exactly the policy. For other technologies, the claim is something stronger than required. With X.509, e.g., all cards' attributes are revealed, no matter what the policy requires.

The evidence is derived from the card's pre-evidence, and used by the server to check (1) the integrity of the claim, (2) the freshness of the evidence, and (3) the rightful ownership of the card. Depending on the technology, the user may be able to derive the evidence herself, or she may need to interact with the card issuer. Likewise, the server may be able to independently verify the evidence, or may need the help of the issuer.

For X.509, for example, the evidence consists of pk_{CA}, pk_{U}, σ_{CA}, and a signature σ_{U} created by the owner on the claim statement and a random nonce chosen by the relying party. Here, σ_{CA} protects the integrity of the attributes, while σ_{U} simultaneously acts as a proof of freshness of the claim and a proof of ownership (through the knowledge of sk_{U}) of the card. The evidence is independently created by the owner; the relying party may have to contact the issuer however to check that pk_{U} has not been revoked.

3.2 Example Technologies

By above card abstraction our policy language can specify access control restrictions without having to worry about the underlying technology. This means that a card of any supported technology can be used to satisfy a policy, and even that cards of different technologies can be combined in a single claim.

We already described how X.509 certificates fit our card model. Below, we sketch how anonymous credentials and trusted LDAP directories fit our model, and in Appendix A we point out how OpenID, Kerberos, SAML, and even everyday email accounts can be seen to do so as well. This list is by no means exhaustive; in fact, we envision that most existing and even future technologies will fit our model.

Anonymous Credentials Much like an X.509 certificate, an anonymous credential [20, 12, 16] can be seen as a list of attribute-value pairs signed by the issuer with an underlying secret signing key for the user. Unlike X.509 certificates, however, anonymous credentials have the advantage that the owner can reveal subsets of the attributes, or merely prove that a condition over the attributes holds. Also, they provide additional privacy guarantees like unlinkability, meaning that, even when colluding with the issuer, a server cannot link multiple visits by the same user or link a visit to the issuing of the card.

Two main anonymous credential systems have been implemented today, namely Identity Mixer [16, 19] and UProve [24]. We will focus mainly on Identity Mixer in this paper because of its multitude of associated cryptographic tools such as verifiable encryption and consumption control (also known as "limited spending"), but we stress that UProve credentials can be used as a technology for our policy language as well.

LDAP The Lightweight Directory Access Protocol defines a standard interface to directory servers containing information about users. If the LDAP server is trusted, one can see each user directory entry as a card belonging to that user and issued by the LDAP server. The user authenticates to the LDAP server using a password or using a more advanced authentication mechanism.The evidence of a claim is simply the LDAP server's URL, the relying party can verify the attributes simply by looking them up in the directory. Any subset of attributes can be revealed, even though a cheating server can always look up other attributes as well.

3.3 Functionality

We now describe a number of special features of cards that our policy language supports and sketch how these features are or could be implemented in the different technologies.

Proof of ownership To bind a card to its legitimate owner, the card's pre-evidence may contain information that is used to authenticate the owner. This could be a picture of the user, a PIN code, a password, or a signing key. Proving card ownership in our notion means that the owner authentication is successfully performed with whatever mechanism is in place. Depending on the employed mechanism, successful authentication may also provide a liveness guarantee (to prevent replay attacks).

The actual implementation of the ownership proof is technology dependent. For X.509 certificates, ownership can be proven by signing a random nonce. In anonymous credentials, users construct a zero-knowledge proof of knowledge of an underlying master secret. OpenID, LDAP, Kerberos, and email accounts all work with a password-based approach.

Related to the question of ownership is the *transferability* of a card from one user to another. Some cards may be transferable (e.g., movie tickets) while others may not (e.g., driver's licenses and identification cards in general). We do not further elaborate this concept here, but rather assume that the underlying technology prevents abuse if necessary, for example by letting the underlying authentication secret be the same as that of a highly sensitive card, e.g., the signing key of a national ID card.

Selective attribute disclosure Some technologies allow attributes within a card to be revealed selectively, i.e., the relying party only learns the value of a subset of the attributes contained in the card.

Not all technologies support this feature. Verification of the issuer's signature on an X.509 certificate requires all attribute values to be known. The LDAP protocol specifies the attributes to fetch, but the user has no control over which attributes a cheating server looks up. Anonymous credentials, SAML, and (under certain settings) OpenID, on the other hand, have native support for this feature, although the mechanisms are quite different. For OpenID and SAML the provider simply only reveals those attributes that were explicitly requested, while for anonymous credentials it is the cryptography that ensures that no information is leaked about non-disclosed attributes.

Proving conditions on attributes Anonymous credentials even allow one to prove conditions over attributes without revealing their actual values. (While in theory any condition can be proved, this mechanism is only truly practical for certain classes of conditions. We refer to [3] for details.)

For all other technologies the only way to prove that an attribute satisfies a condition is by revealing its value. Perhaps future versions of technologies with online verification such as OpenID or SAML will enable the identity provider to confirm that conditions over attributes hold, rather than having to reveal their values.

Attribute disclosure to third parties Usually attributes will be revealed to the relying party enforcing the policy, but the policy could also require certain attributes to be revealed to an external third party. For example, the server may require that the user reveals her full name to a trusted escrow agent, so that she can be de-anonymized in case of fraud, thereby adding accountability to otherwise anonymous transactions. As another example, an online shop could require the user to reveal her address to the shipping company directly, rather than disclosing it to the shop.

Of course, the relying party needs some sort of evidence that the user actually did reveal the necessary information to the third party. Identity Mixer elegantly supports this feature using verifiable encryption [14, 5, 18]. Here, the user hands to the relying party a ciphertext containing the relevant attribute(s), encrypted under the third party's public key, and adds a zero-knowledge proof that the correct attribute was encrypted. Moreover, a data handling policy can be bound to the ciphertext, e.g., to describe under what conditions it can be decrypted. The relying party is unable to change the data handling policy when forwarding the ciphertext to the third party.

This feature could be added to other technologies as well by letting either the user or the issuer send the necessary attributes to the third party directly, and letting the third party sign a receipt that can be shown to the relying party.

Signing of statements Our policy language also enables the server to require the user's explicit consent to some statement, e.g., the terms of service or the privacy policy of the site. The signature acts as evidence that this statement was agreed to by a user fulfilling the policy in question.

There are various ways in which users can express consent; our language does not impose a particular implementation. Using X.509 certificates the most straightforward way is to digitally sign the document. Anonymous credentials allow to sign statements while maintaining maximal privacy: anyone can verify that the signature was placed by a user satisfying the access control policy, but only a trusted opening authority can tell who exactly the user was. Using other technologies, the relying party could simply record the fact that the user confirmed the statement by clicking an OK-button.

Note that these signatures, together with attribute disclosure to third parties, are the key to make anonymity revocable and actions accountable. Consider, e.g., a policy requiring a user to reveal her real name n and an order confirmation number o to a trusted third party. In addition, the user has to sign a statement involving o. In case of a dispute, the server forwards the signature to the trusted party who can now resolve the issue with the person named n. The server itself, however, cannot find out this name. With conventional non-privacy-friendly technologies, the signature will also reveal the real name of the user to the server though.

Consumption control The server may want to impose limitations on the number of times that the same card can be used (or "consumed") to access a resource, e.g., to specify that each national ID card can only be used once to vote in an online opinion poll.

Bowers et al. [11] previously proposed a logic-based policy language for consumable cards (or *credentials* in their work). To each card, they associate a single global consumption limit and a *ratifier*, a central entity who keeps track of each usage of the card. This works well for settings such as electronic cash where the issuer of the card determines how often the card is to be consumed *globally*, but falls short of covering use cases like the opinion poll example where the relying party wants to impose a limit how often the same card can be used *locally*.

We therefore extend their consumption functionality in our language such that the relying party itself can specify (1) the *consumption amount*, i.e., the number of units that is to be consumed from the card per access; (2) the *consumption limit*, i.e., the maximum number of units that can be consumed from the card before access is refused; and (3) the *ratification scope* being a URI defining the "scope" of the consumption, i.e., the limit becomes relative to this scope.

Part of the scope URI could be used to identify the central entity who keeps track of the number of consumptions, but it can additionally encode the scope within which consumption is to be tracked. For example, the access restrictions of the opinion poll service would be such that a different URI is used for each poll, specifying that each ID can be used once *for each poll*, rather than that the user can only take part in a single poll. The scope mechanism allows also to express more complex limitations. For example, to prevent two resources A and B from being accessed more than n times total per month, the scope URI in both policies is set to $\mathit{append}(\text{`examplescope:AorB:'}, \mathit{currMonth}(), \text{`/'}, \mathit{currYear}())$.

Some anonymous credential systems support consumption control (usually referred to as *limited spending*) natively in the underlying cryptography [13, 15].

They do so in a highly privacy-preserving way: the user's anonymity and unlinkability remain guaranteed until overspending occurs.

None of the other technologies have native support for consumable cards, but for all of them support could be added by letting the issuer play the role of ratifier, or by adding a unique serial number that is revealed whenever the card is consumed. (The latter approach makes all transactions of the same card linkable, though.)

3.4 System Model

Our system model involves three kinds of entities: users, servers (also called service providers), and issuers. Users hold certified cards $C_1, \ldots, C_k$ that they have obtained from issuers and want to access protected resources $R_1, \ldots, R_m$ (e.g., web pages, databases, web services) hosted by the servers. Servers restrict access to their resources by means of access control policies $A_1, \ldots, A_\ell$. Rather than simply specifying which user is allowed to access which resources by means of a classical access matrix, the policies contain requirements in terms of the cards that a user needs to own in order to be granted access.

Figure 1 depicts how authorization decisions are rendered in our model. Initially, a user contacts a server to request access to a resource she is interested in (1).

Having received the request, the server responds with the access control policy applicable for the resource (2). The applicable policy may be a composition (1a) of multiple policies the server holds, e.g., it may contain a number of alternative policies of which one must be satisfied to obtain access. It contains the *card requirements* expressing which conditions on which card attributes have to hold, which attributes have to be revealed to whom, and who the server trusts as issuers for these cards. For each attribute to be revealed, the server may also specify in a *data handling policy* what he aims to do with the obtained information, e.g., how long he will retain the data, or for which purposes he will use it.

Upon receiving the policy, the user's system evaluates which claims she can derive from her available cards that fulfill the given policy (2a). For example, a policy requiring a valid travel document may be fulfilled by means of the user's national identity card or her passport, while the validity may be shown by proving its expiration date to be in the future or by disclosing the exact date. The favored claim is then chosen interactively by the user, or automatically by a data-minimizing heuristic (2b). In addition, the user decides whether she agrees with the server's data handling policy, and whether she wants to proceed. In this case, evidence for the chosen claim is generated (2c) and sent, together with the claim and the attributes to reveal, to the server (3). As claims are expressed independent from a concrete technology, the user's systems must have respective means available (e.g., in the form of a plug-in for technology t_a) to generate evidence specific to the technology of the used cards.

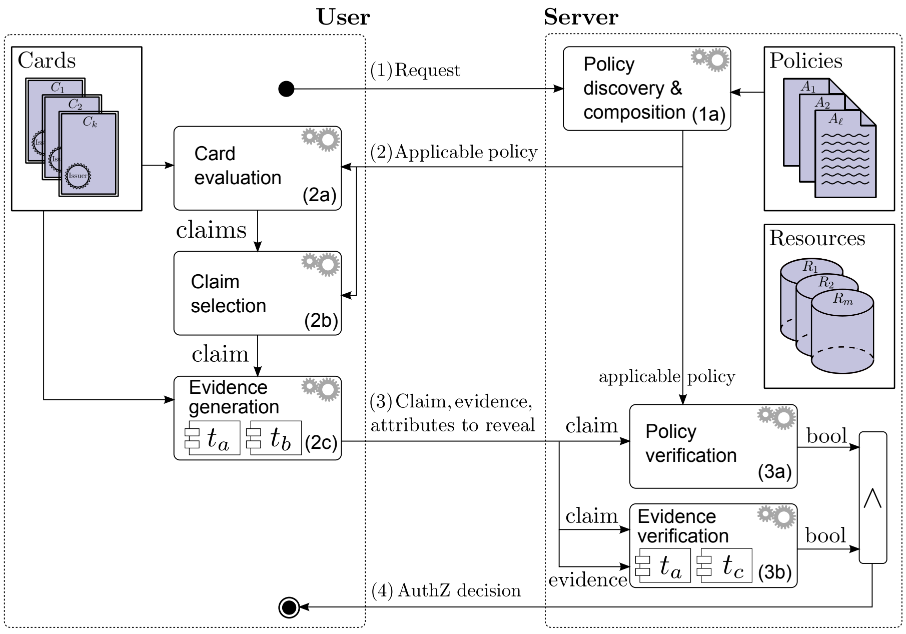


Figure 1: Decision rendering in our system model

Finally, the server verifies whether the policy is implied by the claim (3a) and if the evidence supports the validity of the claim (3b). If so, access to the resource is granted (4). Clearly, to verify the evidence the server must support the same technology that was used to generate the claim.

Finding possible claims in step (2a) could, e.g., be done by finding logical proofs with the policy formula as proof goal using the available cards as premises. A user could then select one proof and according to the proof-carrying paradigm [1] this proof would be sent to the server as sample solution for the claim verification in step (3a).

To implement the scenario sketched above, a number of languages need to be available. In particular, languages to express the access control policy on the server (which includes the card requirements and the data handling policies), to communicate the policy to the user, to express the claims as sent by the user, and to describe the cards that the user owns. In case the user's evaluation of the data handling policy shall be automated, also a language to express a user's privacy preferences is needed.

In this paper, we focus on the card requirements language, which is used as part of the access control policy on the server, as well as of the communicated policy to the user. The language contains placeholders for the data handling policies associated to revealed attributes, but we do not make this language explicit. We do not make any of the other languages explicit either.

4 Language

The language we propose is intended for expressing requirements on cards that have to be fulfilled in order to gain access to some resource. These requirements involve (1) the ownership of cards of the right type and issued by the right authority, (2) the disclosure of attributes certified in these cards to the verifier or to third parties, and (3) conditions on attributes expressed in a mathematical formula. In addition, (4) the signing of statements as well as (5) the consumption of cards can be required. To fulfill a policy, *all* requirements stated in it must be fulfilled.

A CARL policy can only be fulfilled 'as a whole' (cf. Section 5.2). To combine multiple policies, e.g., offering a choice among a set of alternative claims to satisfy, they need to be embedded into other languages. In Appendix D.2 we describe how this can be done for XACML.

Our language is typed, therefore we begin with depicting the basic properties of the typing. Then we illustrate how the particular requirements are expressed in our language.

4.1 Typing and Ontology

To help policy designers avoid specification and interpretation mistakes, we make use of a type system in a similar way as many programming languages do. We distinguish between *card types* and *data types*. We first motivate the use of card types and then describe the type system's properties.

For a server to make access control decisions, he needs to distinguish the different kinds of cards and attributes he processes, as their meanings may differ depending on their type. For example, consider a university issuing digital student IDs and diploma certificates having the same basic format (e.g., both contain the student's name and field of study). These cards do clearly have different purposes and must be distinguishable from one another. Relying only on the fact of who issued a card would be insufficient for determining its purpose and trustworthiness. Also, standard attributes such as 'name' have meaning only in conjunction with additional information such as a card type. To this end, we make use of card types (e.g., drivers license, residence permit) that specify the attributes contained in cards of this type.

We therefore assume as given an ontology $\mathfrak{T}$ that defines a set of data types, a set of card types, a partial order on those card types, as well as a set of functions and relations on the data types.

First, we assume that the ontology defines the data types $\beta_1, \ldots, \beta_n$. Examples of data types include *Int*, *String*, *Date*, *Boolean* and *URI*. Data types need not necessarily be disjoint, e.g., URIs are typically also strings. We denote by $[\![\beta]\!]$ the extension of a data type β (the set of constants of type β). A constant c can have several types, i.e., every type β such that $c \in [\![\beta]\!]$.

Further, we assume that the ontology specifies a set of card types $\tau_1, \ldots, \tau_m$. These types are similar to record types in programming languages: every card type τ is defined by a set of attributes with their types $A_\tau = \{a_1 :: \beta_1, \ldots, a_l :: \beta_l\}$, where we denote with $a :: \beta$ an attribute a that has type β. Like a record in a programming language, a card can be represented as a function from the attributes to values of the respective attribute type. For instance a card with attributes $\{a_1 :: \beta_1, a_2 :: \beta_2\}$ is a function f with domain $\{a_1, a_2\}$ such that $f(a_1) \in [\![\beta_1]\!]$ and $f(a_2) \in [\![\beta_2]\!]$. Thus, the extension of a card type τ (the set of all cards of type τ) is defined as the set of all such functions:

$$[\![\tau]\!] = \{f :: (A_\tau \to [\![\beta_1]\!] \cup \ldots \cup [\![\beta_n]\!]) \mid \forall (a :: \beta) \in A_\tau .\, f(a) \in [\![\beta]\!]\}$$

Additionally, like classes in object-oriented programming languages, a card type τ_1 may be *extended* by another card type τ_2, i.e., τ_2 *inherits* all attributes of τ_1 and includes further attributes. This induces a partial order $\leq_{\mathfrak{T}}$ on card types such that if $\tau_2 \leq_{\mathfrak{T}} \tau_1$ then $A_{\tau_1} \subseteq A_{\tau_2}$. Thus every card of type τ_2 may be used in place of τ_1. For example, *Passport* $\leq_{\mathfrak{T}}$ *PhotoID* models that *Passport* is lower in the type hierarchy than (i.e., a subtype of) *PhotoID* and can be used in place of it.

Finally, the ontology should define a set of functions and relations on the data types. We assume to have at least the equivalence relation $=_\beta$ on every data type β. We also assume the addition and a total order on both types, *Int* and *Date*. Functions may depend on the state of the access control system on which it is evaluated, e.g., the function *today*() can be defined to return the current date. For a function symbol f of the ontology, we denote by $f^{\mathfrak{T}}$ the function defined for this symbol by the ontology, similarly the relation symbol R is interpreted as the ontology-defined relation $R^{\mathfrak{T}}$.

In the following we only consider specifications that are type correct w.r.t. the type inference system given in Appendix C.

4.2 Expressing requirements

We now describe how the different kinds of card requirements are expressed in our policy language. We first give a basic intuition for our language by means of an example policy, and then use it as a running example to discuss the different elements of our language in detail. The full grammar of our language can be found in Appendix B.

```
01: own $p::Passport$ issued-by $\text{USA}\textsc{gov}$
02: own $r::ResidencePermit$ issued-by $\textsc{PittsbghTownhall}$
03: own $c::CreditCard$ issued-by $\textsc{Visa}, \textsc{Amex}$
04: reveal $c.number$, $c.expDate$ under ‘purpose=payment’
05: reveal $r.address$ to $\textsc{ShipCo}$ under ‘purpose=shipping’
06: sign ‘I agree with the general terms and conditions.’
07: where $p.dateOfBirth \leq dateMinusYears(today(), 21) \wedge$
08:         $c.expDate > today()$
```

The policy states that access is granted to users who (1) are at least twenty-one years old, (2) reveal their valid credit card information for payment purposes, (3) reveal their address to the shipping company ShipCo for shipping purposes and (4) agree to the general terms and conditions. Here, an American passport must certify the age, the payment data must be certified by a valid Visa or American Express card, and a residence permit from the city of Pittsburgh must certify the address. We assume the ontology defines the functions *dateMinusYears*(*ref*, *k*), subtracting *k* years from date *ref*, and *today*(), returning the current date.

Proof of ownership

The most basic requirement expressible is the ownership of a card of a specific type by a specific issuer. For each required card the policy contains a line of the form (cf. lines 01–03 in the example policy):

$$\text{own } c :: \textit{Type} \text{ issued-by } I_1, \ldots, I_n$$

Here, c is a *card variable* used to refer to this card within the policy. We already discussed card type ontologies in Section 4.1; we assume that each type *Type* is unambiguously defined by a uniform resource identifier (URI). In the same way, we assume that the allowed card issuers $I_1, \ldots, I_n$ are referred to by means of URIs. Depending on the underlying technology, these URIs may be mapped to public keys (for X.509 and anonymous credentials), server URLs (for LDAP and OpenID), or any other way of pointing to a card issuer. Specifying multiple issuers indicates that any of these issuers is accepted. Omitting the issued-by clause indicates that no restrictions are imposed on the issuer, meaning that even self-issued cards are allowed.

Attribute disclosure

To require disclosure of a card attribute’s value, a line of the form (cf. lines 04–05 in the example policy) is stated:

$$\text{reveal } c.att \text{ to } \textsc{Recipient} \text{ under } dhp$$

The attribute to reveal is specified by the card variable c as defined in a previous own line, and the attribute name *att* as defined by the card type, separated by a dot.

By using the keyword to, the policy can optionally specify the recipient (identified by a URI) to whom the attribute is to be revealed. If no recipient is specified, then the intended recipient is assumed to be the server enforcing the access control. For proving the policy, the server has to be provided with evidence showing that the attribute values were indeed transmitted to the recipient. In the example policy, the credit card data has to be revealed to the server itself (line 04) and the address to the shipping company (line 05).

Optionally, the server may attach a data handling policy *dhp* describing how the server intends to treat the information after receiving it, e.g., the intended purpose of the data, retention periods, further recipients, etc. The policy could be specified in natural language or in a machine-interpretable language like P3P [36]; we do not impose any particular format though.

Conditions on Attributes

Using the keyword where, a policy may state a formula ϕ expressing conditions on the cards' attributes. For example, lines 07–08 require the date of birth to be at least 21 years in the past and the credit card to be valid.

In general, a formula allows for applying the standard operators for comparison (numeric and non-numeric), arithmetics and logics (cf. Appendix B). Functions and relations may be applied to expressions whereby we assume a built-in standard set of these (such as date and time arithmetics, string manipulation, etc.) defined in the ontology $\mathfrak{T}$ together with their meaning. Expressions may further be qualified attributes, constants or basic variables. Basic variables are different from card variables as they are of a data type, rather than a card type. Basic variables that appear in ϕ may act as substitute for (1) a card's issuer, (2) an attribute that has to be revealed, (3) the purpose and (4) recipient of attributes to reveal as well as (5) the statement to sign. Our running example could, e.g., as well have been written as own $c{::}\mathit{CreditCard}$ issued-by i while extending ϕ with $i = \textsc{Visa} \lor i = \textsc{Amex}$. In case a basic variable x appears in the list of attributes to reveal, the concrete value for x, under which ϕ is satisfied, must be revealed (cf. Section 5.3).

Signing of statements

By including a line of the form

sign statement

a policy requires to sign the statement in the following sense: it is ensured that the signature was produced by someone who fulfills (the rest of) the policy. For example, line 06 of the policy example requires signing of the statement 'I agree with the general terms and conditions'.

Consumption control

Limitations on the number of times the same card can be used to obtain access can be imposed by lines of the form

$$\textsf{consume}\ \mathit{amount}\ \textsf{maximally}\ \mathit{limit}\ \textsf{of}\ c\ \textsf{scope}\ \mathit{scope}$$

where *amount* is the consumption amount, *limit* is the consumption limit, c is the card to be consumed, and *scope* is the ratification scope (cf. Section 3.3).

For example, the Pittsburgh Theater hands out discount cards entitling its holder to buy theater tickets at a reduced price. The below policy protects the theater's discounted ticketing service and states that students of the University of Pittsburgh are eligible for six discounts per year:

```
own sid::StudentID issued-by PittsbghUniversity
own dc::DiscountCred issued-by PittsbghTheater
consume 1 maximally 6 of dc scope s
where s = append('urn:scope:pbgTheater:year:', currYear())
```

To access the service, proofs of ownership for a student card and a discount card need to be provided to the server. For granting access, the server verifies that the sum of the amounts of all consumptions of the discount card (including the current one) in the given scope does not exceed the limit. Each time a student successfully purchases a discounted ticket, one unit is consumed. (More units could, e.g., be charged for extra-long shows.)

5 Semantics

We now provide a formal semantics for our language. The semantics abstractly defines the intended behavior of an access control system for a given policy and thereby defines the obligations that an actual realization must meet. The context of this definition is a state transition system with transitions for issuing cards, proving a policy, and other transactions such as revocation of cards or application-specific actions. As our CARL is concerned only with the conditions for fulfilling a policy and the effects of proving a policy, so is our semantics.

We define the formal semantics of CARL in three steps. First, we define the meaning of a policy formula based on the ontology and an interpretation of the variables. Second, for a given policy and a set of cards that a user owns, we define if the user fulfills the policy. Third, a user who fulfills the policy can prove this fact to the server and we call this simply *proving the policy*; we define the effect that proving the policy has on the knowledge of the server and the trusted third parties.

Note that, in the last step, we define only the *effect* of proving a policy, rather than describing how such a proof could be performed. This is because such actions depend on the concrete technology that is employed, from which our language abstracts. Indeed, when mapping CARL onto a concrete technology by translating the policy specification into a sequence of actions within the respective card system, our semantics sets out two obligations for this translation: 1) actions can only be performed by a user who owns the necessary cards and 2) other parties learn from the action exactly the information specified by the semantics. As not all technologies offer the same level of privacy protection, we also provide a variant of our semantics that allows for more information to be disclosed than what is required by the policy. An example for such a mapping is provided by Mödersheim and Sommer for the Identity Mixer anonymous credential system [17].

5.1 Formula Semantics

We define the meaning of policy formula with respect to a given ontology $\mathfrak{T}$ and an interpretation $\mathcal{I}$ that maps all basic and card variables to values of the respective type. Recall that the ontology provides the interpretation $f^{\mathfrak{T}}$ for every function symbol f and $R^{\mathfrak{T}}$ for every relation symbol R, respectively. For a card C, an attribute a, and terms $t_1, \ldots, t_n$, we define $(C.a)^{\mathcal{I}} = (C^{\mathcal{I}})(a)$ and $f(t_1, \ldots, t_n)^{\mathcal{I}} = f^{\mathfrak{T}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$.

Our typing system ensures that all these values are well-defined for a given $\mathcal{I}$. The satisfaction relation is defined as:

$$\mathcal{I} \models \phi_1 \wedge \phi_2 \text{ iff } \mathcal{I} \models \phi_1 \text{ and } \mathcal{I} \models \phi_2 \ , \quad \mathcal{I} \models \neg\phi \text{ iff } \mathcal{I} \not\models \phi \ ,$$

$$\mathcal{I} \models R(t_1, \ldots, t_n) \text{ iff } R^{\mathfrak{T}}(t_1^{\mathcal{I}}, \ldots, t_n^{\mathcal{I}})$$

Other constructs of the CARL formulae are the usual abbreviations, e.g., $\phi_1 \vee \phi_2$ is short for $\neg(\neg\phi_1 \wedge \neg\phi_2)$. Further, we define $\phi_1 \models \phi_2$ iff for all $\mathcal{I}$ holds $\mathcal{I} \models \phi_1$ implies $\mathcal{I} \models \phi_2$. Finally, we define $\phi_1 \equiv \phi_2$ iff $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$. These definitions are similar to other definitions in formal logic; following these standards improves the understanding of the concepts described by our language.

5.2 Fulfilling a Policy

The second step is concerned with the question whether a user owns cards sufficient to satisfy a given policy. We denote with $\mathfrak{P}_U$ the set of cards that the user U owns. We now consider the local state of the parties, the conditions on a party's state to fulfill a policy, and the transition of parties' states when proving policies. For now, we assume that a U's state contains at least the set of cards $\mathfrak{P}_U$ the party U owns.

We first look at a policy without consume lines, which we consider in Section 5.4. Let U be a user and S be a server and U be known to S under the identifier I. This identifier may for instance be a one-time pseudonym in privacy-friendly technologies or (linkable) username otherwise.

Let P be the policy of S that U needs to satisfy in the following form:

own $C_1 :: \tau_1, \ldots, C_n :: \tau_n$
reveal $t_{(1,S)}, \ldots, t_{(n_S,S)}$
reveal $t_{(1,S_1)}, \ldots, t_{(n_{S_1},S_1)}$ to S_1
$\ldots$
reveal $t_{(1,S_m)}, \ldots, t_{(n_{S_m},S_m)}$ to S_m
where ϕ
sign s

We require that the variables of P and of the current state are disjoint (which can be achieved by a suitable renaming of the policy variables). Note that we here omitted the issued-by part of the own line because we treat the issuer of a card simply as an attribute (of type URI) and the issuer can thus be specified as part of the formula ϕ itself. Indeed, issued-by $I_1, \ldots, I_n$ for a card C is just an abbreviation for $C.issuer = I_1 \vee \ldots \vee C.issuer = I_n$ (as an additional conjunct of ϕ).

We say that *U can fulfill the policy P under card assignment p and interpretation $\mathcal{I}$* iff

1. p is a total mapping from $\{C_1, \ldots, C_n\}$ to $\mathfrak{P}_U$,

2. for every $1 \leq i \leq n$ it holds that $p(C_i) :: \sigma_i$ for a type $\sigma_i \leq_{\mathfrak{T}} \tau_i$ and $C_i^{\mathcal{I}} = (A_{\tau_i} \lhd p(C_i))$ where $A \lhd f$ is the domain restriction of function f to set A, and

3. $\mathcal{I} \models \phi$.

We say that *U can fulfill the policy P* iff there exists such an interpretation $\mathcal{I}$ and mapping p.

5.3 Proving a Policy

When a user can fulfill a policy and chooses to prove the policy to another participant, the effect of this transition is an increase of knowledge of all participants to which information is revealed. We denote with $\mathfrak{K}_S(I)$ and $\mathfrak{K}'_S(I)$ formulae characterizing the knowledge of a party S about a user I before and after the transition, respectively. As before, I is an identifier under which the user is known to S.

For each part $O \in \{S, S_1^{\mathcal{I}}, \ldots, S_m^{\mathcal{I}}\}$ to whom information is revealed, we require

$$\mathfrak{K}'_O(I) = \mathfrak{K}_O(I) \wedge \phi \wedge (t_{(1,O)} = t_{(1,O)}^{\mathcal{I}}) \wedge \ldots \wedge (t_{(n_O,O)} = t_{(n_O,O)}^{\mathcal{I}}) .$$

This models that each party O learns that the user acting under the identifier I owns cards with property ϕ together with the concrete values $t_{(i,O)}^{\mathcal{I}}$ of the terms $t_{(i,O)}$ that have been revealed to O. Additionally, the server S obtains a signature on the statement $s^{\mathcal{I}}$ with respect to the formula proved to S, i.e., $\phi \wedge (t_{(1,S)} = t_{(1,S)}^{\mathcal{I}}) \wedge \ldots \wedge (t_{(n_S,S)} = t_{(n_S,S)}^{\mathcal{I}})$.

To model unlinkable values, we use variables in a particular way here. For this, first recall that we require a renaming of the policy variables so that no variable occurs in the considered state. Therefore, when a user uses the same card several times in proving policies to the same or to different servers, this card is referred to by a fresh variable each time. This reflects that the servers, even when they cooperate, are unable to decide whether particular authentications have been performed with the same or with different cards. However, for what concerns all the information revealed by a single transaction for fulfilling a policy P to several different parties, the additional knowledge uses the same variable names for all parties. For two servers S and S' who cooperate, we define their shared knowledge about a user identified as I simply as: $\mathfrak{K}_{S,S'}(I) = \mathfrak{K}_S(I) \wedge \mathfrak{K}_{S'}(I)$. This reflects that they can relate only information about a transaction that was done under the same pseudonym I.

For proving a policy in a typical system realization, the user's system creates a claim that implies the policy to be fulfilled and sends it with accompanying evidence to the server (cf. Section 3.4).

Our definition of the knowledge of servers characterizes the ideal case of privacy-friendly technologies: one does not reveal more information than required by the policy. Conventional card systems cannot achieve this as cards are transmitted as a whole. Thus, for conventional technologies we relax the constraints of our semantics, allowing for the release of more information than necessary, formally, any formula ψ that implies the above $\mathfrak{K}'_O(I)$ formula.

5.4 Consumption Control

We now extend the semantics to specify the behavior of consumable cards. In our model, consumption of cards is tied to a ratification scope that controls the consumption (cf. Section 3.3). In a nutshell, one can spend a card if the value of all its consumptions (including the current one) in the ratification scope does not exceed the consumption limit.

To model the consumption, we introduce global *log files*, one log file for each consumption scope (these log files are only part of the ideal world that is modeled by our transition system; they usually will not have a counterpart in an real implementation). Each log file is a list of pairs $\langle C, k\rangle$ of a card C and a positive integer k, representing that k units of card C are spent in the domain represented by the log file. The *balance of card* C in a log file is the sum of all consumptions of C in the log file.

Consider a policy P as before with the following additional consumption line (where C is a card of some own line of P):

consume k maximally n of C scope *scope*

We define that P can be fulfilled for $\mathcal{I}$ and p if 1) the policy resulting from P by removing the consumption line according to the definition in Section 5.3 can be fulfilled and 2) the balance of card $p(C)$ in the log file of $scope^{\mathcal{I}}$ plus $k^{\mathcal{I}}$ is less than or equal to $n^{\mathcal{I}}$.

Similarly, we extend the definition of proving P, i.e., it will have the same effects as before, plus the effect that the log file of $scope^{\mathcal{I}}$ is augmented with the pair $\langle p(C), k^{\mathcal{I}} \rangle$.

We covered here only a policy with one consumption line, the extension to several consumption lines is as expected, where we assume that the interpretation of the consumption scope $scope^{\mathcal{I}}$ (which usually is a constant chosen by the server) is never the same for different consumption lines.

6 Conclusions & Future Work

We presented a language for specifying requirements on a user's cards to be used for obtaining access in any kind of open access control setting. We not only provide a formal language specification, but also formally define the semantics as to help future systems designers and implementers to avoid mistakes through ambiguous interpretation of the text. Our language aims to serve as a central piece in an open access control setting. It enables the use of a plurality of underlying card technologies that are available today or are becoming available, such as OpenID or anonymous credentials. Each card technology can be used for access control by providing a mapping from the verification process of the respective technology to a policy in CARL. Our language enables properties of privacy and anonymity through data minimization while retaining accountability. When being deployed together with technologies such as anonymous credential systems, a privacy-preserving user-centric model of access control can be realized as the user is put into full control over her data.

Our next step towards an open and privacy-enhancing system for access control is to link the feature set and formalism of CARL with concrete card systems. Mödersheim and Sommer [17] describe such a connection for the Identity Mixer anonymous credential system. A proof that this connection agrees with the CARL semantics, as well as providing similar connections for other technologies are part of our ongoing work.

7 Acknowledgments

The authors thank Mario Verdicchio and Günter Karjoth for fruitful discussions. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) for the projects "PrimeLife" (grant agreement no. 216483) and "AVANTSSAR" (grant agreement no. 216471).

References

[1] Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *ACM CCS 99: 6th Conference on Computer and Communications Security*, pages 52–62, Kent Ridge Digital Labs, Singapore, November 1–4, 1999. ACM Press.

[2] C. A. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. A privacy-aware access control system. *Journal of Computer Security*, 16(4):369–397, 2008.

[3] Claudio A. Ardagna, Jan Camenisch, Markulf Kohlweiss, Ronald Leenes, Gregory Neven, Bart Priem, Pierangela Samarati, Dieter Sommer, and Mario Verdicchio. Exploiting cryptography for privacy-enhanced access control. *Journal of Computer Security*, 18(1):123–160, 2010.

[4] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.2). `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/`, November 2003.

[5] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[6] Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 40–46, Alexandria, VA, USA, 2005. ACM.

[7] Moritz Becker, Cedric Fournet, and Andrew Gordon. Design and semantics of a decentralized authorization language. In *Proc. of the20th IEEE Computer Security Foundations Symposium (CSF)*, pages 3–15, Washington, DC, USA, 2007. IEEE Computer Society.

[8] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. of 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, California, USA, May 1996. IEEE.

[9] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos Keromytis. The role of trust management in distributed systems security. In Jan Vitek and Christian Jensen, editors, *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer Berlin / Heidelberg, 1999.

[10] P.A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[11] Kevin D. Bowers, Lujo Bauer, Deepak Garg, Frank Pfenning, and Michael K. Reiter. Consumable credentials in linear-logic-based access-control systems. In *ISOC Network and Distributed System Security Symposium – NDSS 2007*, San Diego, California, USA, February 28 – March 2, 2007. The Internet Society.

[12] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy.* PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.

[13] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04: 11th Conference on Computer and Communications Security*, pages 132–145, Washington D.C., USA, October 25–29, 2004. ACM Press.

[14] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345, Kyoto, Japan, December 3–7, 2000. Springer, Berlin, Germany.

[15] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 201–210, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press.

[16] 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.

[17] Jan Camenisch, Sebastian Mödersheim, and Dieter Sommer. A formal model of identity mixer. In *Proc. of the 15th International Workshop on Formal Methods for Industrial Critical Systems (FMICS)*, Lecture Notes in Computer Science. Springer, 2010.

[18] 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 17–21, 2003. Springer, Berlin, Germany.

[19] 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.

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

[21] Microsoft Co. Microsoft's vision for an identity metasystem. `http://msdn.microsoft.com/en-us/library/ms996422.aspx`, May 2005.

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

[23] 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.

[24] Credentica. U-Prove SDK overview: A Credentica white paper. `http://www.credentica.com/files/U-ProveSDKWhitepaper.pdf`, 2007.

[25] David Ferraiolo and Richard Kuhn. Role-based access control. In *Proc. of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[26] Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of authorization and knowledge. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS 2006: 11th European Symposium on Research in Computer Security*, volume 4189 of *Lecture Notes in Computer Science*, pages 297–312, Hamburg, Germany, September 18–20, 2006. Springer, Berlin, Germany.

[27] S. Gevers and B. De Decker. Automating privacy friendly information disclosure. Technical Report CW441, K.U. Leuven, Dept. of Computer Science, April 2006.

[28] Yuri Gurevich and Itay Neeman. DKAL: Distributed-knowledge authorization language. In *Proc. of the21th IEEE Computer Security Foundations Symposium (CSF)*, pages 149–162, June 2008.

[29] D. Hardt, J. Bufu, and J. Hoyt. OpenID attribute exchange 1.0. `http://openid.net/developers/specs/`, December 2007.

[30] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05: 12th Conference on Computer and Communications Security*, pages 46–57, Alexandria, Virginia, USA, November 7–11, 2005. ACM Press.

[31] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information and System Security*, 6(1):128–171, February 2003.

[32] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.

[33] 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.

[34] 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.

[35] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[36] W3C. The platform for privacy preferences 1.1 (P3P1.1) specification. `http://www.w3.org/TR/P3P11/`, November 2006.

[37] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *ACM Workshop on Formal Methods in Security Engineering (FMSE)*, pages 45–55. ACM, 2004.

[38] W.H. Winsborough, K.E. Seamons, and V.E. Jones. Automated trust negotiation. In *Proc. of the DARPA Information Survivability Conference and Exposition (DISCEX)*, volume 1, pages 88 –102 vol.1, 2000.

A Other Card Technologies

OpenID In OpenID [22] each account is identified by a unique URL bound to the user by the OpenID provider. The recent OpenID Attribute Exchange extension [29] allows for the exchange of user-defined attributes. When logging on to a website, the user sends her unique URL to the relying party. The user is then redirected to her OpenID provider where she authenticates using a password. If the authentication is successful, the OpenID provider either sends a confirmation and the requested attributes directly to the relying party, or sends these through the user with integrity protection with a MAC (under a key that the provider and relying party agreed upon directly).

An OpenID account can be seen as a card issued by the OpenID provider. Any subset of attributes can be revealed in a claim, which is good for privacy, but on the negative side all transactions with the same card are linkable through the unique URL. Also, since claim creation and verification both involve the OpenID provider, he learns which user authenticates to which server at which time. The user's password acts as a proof of ownership; the integrity of the attributes is protected by MACs.

Kerberos A Kerberos user account [32] could also be seen as a card issued by the key distribution center (KDC) containing the user's identity. Each claim derived from this card contains the user's identity; the Kerberos ticket acts as the evidence. The pre-evidence is the user's password and perhaps connection information to the KDC. Both claim creation by the user and verification by the relying party require interaction with the KDC.

SAML SAML [33] is an XML-based framework for communicating user authentication, entitlement, and attribute information. The user's attributes are stored by an Identity Provider (IP). When logging in to a relying party, the user authenticates to the IP and requests a signed claim (called *assertion* in SAML) containing the attribute values that the relying party requires. SAML is somewhat technology agnostic in the sense that it does not specify how the user has to authenticate to the identity provider, but the evidence of the claim is always an XML Signature by the IP. The user's account at the IP containing all the attributes can be seen as a card in our framework. The IP is the issuer of the card, the user's pre-evidence consists of her authentication secret with respect to the IP.

Email Even a simple email account can be seen as a card: the only attribute is the email address itself, the mail server acts as the issuer. The user's pre-evidence is her account password, the integrity and ownership of the email address are checked by the owner's ability to click on a link or enter a code sent to her by email.

B Grammar

The following shows the grammar of our policy language:

Policy	=	CredDef$^+$ RevealDef* SpendDef* [‘where‘ Formula] [SignDef] ;
CredDef	=	‘own‘ CredVar ‘::‘ CredTypeIdent [‘issued-by‘ IdentityTerm (‘,‘ IdentityTerm)*] ;
RevealDef	=	‘reveal‘ ValueList [‘to‘ IdentityTerm] [‘under‘ Term] ;
SpendDef	=	‘consume‘ Term ‘maximally‘ Term ‘of‘ CredVar ‘scope‘ Term ;
SignDef	=	‘sign‘ Term ;
CredVar	=	Identifier ;
CredTypeIdent	=	Identifier ;
IdentityTerm	=	Term ;
ValueList	=	Value [‘::‘ TypeIdent] [‘,‘ ValueList] ;
TypeIdent	=	Identifier ;
Formula	=	Formula ‘∧‘ Formula
	|	Formula ‘∨‘ Formula
	|	‘¬‘Formula
	|	‘(‘ Formula ‘)‘
	|	Exp RelationOp Exp
	|	RelationName ‘(‘ [ExpList] ‘)‘ ;
Exp	=	Term
	|	Exp ArithmeticOp Exp
	|	FunctionName ‘(‘ [ExpList] ‘)‘ ;
RelationOp	=	‘=‘ | ‘<‘ | ‘>‘
	|	‘≤‘ | ‘≥‘ | ‘≠‘ ;
Term	=	Value | Constant ;
ArithmeticOp	=	‘+‘ | ‘−‘ | ‘·‘ | ‘÷‘ ;
Value	=	CredVar ‘.‘ AttrIdent
	|	BasicVar ;
ExpList	=	Exp [‘,‘ ExpList] ;
RelationName	=	Identifier ;
FunctionName	=	Identifier ;
Constant	=	Identifier ;
AttrIdent	=	Identifier ;
BasicVar	=	Identifier ;
Identifier	=	Alpha Alphanum* ;

Alpha and Alphanum are alphabetic and alphanumeric characters. IdentityTerm must map to the URI of an identity. CredTypeIdent must map to a credential type.

TypeIdent must map to a data type. An Identifier cannot be a keyword.

C Type Inference System

To limit specification mistakes in CARL policies, we have introduced typing and though types can be automatically inferred in many cases, it is recommended for the policy writer to explicitly specify the types of all variables, both basic and credential variables.

We describe the type system now by a type inference system similar to the ones used in the definition of programming languages. We use type judgments of the form $\Gamma \vdash t :: \tau$ to denote that under the type definitions in Γ, it can be derived that term t is of type τ. Further, we use type judgments of the form $\Gamma \vdash \phi$ to express that ϕ is type correct according to Γ. Initially, the set Γ contains the following. First, it includes the types for all constant, function, and relation symbols that are defined by the ontology. Note that we do allow several type definitions for the same symbol, e.g. '=' is a comparison relation for each type, e.g. $\mathbb{P}(Int \times Int)$; similarly, e.g., a string constant may also have type *URI*. This models the overloading of symbols. Second, the initial Γ includes a unique type for every variable of the specification. Since for basic variables the type specification is optional, one must "guess" a type for each unspecified variable initially and verify that with that guess, the policy is well-typed. Note that variables for an issuer or the recipient of revealed data are always of type *URI*.

The axiom is that we can derive anything given:

$$\frac{}{\Gamma \cup \{t :: \tau\} \vdash t :: \tau}$$

We can now derive types of terms and formulae:

$$\frac{\Gamma \vdash t_1 :: \alpha_1 \quad \ldots \quad \Gamma \vdash t_n :: \alpha_n \quad \Gamma \vdash f :: \alpha_1 \times \ldots \times \alpha_n \to \beta}{\Gamma \vdash f(t_1, \ldots, t_n) :: \beta}$$

$$\frac{\Gamma \vdash C :: \tau}{\Gamma \vdash C.a :: \beta} \ (a :: \beta) \in A_\tau$$

$$\frac{\Gamma \vdash t_1 :: \tau_1 \quad \ldots \quad \Gamma \vdash t_n :: \tau_n \quad \Gamma \vdash R :: \mathbb{P}(\tau_1 \times \ldots \times \tau_n)}{\Gamma \vdash R(t_1, \ldots, t_n)}$$

$$\frac{\Gamma \vdash \phi_1 \quad \Gamma \vdash \phi_2}{\Gamma \vdash \phi_1 \wedge \phi_2} \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \neg\phi}$$

Given a formula ϕ and an initial set of type definitions Γ, we say that ϕ is *well typed* iff $\Gamma \vdash \phi$ can be derived. If there is more than one derivation, we say that ϕ is *ambiguous*. While it is common that subterms of a formula may have different type derivations, there should be only one derivation for the entire formula. Otherwise, there is probably a typing problem. This could, e.g., lead to using a wrong instance of an overloaded function symbol. In such a case, the policy specifier should thus be advised of the ambiguity.

D Language Integration

The language we present addresses the specification of credential requirements in credential-based access control. In this section we sketch how our language can be integrated with other languages to utilize the strengths of those.

D.1 Abstract Authorization Languages

For expressing authorization, delegation, and trust, e.g., within a large organization, a complete enumeration of business units, employees, and their relations and access rights is usually not feasible and too inflexible. Therefore, authorization languages such as SecPal [7] and DKAL [28] have been developed. They are based on a simple but powerful mechanism: they specify Horn clauses on abstract facts, i.e., rules of the form "if facts $f_1, \ldots, f_n$ hold, then also fact f holds." The facts represent either direct statements, e.g., "X says Y", initially given relationships such as trust relationships, or derived facts that are consequences of other facts by the Horn-clauses.

All these facts thus represent a high-level view that is not related to concrete credentials, signatures and the like. Our CARL language can provide this relationship, i.e. specifying precisely what credentials and properties correspond to a particular abstract fact like "X says Y" or "Z is-an-adult". A connection between CARL and Horn-clause based authorization language can thus be made by specifications of the form $f \Leftarrow P$ for an abstract fact f and a CARL policy P. This means that one way to derive the fact f is to fulfill the policy P.

For example, one may specify with Horn clauses that X can access resource R if owner O of R has given permission to X. Permission can be given directly by O, or O could have delegated that to a deputy D and D granted access. CARL policies specify the concrete credentials and properties that correspond to the access permission from O or from D as well as a credential for the delegation from O to D. The combination tells us that one may either access R when (1) one has the credential that corresponds to the permission from O or (2) when there are credentials that show that O delegated this right to D (this may already be known by the respective policy enforcement point) and the credential that D issued the permission for access.

Combining the CARL with the abstract facts of a Horn-based language gives a powerful and high-level method to specify complex authorization, trust, and delegation rules on credentials. Observe that a delegation chain as in the example cannot be specified in CARL (because the length of the chain is unbounded). In fact, the focus of CARL is on the specification of the credentials, their conditions, and the revealing of information to different parties, exactly what the abstract Horn-based languages abstract from. The languages are thus complementary and it makes sense to distinguish between a high-level deduction engine and the connection to concrete credentials.

D.2 XACML

The access control language XACML [34] is the de facto standard for attribute-based access control policies. In order to reuse existing XACML infrastructure for privacy-friendly credential-based access control, we briefly describe how CARL can be integrated into XACML.

First, we define a number of new XML elements that can occur inside an XACML `<Rule>` to make XACML credential-aware. Each own, reveal, sign, and consume line in CARL is translated into a corresponding `<Own>`, `<Reveal>`, `<Sign>`, and `<Spend>` element in the XACML `<Rule>`. The schema of these new elements is such that they encode in a structured way all arguments on the corresponding lines in a CARL policy. Each `<Own>` element has an attribute `CredentialId` containing a URI by which this credential can be referred to within this `<Rule>`. Finally, the formula ϕ is encoded within the standard XACML `<Condition>` element using built-in data types and functions [34, Appendix A], but we extend the `<AttributeDesignator>` element with an extra attribute `CredentialId` to indicate from which credential the attribute should be taken.

Second, we make a number of architectural changes to make XACML privacy-friendly. Namely, standard XACML does not provide a mechanism for conveying an access control policy to the user since it does not assume that the policy is known by the user. This is against the idea of privacy-aware access control where a user provides only the credentials/attributes necessary for fulfilling a particular policy. A possible way of conveying a policy to the user would be to embed the policy in XACML's `<StatusMessage>` element that is optionally contained in an XACML access response.

In order to solve the architectural issue, a server could run a modified XACML policy decision point (PDP) that in case an access request is denied *and* the applicable XACML policy contains a CSL specification in its `<Condition>` element, it returns these CSL requirements embedded in the `<StatusMessage>` of the negative access response. In order for a user to learn the policy for a specific resource, she makes a request without providing any attributes whereupon the PDP will respond with the negative response that contains the CSL policy. Knowing that policy, the user can provide only the attributes necessary for this particular policy.

Curriculum Vitae

Franz-Stefan Preiss was born in 1983[1] in Zwettl, Austria. He received master's degrees in *Software Engineering & Internet Computing* and in *Computer Science Management* from the Vienna University of Technology (Technische Universität Wien) in Austria in 2007. Franz-Stefan wrote his master's thesis with the title "Access Control Policy Editor and Analyzer for Policies on a Business Level" during an internship at IBM Research – Zurich in Switzerland. In May 2008, he joined the Security and Cryptography group at IBM Research – Zurich to work on a research project related to security, privacy, and identity management. Franz-Stefan started the doctoral program of KU Leuven, Belgium, in September 2009 as external doctoral student performing his research at IBM Research in Switzerland.

[1]Exact date of birth omitted for privacy reasons.