Federated Identity Management

Jan Camenisch and Birgit Pfitzmann

IBM Zurich Research Lab, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{jca,bpf}@zurich.ibm.com

Summary. The more real business and interaction with public authorities is performed in digital form, the more important the handling of identities over open networks becomes. The rise in identity theft as a result of the misuse of global but unprotected identifiers like credit card numbers is one strong indicator of this. Setting up individual passwords between a person and every organization he or she interacts with also offers very limited security in practice. Federated identity management addresses this critical issue. Classic proposals like Kerberos and PKIs never gained wide acceptance because of two problems: actual deployment to end users and privacy. We describe modern approaches that solve these problems. The first approach is browser-based protocols, where the user only needs a standard browser without special settings. We discuss the specific protocol types and security challenges of this protocol class, as well as what privacy can and cannot be achieved within this class. The second approach, private credentials, solves the problems that none of the prior solutions could solve, but requires the user to install some local software. Private credentials allow the user to reveal only the minimum information necessary to conduct transactions. In particular, it enables unlinkable transactions even for certified attributes. We sketch the cryptographic solutions and describe how optional properties such as revocability can be achieved, in particular in the idemix system.

1 Introduction

In many areas of society, transactions of increasing importance and volume are performed digitally. This concerns business-to-consumer and business-to-business commerce as well as public administration and direct interactions between individuals. In most cases, the first challenge is identity management: How do the partners recognize each other if they want to interact with a specific partner (*authentication*)? And how do they obtain the information about each other that is needed to perform the desired transaction (*attribute exchange*)? These questions become even more critical when the goal of making the transactions digital goes beyond re-implementing the paper

world, towards what is known as *on-demand business*. This means more flexible business relationships where everyone can perform every transaction with whatever enterprise or organization performs them best at just that time. On-demand business increases the identity management problem from trying to bootstrap existing relationships from the paper world to the digital world to a much more dynamic case: Where there is no prior direct relationship between two partners, all information that must be trusted requires third-party confirmation. Approaches at optimizing the exchange of identity-related information across several relationships are called *Federated Identity Management* or *FIM*. Identity management in general also has an enterprise-internal aspect of how to manage identity information received from partners consistently, with quick updates and suitable privacy; this is a significant problem in practice. In this article, however, we concentrate on recent approaches to the exchange of identity-related information between different individuals and organizations.

With respect to end users, there are two major challenges in federated identity management: Ease of use, in particular very easy start of usage, and privacy.

A market observation currently taken for granted by the major players is that a large segment of users will not install additional software or even hardware such as card readers for identity management or electronic commerce in general, neither for ease of use nor for security or privacy. Identity management protocols must accommodate this user segment, i.e., people using nothing but a commercial browser. We call such protocols *browser-based* and the feature *zero-footprint*. A similar precondition is *mobility* in the sense that a user should be able to use the protocols from varying browsers, such as several personal devices or even Internet kiosks. While we stress that authenticating via unknown browsers is dangerous for security, nobody is forced to do this (at least in developed countries). These functional requirements distinguish FIM proposals made over the last five years such as Microsoft Passport, the OASIS SAML standard, the Liberty Alliance specifications, and Web Services Federation from classical approaches such as Kerberos, PKIs, form fillers, and wallets (which we do not survey here). Very recently, however, proposals that concentrate on local clients are coming into favor again, in particular the PRIME project (`http://www.prime-project.eu`), Microsoft CardSpace (`http://www.microsoft.com/presspass/features/2006/feb06/02-14InfoCards.mspx`), and the Higgins project (`http://www.eclipse.org/higgins/`), but now at least the last two emphasize making flexible identity management a standard feature of basic client software.

Privacy surveys show consistently that 80 to 90% of all people are concerned about privacy, and that 25% are willing to pay a considerable price in money or inconvenience for it, e.g., [27, 43]. These numbers are actually increasing, partially due to the special problems of identity theft based on the misuse of widely available, unprotected numbers as authentication "secrets". It is also known that about half of all people at least occasionally give

wrong data to web sites on purpose because of privacy concerns. The privacy problem increases with on-demand business because users will interact with more partners, have less a-priori trust into most of them, and the third parties that may be needed to confirm information may need very complex privacy policies because they no longer only collect and use identity-related information for specific well-defined business purposes. Altogether it is well-accepted today that unaddressed privacy concerns would be a major inhibitor for electronic commerce, as visible by privacy and trust initiatives by major software providers such as IBM and Microsoft and the fact that privacy statements become common on e-commerce sites.

An important distinction in actual FIM protocols is between the recognition of a previously known partner, and the transfer of certified and uncertified attributes about a partner. The former is typically called authentication; it can happen under a "real" identity or under a *pseudonym*. All modern proposals for identity management allow for pseudonyms. A *certified attribute* is one that is confirmed by a third party, while an *uncertified attribute* comes witout such a confirmation. Currently, almost all attributes in electronic commerce are uncertified: Users simply input their names, addresses, credit card numbers, preferences etc. into web forms. Some attributes, such as preferences, cannot and need not be certified by nature. For other attributes it is usually contrary to the interest of the user to give a wrong one, e.g., for delivery addresses. For yet other attributes, mechanisms outside the identity management make certification unnecessary; e.g., the right to revoke unsigned credit-card transactions. However, in the future, certification of attributes will gain in importance. For instance, if a person in a business-to-business scenario claims to be an employee of a partner company, this should be certified by the partner company. Or if individuals take advice from an online medical service, they would be well advised to expect a credential of the quality of this service.

In the following, we first give an overview of browser-based FIM protocols, i.e., protocols designed to solve the first of the end-user identity-management challenges by allowing users to work entirely with standard browsers (Sections 2 to 5). We survey the basic techniques used in such protocols, emerging standards, security issues, and privacy options and limitations. In particular it turns out that privacy can be almost optimal for uncertified attributes, while it is limited for certified attributes.

This limitation is overcome by *private credentials*, presented in Sections 6 to 8. These are cryptographic protocols that essentially allow a user to transform a certified attribute obtained under one pseudonym so that it can be shown under another pseudonym of the same user in a secure but *unlinkable* way, i.e., the credential issuer and the party to whom the credential was shown cannot find out that these two pseudonyms refer to the same person. (Assuming that at least two persons hold such a credential.) The credential can be transformed and shown multiple times in unlinkable ways, unless specific options to exclude this are selected. We sketch the underlying cryptographic

ideas, in particular of the efficient and flexible *idemix* system, discuss additional features such as anonymity revocation and attribute combination, and survey the major initiatives where private credentials may come into practice soon.

We will use car rental as a running example. Figure 1 shows a simple scenario where browser-based FIM is appropriate: A person, called *user*, wants to rent a car for a business trip. The employer has a contract with a car-rental company that gives its employees favorable conditions and settles insurance issues. Hence the user must be certified as an employee of the company at the

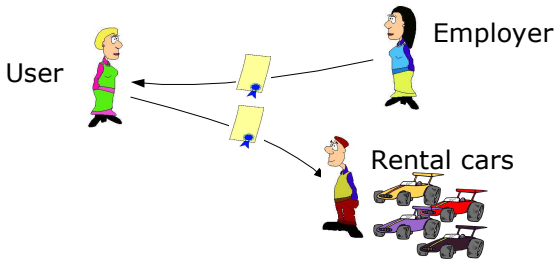

Fig. 1. Example scenario for browser-based FIM

time of the rental. Furthermore, at least during the first rental, the car-rental company may need several other attributes about the user that the employer already knows, such as name and address; these may be transferred by the FIM protocol for the convenience of the user. This is a scenario without many privacy requirements, except that the employer and the car-rental company should not exchange unnecessary information about the user. We will later also consider scenarios where the user rents the car for a private trip; we then consider stronger privacy requirements.

2 Example of a Browser-Based FIM Protocol

The easiest way to introduce the techniques used in browser-based protocols is to look at one such protocol. Figure 2 shows the message flow of the WSFPI protocol when no error occurs; WSFPI is a strict instantiation of the WS-Federation Passive Requestor Interop scenario [26] based on [29]. Strict means that we regard discretionary security-relevant constraints as mandatory and prescribe the use of secure channels. While these measures are not necessary in certain scenarios, we include them to get a general-purpose protocol.

User U at browser B wants to sign-in at a party C called *identity consumer* with the help of an *identity supplier* S where U has registered earlier. In our car-rental example, the employer is the identity supplier and the car-rental company is the identity consumer. Steps 1 and 10 show that user U is assumed

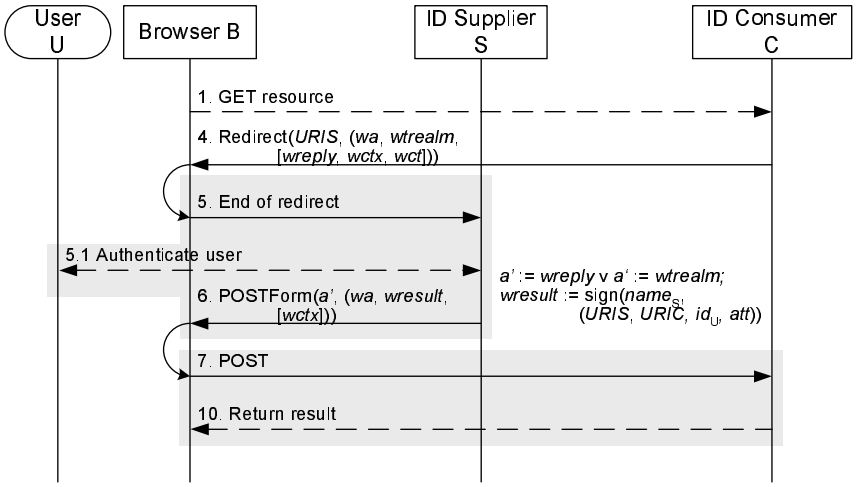

Fig. 2. WSFPI protocol with abstract parameters. Steps with uninterrupted lines are actually specified in the protocol. The gray boxes denote secure channels.

to browse at identity consumer C before the protocol (Step 1), e.g., at a car-rental front page, and to get some application-level response after the protocol (Step 10), e.g., a specific page that enables the rental under the favorable conditions agreed with the employer. To answer the request from Step 1, the identity consumer C desires some authentication or attributes from the user. Thus in Steps 4-5 the identity consumer redirects the browser to the identity supplier S. In Step 5.1, the identity supplier authenticates the user, e.g., with a username and password exchanged in registration. In Steps 6-7 the identity supplier S essentially redirects the user back to the identity consumer C with a so-called *token* denoted by *wresult* that contains the user identity id_U; in other terminologies it might be called a ticket or a credential.

The messages Redirect and POSTForm (Steps 4 and 6) abstract from the syntax of browser redirects, i.e., HTTP 302 or 303 messages, and form posting, respectively. The first parameter in Figure 2 is the address and the second parameter the payload, here a list of protocol parameters. In a real redirect, these parameters are represented in the query string, i.e., the part of a URL behind a question mark. In a form POST, an HTML form, typically including a script, induces the user or the browser, respectively, to send the payload to the given address using an HTTP POST. The end of a redirect message (Step 5) gets its parameters from the redirect message, and the POST message (Step 7) from the POSTForm message. The gray boxes in the figure denote secure channels, i.e., HTTPS.

Figure 2 shows all the exchanged top-level parameters with their original names, as well as the most important elements of the token *wresult*. Square brackets include optional parameters. At the address *URIS*, the identity supplier S expects WSFPI redirects. The parameter *wa* is a constant denoting

the so-called protocol action and version. The parameter *wtrealm* is the security realm of C under which it executes the WSFPI protocol; it should equal a fixed address *URIC* of C. With *wreply*, C may specify a URI to which S should redirect the browser in this specific protocol run; it must be within *wtrealm*. Furthermore, C may transport information through the WSFPI protocol using the opaque context parameter *wctx*, and add a timestamp *wct*. The identity supplier S selects the return address a' as *wreply* if present, else as *wtrealm*. The token *wresult* is syntactically a signed SAML assertion; the Security Assertion Markup Language (SAML) is an underlying standard for flexible statements about identities [34]. In our abstract representation $name_{\mathsf{S}}$ is the name under which the identity supplier signs. In the assertion it includes its own address *URIS* and the address *URIC* of the intended identity consumer, the user identity id_{U} as derived in Step 5.1, and additional attributes *att* about this user if desired. In our car-rental example, the user identity might be an employee number, the most important attribute is the fact that the user works for the employer, and other attributes are the name and address.

3 Properties of Browsers and Protocol Variants

Scientifically, WSFPI is a 3-party authentication protocol, but instead of a real protocol machine representing the user there is only a standard browser, and the user must even take one step in person. This distinguishes browser-based FIM protocols from all prior protocols.

The main capabilities that enable a browser to participate in a 3-party protocol at all are redirects and form posting. (Cross-domain cookies, here for the identity supplier and identity consumer, are not considered because they are a too privacy-unfriendly concept.) Redirects can only carry a limited amount of information as URI lengths are limited in the network, while using form posting is not strictly zero-footprint because it typically relies on scripting; otherwise the user sees a rather strange form to submit. Hence protocols with both variants exist. To enable long tokens (e.g., digitally signed ones with credentials) while being strictly zero-footprint, some protocols first use a redirect to exchange a short one-time pseudonym of the user (instead of Steps 6-7 of WSFPI) and then use a *backchannel*, i.e., a direct channel between S and C, to transport the actual token while referencing the one-time pseudonym, e.g., the SAML browser/artifact profile [34].

An advantage of browsers is that they can execute HTTPS and are preconfigured with trust roots for server credentials, i.e., scientifically they can set up secure channels with one-sided authentication. All browser-based FIM protocols make essential use of this capability.

Besides the functional restrictions, a problem with browsers is that they produce additional information flow of protocol variables, in contrast to the usual assumption that participants in security protocols carry out precisely

the defined protocol and do nothing else. For instance, the browser divulges the previous URL it visited in the HTTP referrer tag if the new page is reached by the selection of a link on the previous page. Furthermore local storage such as the cache and the history may be exploited, e.g., by a later user in a kiosk scenario.

A core protocol like WSFPI will usually be augmented by an initial interaction between the identity consumer C and the user U to agree on a suitable identity provider, and by a detailed request by C within Step 4 describing the desired attributes. For instance, SAML also defines request message formats. Furthermore, for privacy there may be constraints on how the identity supplier answers requests, e.g., whether it sends a real identity id_{U}, a previously used pseudonym, or a fresh pseudonym.

The first browser-based protocol was Microsoft's Passport [33]; however, its details are not public. The SAML standardization produced protocols with form posting as well as with backchannels [34]. The Liberty Alliance project extended these protocols by further parameters and variants [31], which now seem to converge back into SAML 2 [35]. The Shibboleth project for university identity federation can be seen as a more complex SAML profile [14]. WS-Federation [28] is part of the IBM/Microsoft web services security roadmap. It links the web services world and the browser world by defining a joint identity-federation basis for both client types. Special aspects for browser-based FIM are defined as a Passive Requestor Profile [29].

4 Security of Browser-based FIM

The security of browser-based FIM has operational and protocol aspects.

Operationally, browser-based protocols belong to a rather risky class, but this is the price for the zero-footprint property, and one has to keep in mind that no other protocols for cross-enterprise 3-party authentication or the exchange of certified attributes are widely deployed, except for the server authentication in SSL. The user has to trust his or her browser and operating system. If the user authentication (as in Step 5.1 of WSFPI) is done by username and password, the security relies on the quality of this password and the user's resistance to phishing. Compared with using individual passwords with all services, a user of browser-based FIM needs fewer passwords and can thus choose them more carefully, remember them better, and be less likely to use the same password with partners of very different trustworthiness. On the other hand, the user may get used to being often redirected to his or her identity supplier S and get careless in using the browser capabilities to verify that he or she really has an HTTPS connection to S before inputting the password. A dangerous feature in Passport and the Liberty proposals is "inline single signon", where the identity supplier uses a part of the identity consumer's window, because it disables these verification capabilities. Operational issues were first analyzed in detail in [30].

While all standards and standards proposals come with security considerations, several problems were later found (and then removed) [40, 22]. This is not surprising as the design of cryptographic protocols is notoriously error-prone, and browser-based FIM protocols are rather complex, e.g., because of the modularity of the standards. One attack from [22] is particularly interesting as it exploits the HTTP referrer tag, i.e., information added automatically by the browser that the protocol designers did not take into account. This shows that one cannot analyze a browser-based FIM protocol by itself but needs to consider it together with features of the browsers.

We have recently defined a detailed *browser model* that can be used as a basis for security proofs of browser-based FIM protocols, and have proved that WSFPI establishes authentic secure channels based on this model [24, 25] (extending a more abstracted proof without a complete browser model [23]). Very roughly, the security holds by the following chain of arguments: The identity consumer C only accepts tokens signed by the identity supplier S. The identity supplier S only signs messages of the token format, assuming it only does so in WSFPI, if it has authenticated the corresponding user over a secure channel. Then S sends the token in a form POST over the same secure channel. The interesting part is now to show that the adversary cannot get the token. Here first the details of the choice of the address a' play a role, and secondly the use of *URIC* in the token to guarantee that one identity consumer, if dishonest, cannot reuse a token to impersonate the user at another identity consumer.

5 Privacy of Browser-based FIM

Privacy, when broken down to a FIM protocol, essentially means that no information about a user should be transferred between the identity supplier and the identity consumer or stored there unless either the user has consented to this, or another policy such as an employment contract or law enforcement issue permits it. We distinguish the privacy issues of the explicit attributes such as *att* in WSFPI, and those of protocol-intrinsic information.

For the attributes, all general-purpose solutions must incorporate a *real-time release* step, i.e., after receiving the request from an identity consumer C and authenticating the user U, the identity supplier S asks the user whether it is OK to release the desired attributes. In some business-to-business scenarios, this step can be skipped. One may work towards enabling the user U to set policies in scenarios where U acts as an individual; however, initially this is not easy. For instance, even if the user wanted to permit S to release his or her address to every airline, there is no general attribute credential structure in place by which S would recognize all airlines. Furthermore, the user might actually only desire the release when buying a ticket at an airline, not when just looking at offers, but the identity supplier cannot distinguish this reliably. Another issue, not yet really covered by standards proposals, is

that the release of attributes might depend on privacy promises from C or come with usage restrictions from S for C (such as "you may use this address for this transaction and your own marketing but not share it with further parties"). At the moment this must be fixed in a-priori bilateral contracts between identity suppliers and identity consumers.

The main protocol-intrinsic parameters that may violate privacy are user identities such as id_{U} in WSFPI and URIs that might allow user tracking. As to identities it is easiest to put at most a one-time pseudonym into specific places in the protocols (such as in the redirect URI in protocols with backchannel) and to treat everything else as attributes that are put under policies. Indeed, no proposal after Microsoft Passport prescribed a global identity again; however, some have partner-specific long-term pseudonyms as a standard, which is not completely flexible. As to user tracking the main question is how much the identity supplier S learns about the browsing behavior of the user U. It seems unavoidable that S learns the identity of C in each execution of a browser-based protocol, both for the final redirect or form POST (Steps 6-7 in WSFPI) and because the identity of C must be present in the token to prevent man-in-the-middle attacks (such as *URIC* in *wresult* in WSFPI). However, some standards or proposals recommend that one of the parameters in the request (corresponding to Step 4 in WSFPI) is the entire URI of the resource that U requested; this seems an unjustified information release in general. Furthermore, services should have consent from U or another permission before even starting a browser-based FIM protocol because of the user tracking that it enables at S. For instance, our example user might not want her employer to be asked by the car-rental company also when she is planning a private trip, and even less would she desire a mechanism that would cause her employer to be notified whenever she is doing some web shopping.

Another issue is that individuals may not want to share uncertified attributes, e.g., their taste in cars when on private trips, with any identity supplier (and even less with a fixed one, such as an employer). Hence if browser-based FIM becomes prevalent, it becomes important that the protocols can also be executed with *local wallets*. This means that users can store attributes locally instead of remotely if they choose. (These users no longer have the zero-footprint property, but it is their choice.) This seems easy in principle because one can build the local wallets as HTTP servers just like identity suppliers. However, it is not well supported in all standards and proposals for the case that the user acts under a pseudonym. One problem is requirements on the naming and credentials of the token (corresponding to $name_{\mathsf{S}}$ in *wresult* in WSFPI) although this could be the local wallet for uncertified attributes; another is specific address usage in backchannels – here the identity supplier should address the identity consumer so that an anonymous local wallet can play the identity-supplier role.

A detailed study of the consequences of privacy for browser-based FIM protocols can be found in [39], and specific considerations that show how

complicated it can be to align a design with the promises made and the types of consent that the user is asked for in [37, 38].

If all these points are taken care of, the privacy of browser-based FIM can actually be very good. One limit, as already mentioned, is that the identity supplier learns the trail of identity consumers a user visits. Another is that for certified attributes, even if the protocol allows local wallets and pseudonyms in principle, the identity supplier that certifies the attribute and the identity consumer can always link these two transactions, because the browser redirects or posts the token to the identity consumer exactly as the identity supplier sent it. This limit can only be overcome by private credentials as described in the following sections.

6 Private Credentials

In this section we are concerned with achieving optimum privacy for users in electronic transactions. That is, our goal is to enable the user to reveal as little information in transactions as possible. Of course, in most transactions, a user needs to reveal some information for it to take place. So our goal will be that the user need not reveal any further information than necessary to conduct a specific transaction. In particular, compared with the previous sections, we even desire that an identity supplier who certified certain attributes does not learn to which identity consumers the user shows these attributes, and that the two parties cannot link these transactions.

Let us illustrate this in the car-rental example. Now we assume that the user wants to rent a car privately, and thus there is no other party (such as the employer above) that should naturally be informed of this rental. However, the user still has to show some certified attributes, such as a driver's license, and to pay for the rental (digital cash is just another form of a credential - one that can be used only once), see Figure 3. Finally, the user is given a certificate from the rental agency that she can use as key to the car, i.e., to show the car that she has the right to drive it for the rented period. Today, she would show her paper or plastic license and the car-rental company would inspect it and learn her name, address, exact age, etc. However, it would be sufficient if they saw the user's picture, to verify that the license was issued to the individual renting the car, and possibly the expiration date, to verify that the license is still valid. (We will discuss anonymity revocation via a trustee in the case of problems later.) The agency that issued the license does not learn about the rental, and indeed this would seem unjustified with respect to privacy. However, the agency could learn this if it were corrupted and the car-rental company collaborated; we might want to prevent this. If we used a browser-based FIM protocol, we could make the driver's-license agency the identity supplier and transfer only the necessary attributes; then, however, this agency would always learn about all rentals. It would also be conceivable to store the license as a fully signed credential in a local wallet,

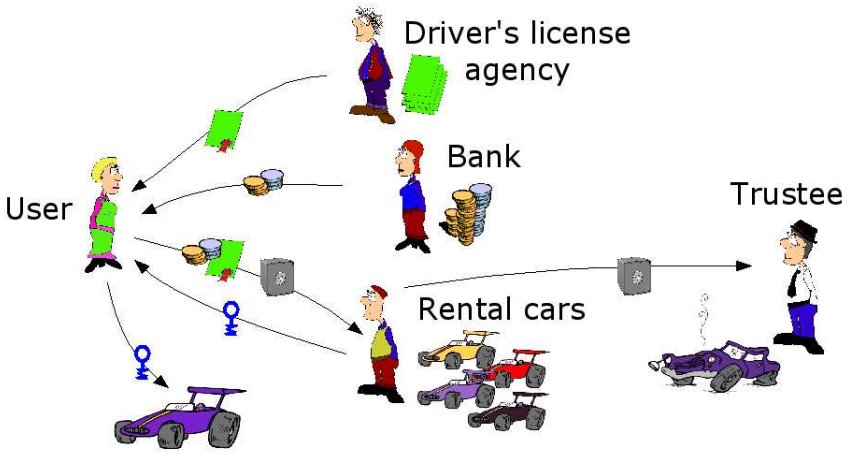

Fig. 3. Example scenario for private credentials

possibly in different versions that contain only pairs of attributes, but still the agency, if corrupted and collaborating with the car-rental company, could find out exactly which user rented which car (by matching the binary strings representing the credential). Regarding the payment, the bank and the car rental agency will need to communicate in order for the rental agency to receive the payment into its account. However, also here we want the user's privacy protected, i.e., the bank shall not be able to learn (even when colluding with the rental agency – assuming that the rental agency is not privy of the user's identity with the bank) where a user spends her money.

With private credentials we can achieve all the properties that were mentioned as desirable above. Such credentials are very similar to a traditional public key infrastructure: a credential is basically a signature by the identity provider on a user's public key and on attributes of the user. The core differences to achieve privacy are that

1. the user uses a *different* public key (pseudonym) with each organization and,
2. instead of the credential (certificate) as obtained from the signer, the user sends the verifier a transformed credential. This transformation is either achieved by using a so-called *blind signature scheme* when issuing the credentials, or by applying so-called *zero-knowledge proofs* when producing credentials to the verifier.

However, for credentials or tokens such as in SAML, Kerberos or X.509 format signed with traditional signature schemes such as RSA or DSA, this would not be practical as we would need to employ general-purpose zero-knowledge proofs that are not efficient. Luckily, there exist special signature schemes

that can be used for issuing credentials such that we can employ particular zero-knowledge proofs that are efficient [10, 11].

In the remainder of this chapter we will first explain the procedures of a private credential system, then describe the particular zero-knowledge proofs protocols and the special signature schemes and finally explain on a high level how one can actually build a private credential system with them.

Let us first, however, discuss the properties such a private credential system must provide. To this end, consider the following variation of our car rental scenario. Assume that our user wanted to rent a racing car and as these are somewhat more dangerous than ordinary cars, the car rental agency require that the user be insured. Thus, when renting such a car, the user not only needs to show that she possesses a driver's license but also that she has insurance. The user wants to do this of course without revealing any further information using a private credential system. Now, if one would implement a private credential system exactly as hinted above, i.e., the user would prove with a zero-knowledge protocol that she possess a driver's license and an insurance policy. Such a solution, however, would not guarantee that the driver's license and the insurance policy were issued to the same person. Indeed, one person could prove possession of a driver's license and then a second person could prove possession of an insurance policy, which is most probably not what the car-rental agency had in mind. Thus a requirement to a private credential system is that it be consistent, i.e., that users can not pool their credentials. In summary, a private credential system should enjoy the following properties [9].

Unforgeability. It must not be possible that a user can prove possession of a credential that was not issued to her.

Privacy. Different transactions by the same user with the same or different organizations must not be linkable (even if the user uses the same credential in some of these transactions), unless the user voluntarily uses the same public key (pseudonym) with these organizations or releases attribute information that by their uniqueness link transactions.

Consistency. Different users must not be able to team-up and use each others' credentials.

Limited-use vs. Multi-use Credentials. Unless otherwise specified, a user should be able to use a credential as many times as she wishes (without the respective transactions becoming linkable). However, she shall only be able to use so-called limited-use credentials as many times as specified. For instance an e-cash credential should be useable only once.

Apart from these basic properties, a credential system should allow for attributes to be included in credentials, e.g., name and address in a driver's license, and for selectively revealing these attributes. Indeed, in some cases the user might not want to reveal the attribute itself, but only certain properties about an attribute. For instance, to rent a car, a user might have to prove that she possesses the driver's license since more than five years and that she is older than 25. Naturally, we want to enable the user to do that without

revealing her age or the date she obtained the driver's license. Also, some scenarios might require conditional anonymity. Consider our car rental example where we assumed that the car rental agency gets not to know the user's name but need to see her picture only for verification purposes. However, in cases the user damages the car, the rental agency probably needs to know the user's name and address. That is, under some well specified conditions (e.g., the user damaging the car) additional information about the user is required. Thus the system needs to allow for the specification of what (additional) information is available to whom under what circumstances. This might require the involvement of a third party, e.g., a trustee, who will make available this information upon checking the circumstances.

6.1 A Historical Notes

As many other privacy-enabling protocol, Chaum put forth the principles of anonymous (or private) credentials [15, 16, 17]. Later, Damgård gave the first proof of concept [19] of an anonymous credential where a credential was represented by a signature on an individual's name, obtained in a way that kept the name hidden from the issuer; while showing a credential was carried out via a general purpose zero-knowledge proof of knowledge. Due to the practical inefficiency of these zero-knowledge proof, this first solution was rather of theoretical interest.

The first step towards efficient systems was Brands e-cash schemes [4] and protocols to issue signature on hidden message [4]. Brands later put these building blocks together to build a private credential system [5]. The first private credential system meeting the major properties as described above was introduced by Lysyanskaya et al. [32]. The first truly efficient and provably secure scheme was put forth by Camenisch and Lysyanskaya [9], whose construction was largely inspired by the Ateniese et al. group signature scheme construction. Their system allowed users for the first time to use a credential more than once. Their system was subsequently refined and extended [8, 11].

Today, the probably most prominent real application is the Direct Anonymous Attestation protocol employed by the Trusted Computing group to authenticate a trustworthy computing platform while retaining the user's privacy [6]. Finally, the PRIME project (Privacy and Identity Management for Europe `www.prime-project.eu.org`) is currently building a holistic framework for privacy-enhancing digital transactions based on private credential systems.

6.2 Building Blocks

A private credential system is typically built with a number of cryptographic primitives. First, a public key signature scheme is used for signing credentials. That is, a credential is basically a signature on a message or a set of messages. A message might encode an attribute of the user or some parameter required

for the system to work. Second, a public key encryption scheme with labels is required, e.g., [12]. Such an encryption scheme allows one to bind a public label to an encryption in a way such that decrypting a ciphertext under any different label reveals no information about the original cleartext. Third, a commitment scheme is required. Such a scheme allows a sender to compute a commitment to some value and then send this commitment to a party without revealing the value itself. Commitment schemes will come handy when a user needs to prove to someone a property of an attribute without revealing the attribute itself. For instance, the user could send the car-rental company a commitment to her birth date and then prove she is older than 21 if that is needed to rent a car, i.e., she proves the committed value encodes a date that lies more than 21 years in the past and that the committed value matches the birth date attribute in her driver's license. Finally, zero-knowledge proof protocols are required as sub-protocols to issue credentials, to show credentials, and to prove properties about committed values.

6.3 Procedures of a Private Credential System

We now describe how a private credential system works. We distinguish the roles of users, organizations (the identity consumers and identity suppliers from before), and trustees, who serve purposes such as anonymity revocation. A party might assume several roles. The private credential system consists of algorithms Setup, OrgKeyGen, TruKeyGen, UserKeyGen, NymGen, IssCert, ShowCert, and RevealAttr.

The first algorithm, Setup, generates the system parameters such as the algebraic groups to be used by all participants.

The algorithms OrgKeyGen and TruKeyGen take as input the system parameters and generate public and private keys for an organization and a trustee, respectively. In the idemix implementation we are going to describe later, the output of OrgKeyGen is essentially a key pair of a (suitable) signature scheme, and the output of TruKeyGen is a key pair of a (suitable) labelled encryption scheme.

The user has two key generation algorithms: The algorithm UserKeyGen generates a secret key for the user and the algorithm NymGen takes as input the user's secret key and generates a new pseudonym based on this secret key. If the user runs NymGen a second time with the *same* secret key, she gets a new pseudonym that is unlinkable to the first pseudonym.

The next algorithm, IssCert, is a protocol between the user and the organization from which the user would like a credential issued, i.e., the organization acts as an identity supplier. A prerequisite of this protocol is that the user has sent the organization one of her pseudonyms generated by NymGen. This could be a freshly generated pseudonym or one that the user has used with that organization (or even other organizations) before, e.g., under which she took driving lessons before getting the driver's license. The user and the organization also have to agree on the of attributes that the credential shall

contain. The typical case is that the attributes are input in cleartext by both parties, e.g., the type of driver's license issued or the user's birthdate. However, there is also the option to input an attribute only via a commitment (about which the user has typically proved something in zero-knowledge) or a random attribute that will get known only to the user at the end of the protocol. These features are required when issuing credentials that should contain as attributes the same attributes as some credentials that the user has shown to the issuer and that should not be known to the issuer. For instance this allows one to bind a credential to a TPM [7]. Another use example of these features is the realization of an anonymous e-cash scheme from these basic protocols.

The algorithm ShowCert is a protocol between the user U and a verifying organization V (corresponding to an identity consumer in the earlier notation). The user holds a credential by some issuing organization I that contains a number of attributes, e.g., the driver's license. Prior to the protocol, U and V agree on which attributes shall be revealed to V, e.g., the driver's license type and the user's picture. They can also agree that U only proves properties about some of the attributes, e.g., that the age is larger than 21; then U additionally provides V with commitments to these attributes. Furthermore, they can agree that V obtains some attributes encrypted under the public key of one or more trustees, e.g., the actual name and address of the user. This is indicated by the locked box in Figure 3. In that case, they also agree on a *decryption policy*, i.e., the conditions under which an encrypted attribute shall be retrieved by the trustee and what the trustee should do with it. For instance, the decryption policy for name and address in the car-rental example could be that the user has had an accident or has disappeared with the car, and that if the car-rental agency claimed the latter, the trustee after decryption first contacts the user and validates that she really did not return the car.

Finally, user U may provide V with a pseudonym in order to prove that the person holding the pseudonym is the same as the one to which the credential was issued. This feature is needed to ensure consistency. Let us expand on this. First, it is often required that the credential was issued to the very user who now shows possession of it; the driver's license is a clear example, while for a concert ticket this is not necessary. Second, if the user proves the possession of several credentials, then all these protocols should be run on input the same pseudonym to ensure that all the credentials were issued to the same person (although they might have been issued to different pseudonyms of this person).

After the two parties have agreed on all these options, they are ready to run the protocol. Their common input to the protocol is the public key of the credential's issuer and, depending on the agreed options, the public key of one or more trustees, commitments to attributes, and a pseudonym of the user. The user U's input additionally comprises secret information, i.e., her secret key, the credential in question and all the attributes contained in it,

and the secret values she used to construct the commitments. If the protocol finishes successfully, party V is convinced that the user possesses a credential by the issuer that contains the attributes that U revealed to him and also those to which he got commitments by U. Furthermore, if that option was chosen, V obtains encryptions of the specified attributes under the specified public keys of trustees with the agreed decryption policies attached as labels. The protocol can be carried out in two modes. A *signature* mode where V will obtain a transcript that will convince any other party that V indeed ran the protocol with a party possessing the respective credential(s) or a *zero-knowledge* mode, where V will not be able to later claim that V ran this protocol.

The last algorithm, RevealAttr, is the procedure for a trustee to reveal attributes that an organization obtained encrypted under this trustee's public key. The input to this algorithm is the trustee's secret key, an encryption, and a decryption policy as well as all the external information that the trustee needs to evaluate the decryption policy, e.g., a proof that the rental car had an accident while this user had rented it. If the external information fulfills the decryption policy, the trustee decrypts the ciphertext with the decryption policy as label and continues with the cleartext as described in the decryption policy.

6.4 Example Use Case: Anonymous Offline E-Cash

Many applications providing privacy can be build on top of a private credential system as sketched above. In this paragraph we will discuss how to realize an anonymous offline e-cash scheme. The involved parties comprise a user, a merchant, and a bank with which both the user and the merchant have an account, i.e., have established a pseudonym with the bank.

An e-coin is basically a credential issued by the bank with special attributes. Thus, to retrieve an e-coin, the user identifies herself to the bank w.r.t. the pseudonym she holds with the bank. Then the two parties run the IssCert such that it will contain as attributes a unique identifier ID of the user (agreed upon jointly by the user and the bank) and a committed attribute s (that the user chose randomly beforehand) and a random attribute b. Recall that the bank does not learn s and b.

For a user to spend an e-coins with a merchant, the two parties proceed as follows. First, the merchant chooses a random integer challenge c. Next, the user computes $u = ID \cdot c + b$ and sends the merchant the values u and s. Finally, the user and the merchant run the ShowCert protocol in signature mode and, in addition, the user also proves to the merchant that 1) s is the second attribute contained in the credential and that the other attributes fulfill the property that u equals the first attribute times c plus the third attribute. If the merchant accepts the protocol, then the payment is done.

At some later point the merchant might want to deposit the obtained e-coin with the bank. To this end, the merchant sends the bank c, u, s and the

transcripts of the ShowCert protocol and the other proof-protocols he executed with the merchant. The bank first verifies the protocol transcript as to see whether (c, s, u) correspond to a valid spending of an e-coin. Next the bank needs to check whether the merchant has already deposited this coin or if the user has spent the coin more than one. That is, the bank checks whether it has seen s before. If not, the bank accepts the coin and puts it the merchant's account. Otherwise, the bank will have record (c', s, u') in it's database. If $c = c'$, the bank reject the coins as the merchant must have deposited the same coin already. If $c \neq c'$, the user must have double spent the coins. In this case the bank also puts the coin in the merchant's account but additionally identifies the user by computing $ID = (u - u')/(c - c')$ and charges the user for the extra spending(s) (plus possibly punishes the user in addition for having misused the system).

Note that if the user does not double spend then, by construction, the values u and s and the transcript do not reveal any identifying information about the user, i.e., the bank will not be able to figure out which user withdrew the coin. Also note that the system is secure for the bank, i.e., 1) no user or merchant can produce money as for each coins put into a merchant's account is taken out of an account of a user and 2) if the user is honest, then her money is taken out of her account only when she withdraws a coin. Finally, the merchant's security is obtained by the zero-knowledge proof protocols, i.e., if the merchant accepts the proof protocol, the bank will accept is as well and credit the merchant account accordingly.

7 Concrete Building Blocks for a Private Credential System

Our goal in the remainder of this chapter it to convey that efficient private credential systems exist and explain how they work. Due to the large number of different option, a full-fledged private credential system can become quite involved. Therefore we are going to concentrate on a private credential system with a reduced set of features, that is, we are not considering (verifiable) encryption of attributes. In this section, we provide concrete instances of primitives to build such a basic private credential system and them put them together in the following section. As we do not consider (verifiable) encryption of attributes, the building blocks we need are an efficient signature scheme, a commitment scheme, and zero-knowledge protocols.

7.1 Protocols to Prove Knowledge of and Relations Among Discrete Logarithms

In the following we will use various protocols to prove knowledge of and relations among discrete logarithms. To describe these protocols, we use notation introduced by Camenisch and Stadler [13]. For instance,

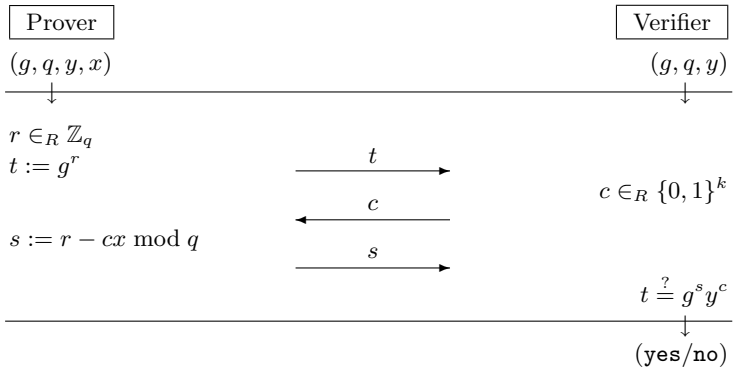

Fig. 4. The protocol denoted $PK\{(\alpha) : y = g^\alpha\}$. The verifier's input is (g, q, y) and the prover's input to the protocol is (g, q, y, x), where the quantity $x = \log_g y$ corresponds to α the knowledge of which the prover is proving. The prover has no output; the verifier's output is either `yes` or `no`, depending on whether or not he accepts the protocol, i.e., whether or not $t = g^s y^c$ holds.

$$PK\{(\alpha, \beta, \gamma) : y = g^\alpha h^\beta \ \wedge \ \tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma \ \wedge \ (v < \alpha < u)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* α*,* β*, and* γ *such that* $y = g^\alpha h^\beta$ *and* $\tilde{y} = \tilde{g}^\alpha \tilde{h}^\gamma$ *holds, with* $v < \alpha < u$," where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. The notation $\langle g \rangle$ means the cyclic group generated by the element g, i.e., all the powers of g. The convention is that Greek letters denote the quantities whose knowledge is proved, while all the other parameters are known to the verifier. More precisely, the property of "proof of knowledge" means that there exists a *knowledge extraction* algorithm that can extract the Greek quantities from a successful prover if given rewinding and reset access to the prover. Thus, using this notation, a proof protocol can be described by just pointing out its aim while hiding the realization details. In the following we first explain how such protocols can be constructed.

7.2 Schnorr's Identification Scheme

The simplest is case is the protocol denoted $PK\{(\alpha) : y = g^\alpha\}$, where $y \in G$ for a group $G = \langle g \rangle$ of prime order q. It is depicted in Figure 4. This protocol is also known as Schnorr's identification protocol [42]. As the first step, the prover chooses a random integer r, computes the *protocol commitment* $t := g^r$ and sends it to the verifier. The verifier replies with a random *protocol challenge* c. Next, the prover computes the *protocol response* $s := r - cx \bmod q$ and sends it to the verifier. Finally, the verifier accepts the protocol if the *verification equation* $t = g^s y^c$ holds.

This protocol is a proof of knowledge of the discrete logarithm $\log_g y$ with a cheating probability (knowledge error) of 2^{-k} (provided that $2^k < q$ which

is typically the case in practice). The protocol is also zero-knowledge against an honest verifier.

To achieve zero-knowledge against an arbitrary verifier, one needs to choose k logarithmic in the security parameter and repeat the protocol sufficiently many times to make the knowledge error small enough, losing some efficiency by this repetition. Reasonable parameters seem to be $k = 10$ and repeating the protocol 8 times to achieve an overall cheating probability of 2^{-80}. Luckily, one can alternatively use one of the many known constructions to achieve zero-knowledge that retain efficiency, e.g., [18]. This discussion holds for all the protocols considered in this chapter.

From the protocol just discussed, one can derive the Schnorr signature scheme denoted $SPK\{(\alpha) : y = g^\alpha\}(m)$, by using the Fiat-Shamir heuristic [20, 41]. Here the the verifier is replaced by a call to a hash function $\mathcal{H}$ and thus the challenge is computed as $c = \mathcal{H}(q\|g\|y\|t\|m)$, where $m \in \{0,1\}^*$ is the message that is signed. The signature of m is the pair (s, c). Verifying a signature entails computing $\hat{t} := g^s y^c$ and then verifying whether $c = \mathcal{H}(g\|y\|\hat{t}\|m)$ holds. This signature scheme is secure in the so-called random oracle model [1].

7.3 Pedersen Commitments

One commitment scheme that is particularly suited for our purposes is the one by Pedersen [36]. Let $G = \langle g \rangle$ be a group of prime order q. Let h be a second generator of G such that $\log_g h$ is not known. To commit to a secret $x \in \mathbb{Z}_q^*$, one chooses a random value $r \in \mathbb{Z}_q^*$ and computes the commitment $C = g^x h^r$.

7.4 Signature Scheme

Camenisch and Lysyanskaya have proposed a number of signature schemes [10, 11] that allow one to efficiently prove possession of a signature with the class of protocols introduced in Section 7.1. The differ in the number theoretic assumption they rely on. In this section we present the one that is based on bilinear maps and relies on the hardness of computing discrete logarithms [11]. It is derived from the group signature scheme due to Boneh, Boyen, and Shacham [2]. It assumes a *bilinear map* setting, i.e., groups $G_1 = \langle g_1 \rangle$ and $G_t = \langle g_t \rangle$, both of prime order q, and an efficiently computable map $e : G_1 \times G_1 \to G_t$. The map e must be bilinear, i.e., it must fulfill

- $e(g^a, h^b) = e(g, h)^{ab}$ for all a, b and for any $g, h \in G_1$, and
- $e(g_1, g_1) \neq 1$.

Such bilinear maps exist for groups G_1 and G_t constructed from elliptic curves. We refer to [3, 21] and references therein for details.

The signer's secret key is a random element $x \in \mathbb{Z}_q$ and the public key consists of $y = g_1^x$ and a number of so-called bases $h_0, \ldots, h_\ell \in G_1$, where ℓ is a parameter.

A signature on messages $m_1, \ldots, m_\ell \in \mathbb{Z}_q$ is a pair (A, s), where $s \in \mathbb{Z}_q^*$ is a value chosen at random by the signer and $A = (g_1 h_1^{m_1} \cdots h_\ell^{m_\ell})^{1/(x+s)}$. A signature (A, s) can be verified by checking whether the single equation

$$e(A, g_1^s y) = e(g_1 h_0^{m_0} \cdots h_\ell^{m_\ell}, g_1)$$

holds.

7.5 Proving Knowledge of a Signature

Now assume that we have a signature (A, s) on messages $m_0 \ldots, m_\ell \in \mathbb{Z}_q$ and want to prove that we indeed possess such a signature. In other words, we need to prove possessions of values $m_1, \ldots, m_\ell$, A, and s such that the verification equation $e(A, y g_1^s) = e(g_1 h_0^{m_0} \cdots h_\ell^{m_\ell}, g_1)$ holds. To do this efficiently, we want to employ the zero-knowledge proof protocols described earlier in this section. So we need to be able to restate the verification equations such that we get a situation such as $y = g^\alpha h^\beta$, i.e., where on the left hand side of the equations all values are known to the verifier and on the right hand side all bases are known to the verifier but all exponents are the prover's secrets the knowledge of which is proved. First note, that we can rewrite the equation as follows $e(A, y)e(A, g_1)^s = e(g_1, g_1)e(h_0, g_1)^{m_0} \cdots e(h_\ell, g_1)^{m_\ell}$ and further as $e(g_1, g_1)/e(A, y) = e(A, g_1)^s e(h_0, g_1)^{-m_0} \cdots e(h_\ell, g_1)^{-m_\ell}$ with which we have almost achieved our goal, except that with A we have a base element that should not be revealed to the verifier. To overcome this, we blind A into $\tilde{A}$ which we can then reveal (basically, we ElGamal-encrypt A where no-one knows the respective secret key). To do so, we need to augment the public key with values $h_t \in G_t$ and $u_1, v_1, w_1 \in G_1$ such that $\log_{g_t} h_t$ and $\log_{g_1} u_1$ are not known to anyone (there are standard procedures for this). This leads to the following protocols.

1. Choose random values $r, r' \in \mathbb{Z}_q$ and compute $\tilde{A} = A u_1^{r+r'}$, $B = v_1^r$ and $C = w_1^{r'}$.
2. Compute the following proof:

$$\begin{aligned} &PK\{(\alpha, \alpha', \sigma, \rho, \mu_0, \ldots, \mu_\ell) : \\ &\qquad B = v_1^\rho \ \wedge\ C = w_1^{\rho'} \ \wedge\ 1 = B^\sigma v_1^{-\alpha} \ \wedge\ 1 = C^\sigma w_1^{-\alpha'} \ \wedge \\ &\frac{e(g_1, g_1)}{e(\tilde{A}, y)} = e(\tilde{A}, g_1)^\sigma e(u_1, y)^{(\rho+\rho')} e(u_1, g_1)^{(\alpha+\alpha')} \prod_{i=1}^{\ell} e(h_i, g_1)^{-\mu_i}\} \ . \end{aligned}$$

Let us explain this proof protocol. The first two statements prove that the prover knows values $\rho = \log_{v_1} B$ and $\rho' = \log_{w_1} C$. The next two statements assert that the prover knows values σ, α, and α' such that $\alpha = \rho\sigma$ and $\alpha' = \rho'\sigma$. The last line asserts the prover's knowledge of further values $\mu_1, \ldots, \mu_\ell$ such that

$$e(\tilde{A}, y)e(\tilde{A}, g_1)^{\sigma} e(u_1, y)^{(\rho+\rho')} e(u_1, g_1)^{\sigma(\rho+\rho')} = e(g_1, g_1) \prod_{i=1}^{\ell} e(h_i, g_1)^{\mu_i}$$

holds, where we made use of the relations $\alpha = \rho\sigma$ and $\alpha' = \rho'\sigma$. Using the fact $e(a, b)e(a, c) = e(a, bc)$ holds for any $a, b, c \in G_1$ (which follows from the bilinearity of the map e) we can reformulate this equation into the following one

$$e(\tilde{A}, yg_1^{\sigma})e(u_1, (yg^{\sigma})^{(\rho+\rho')}) = e(\tilde{A}, g_1^{x+\sigma})e(u_1, g_1^{(x+\sigma)(\rho+\rho')}) = e(g_1 \prod_{i=1}^{\ell} h_i^{\mu_i}, g_1),$$

where $x = \log_{g_1} y$ is the secret key of the signer. Finally, using the property that $e(a^b, c) = e(a, c)^b = e(a^b, c)$ for any $a, c \in G_1$ and $b \in \mathbb{Z}_q^*$ that follows from the bilinearity of e, we have that

$$e(\tilde{A}u_1^{(\rho+\rho')}, g_1^{x+\sigma}) = e(\tilde{A}u_1^{(\rho+\rho')}, yg_1^{\sigma}) = e(g_1 \prod_{i=1}^{\ell} h_i^{\mu_i}, g_1) \ .$$

Comparing this equation with the verification equations, we see that the pair $(\tilde{A}u_1^{(\rho+\rho')}), \sigma)$ must be a valid signature on the messages $\hat{m}_0, \ldots, \hat{m}_\ell$.

8 The idemix Credential System

This section describes how to construct the algorithms of a basic version our concrete private credential system, called idemix [8, 9, 10], from signature scheme discussed in the previous paragraphs.

8.1 Setup – OrgKeyGen and UserKeyGen

We assume all parties agree on a number of parameters. These includes a cryptographic hash function $\mathcal{H}$ such as SHA-256 and on a bilinear maps setting, i.e., on groups groups $G_1 = \langle g_1 \rangle$ and $G_t = \langle g_t \rangle$, and a bilinear map $e : G_1 \times G_1 \to G_t$. Furthermore, let the parties agree on an second generator h_t of G_t such that $log_{g_t} h_t$ is unknown (for instance h_t could be constructed from $\mathcal{H}(g_t)$).

The key generation algorithms for the organizationas and the users are as follows. Each organization (at least if it wants to issue credentials) chooses a secret key x_I and publishes $y_I = g_1^{x_I}$. Each user chooses a random secret key $z \in \mathbb{Z}_q$.

8.2 Generating a Pseudonym – NyMGen

To generate a pseudonym, a user with secret key z chooses a random $r_i \in \mathbb{Z}_q$ and computes $P_i = g_t^z h_t^{r_i}$.

8.3 Issuing a Credential – IssCred

In order to get a credential (on a pseudonym P_i) with attributes $m_2, \ldots, m_\ell$, the user and the issuer perform the following steps.

1. The user chooses a random $r_i' \in \mathbb{Z}_q$, computes $P_i' = h_0^z h_1^{r_i'}$, and sends P_i' to the issuer.
2. The user engages with the issuer as verifier in the following proof:

$$PK\{(\zeta, \rho_i, \rho_i') : P_i = g_t^\zeta h_t^{\rho_i} \ \wedge \ P_i' = h_0^\zeta h_1^{\rho_i'}\}.$$

   This convinces the issuer that the pseudonyms P_i' and P_i encode the same secret key.
3. The issuer chooses a random $s_i \in \mathbb{Z}_q^*$, computes $A_i = (g_1 P_i' \prod_{i=2}^{\ell} h_i^{m_i})^{\frac{1}{x+s_i}}$, and sends (A_i, s_i) to the user.

After this protocol, the user possesses a signature on the tuple of messages $(z, m_1, m_2, \ldots, m_\ell)$, where z and m_1 are "secret messages," i.e., known to the user only.

If the issuer follows the protocol, then it is ensured that the 0-th message signed corresponds to the user's secret key contained in the pseudonym P_i. Moreover, the protocol ensures the issuer that the user that ran the protocol is indeed the holder of the pseudonym P_i as the proof in Step 2 can be run successfully by the user only if she is privy to the secrets encoded in P_i.

8.4 Proving Possession of a Credential Containing a Given Attribute

Assume that the user with secret key z possesses a signature (credential) (A_i, s_i) with attributes $m_2, \ldots, m_\ell$ and wants to prove the possession of this to a verifier to whom she is known under pseudonym P_j. Also, she wants to convince the verifier that, say, the k-th attribute has a value a. To this end, they proceed as follows:

1. The user chooses random values $r, r' \in \mathbb{Z}_q$ and computes $\tilde{A}_i = A_i u_1^{r+r'}$, $B = v_1^r$ and $C = w_1^{r'}$.
2. The user and the verifier engage in the following proof:

$$\begin{aligned} &PK\{(\alpha, \alpha', \sigma, \rho, \rho', \zeta, \mu_0, \ldots, \mu_\ell) : \\ &P_j = g_t^\zeta w_1^{-\alpha'} \ \wedge \ B = v_1^\rho \ \wedge \ C = w_1^{\rho'} \ \wedge \ 1 = B^\sigma v_1^{-\alpha} \ \wedge \ 1 = C^\sigma w_1^{-\alpha'} \ \wedge \\ &\qquad \frac{e(g_1, g_1) e(h_k, g_1)^a}{e(\tilde{A}_i, y)} = e(\tilde{A}_i, g_1)^\sigma e(u_1, y)^{-(\rho+\rho')} \ \cdot \\ &\qquad\qquad \cdot \ e(u_1, g_1)^{-(\alpha+\alpha')} e(h_0, g_1)^{-\zeta} \prod_{i=1, i\neq k}^{\ell} e(h_i, g_1)^{-\mu_i}\}. \end{aligned}$$

This is essentially the same proof as that in Section 7.5 for just showing the possession of a signature, but it additionally shows that the 0-th message signed is the secret key embedded in P_j and that the k-th message signed is a.

It is not hard to adapt the protocol just described to one where several attributes are revealed or where statements about the attributes are proved. An example of such a statement is the assertion that a signed message (e.g., encoding the user's age) is larger than 18.

9 Conclusion

Federated identity management (FIM) means authentication and attribute exchange for users across different interaction partners. The main challenges that have prevented the widespread deployment of earlier FIM proposals such as Kerberos or PKIs are ease of use, in particular ease of initial setup, and privacy. We have described two modern approaches that address these challenges: browser-based FIM, which primarily eases the set-up and usage, and private credentials, which solve the privacy problem that no prior proposal could solve, the unlinkable transfer of certified attributes. Current initiatives at building FIM capabilities into standard clients may ultimately combine the best of both these approaches.

While we concentrated on the protocols, the quality of an overall solution also depends on other factors, in particular the user interface. This starts with the design of real-time release, i.e., forms for the user to consent to authentication and attribute transfer on the fly. The next steps are to enable the user to set meaningful privacy policies, partially at the same time as releasing information, and to keep track of which data were released to whom. For instance, if a user is using several pseudonyms with an ebook shop, he or she needs to keep them apart. Furthermore, to keep some of these pseudonyms really private, the user must not release too much additional information in relation to them. For the private credentials, another aspect that an overall solution must and can offer is the seamless integration with browsers and with simpler types of identity management.

Finally, we note that we only considered part of the information that a user reveals to a communication party. Indeed, the user reveals many other information about her. This starts with the communication protocols typically revealing IP addresses and ends with application/service related information such as preferences, which pages of a news paper a user accesses, etc. While some of this information can be withheld by using technologies such as anonymous communication networks as discussed in another chapter of this book, withholding other information requires that the way applications and services are provided be changed. For instance, service providers often fully identify the user even if they only need to know that she falls in some age group. Another example is the frequent (ab-)use of the social security number as unique

(local) identifier for the user. Thus, protecting the user's privacy in electronic transactions is a non-trivial task that involves many components, areas, and parties. We refer to the PRIME project `http://www.prime-project.eu` for an holistic approach to privacy-enhancing identity management.

10 Acknowledgments

This chapter would not have been possible without a lot of prior work with many coauthors, in particular Endre Bangerter, Thomas Groß, Susan Hohenberger, Anna Lysyanskaya, Dieter Sommer, and Michael Waidner. We thank all of them for the fruitful collaboration.

References

1. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.
2. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *Advances in Cryptology — CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Verlag, 2004.
3. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In X, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume X of *Lecture Notes in Computer Science*, page X. Springer Verlag, 2001.
4. Stefan Brands. Untraceable off-line cash in wallets with observers. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, 1993.
5. Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates—Building in Privacy*. PhD thesis, Eindhoven Institute of Technology, Eindhoven, The Netherlands, 1999.
6. Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proc. 11th ACM Conference on Computer and Communications Security*, pages 225–234. acm press, 2004.
7. Jan Camenisch. *Cryptographic Protocols*, chapter Direct Anonymous Attestation Explained. Wenbo Mao and Markus Jakobsson (Editors). Addison-Wesley, 2006. to appear.
8. Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proc. 9th ACM Conference on Computer and Communications Security*. acm press, 2002.
9. 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 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, 2001.
10. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002*,

volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2003.
11. 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. Springer Verlag, 2004.
12. 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, 2003.
13. Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Burt Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.
14. Scott Cantor and Marlena Erdos. Shibboleth-architecture draft v05, May 2002. `http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-arch-v05.pdf`.
15. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
16. David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.
17. 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 '86*, volume 263 of *Lecture Notes in Computer Science*, pages 118–167. Springer-Verlag, 1987.
18. Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer Verlag, 2000.
19. Ivan Bjerre Damgård. Payment systems and credential mechanism with provable security against abuse by individuals. In Shafi Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 328–335. Springer Verlag, 1990.
20. 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. Springer Verlag, 1987.
21. Steven Galbraith. *Advances in elliptic curve cryptography*, chapter Pairings. Cambridge University Press, 2005.
22. Thomas Groß. Security analysis of the SAML Single Sign-on Browser/Artifact profile. In *Proc. 19th Annual Computer Security Applications Conference*. IEEE Computer Society, December 2003.
23. Thomas Groß and Birgit Pfitzmann. Proving a WS-Federation Passive Requestor profile. In *ACM Workshop on Secure Web Services (SWS)*. ACM Press, to appear, 2004.
24. Thomas Groß, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. Browser model for security analysis of browser-based protocols. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS)*, volume 3679 of *Lecture Notes in Computer Science*, pages 489–508. Springer, 2005.

25. Thomas Groß, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. Proving a WS-Federation Passive Requestor profile with a browser model. In *ACM Workshop on Secure Web Services (SWS)*, pages 54–64. ACM Press, 2005.
26. Matt Hur, Ryan D. Johnson, Ari Medvinsky, Yordan Rouskov, Jeff Spellman, Shane Weeden, and Anthony Nadalin. Passive Requestor Federation Interop Scenario, Version 0.4, February 2004. `ftp://www6.software.ibm.com/software/developer/library/ws-fpscenario2.doc`.
27. Harris Interactive. First major post-9/11 privacy survey finds consumers demanding companies do more to protect privacy. Rochester, `http://www.harrisinteractive.com/news/allnewsbydate.asp?NewsID=429`, February 2002.
28. Chris Kaler and Anthony Nadalin (ed.). Web Services Federation Language (WS-Federation), Version 1.0, July 2003. BEA and IBM and Microsoft and RSA Security and VeriSign, `http://www-106.ibm.com/developerworks/webservices/library/ws-fed/`.
29. Chris Kaler and Anthony Nadalin (ed.). WS-Federation: Passive Requestor Profile, Version 1.0, July 2003. BEA and IBM and Microsoft and RSA Security and VeriSign, `http://www-106.ibm.com/developerworks/library/ws-fedpass/`.
30. David P. Kormann and Aviel D. Rubin. Risks of the Passport single signon protocol. *Computer Networks*, 33:51–58, 1994.
31. Liberty Alliance Project. Liberty Phase 2 final specifications, November 2003. `http://www.projectliberty.org/`.
32. Anna Lysyanskaya, Ron Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard Heys and Carlisle Adams, editors, *Selected Areas in Cryptography*, volume 1758 of *Lecture Notes in Computer Science*. Springer Verlag, 1999.
33. Microsoft Corporation. .NET Passport documentation, in particular Technical Overview, and SDK 2.1 Documentation (started 1999), September 2001.
34. OASIS Standard. Security assertion markup language (SAML) V1.1, November 2002.
35. OASIS Standard. Security assertion markup language (SAML) V2.0, March 2005.
36. Torben Pryds 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. Springer Verlag, 1992.
37. Birgit Pfitzmann. Privacy in enterprise identity federation – policies for Liberty single signon. In *Proc. 3rd International Workshop on Privacy Enhancing Technologies (PET)*, volume 2760 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2003.
38. Birgit Pfitzmann. Privacy in enterprise identity federation - policies for Liberty 2 single signon. *Elsevier Information Security Technical Report (ISTR)*, 9(1):45–58, 2004. `http://www.sciencedirect.com/science/journal/13634127`.
39. Birgit Pfitzmann and Michael Waidner. Privacy in browser-based attribute exchange. In *Proc. 1st ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 52–62, 2002.
40. Birgit Pfitzmann and Michael Waidner. Analysis of Liberty single-signon with enabled clients. *IEEE Internet Computing*, 7(6):38–44, 2003.

41. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer Verlag, 1996.
42. Claus P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
43. Alan Westin. Consumer privacy attitudes and actions: What the surveys find 2005-2006. Privacy Year in Review, Projections and Trends for 2006, Privacy & American Business, January 2006.