A client-side approach for privacy-preserving identity federation

Sébastien Canard · Eric Malville · Jacques Traoré

Received: 1 April 2009 / Accepted: 26 July 2009 / Published online: 1 October 2009
© Identity Journal Limited 2009

Abstract Providing Single Sign-On (SSO) between service providers and enabling service providers to share user personal attributes are critical for both users to benefit from a seamless access to their services, and service providers to realize new business opportunities. Today, however, the users have several independent, partial identities spread over different service providers. Providing SSO and attribute sharing requires that links (federations) are established between (partial) identities. In SAML 2.0 (Maler et al. 2003), the links between identities are stored and managed at the network side by the identity providers (network-side identity federation). This model prevents the service providers from mass-correlating the partial identities they have, but the users must fully trust the identity providers. In this paper, we propose a complementary approach where the users have a full control of the links between their partial identities. It is a client-side identity federation approach, which relies on the introduction of a new cryptographic tool, called invariable partially blind signature scheme, that may be of independent interest.

Keywords Cryptography · Identity federation · Identity management · Privacy · Blind signatures · User centric

S. Canard (✉) · E. Malville · J. Traoré
Orange Labs, 42 rue des Coutures, BP6243, 14066 Caen Cedex, France
e-mail: sebastien.canard@orange-ftgroup.com

E. Malville
e-mail: eric.malville@orange-ftgroup.com

J. Traoré
e-mail: jacques.traore@orange-ftgroup.com

Introduction

As Internet is a prime vehicle for business and personal interactions, more and more organizations, today, provide their customers with personalized on line services. This requires that the users have, at each service provider (SP), an identity associated to credentials for authentication and attributes (personal information) for personalization. The users have, therefore, several independent, partial identities spread over SPs. The relative independence of these partial identities enables the user to control, to some degree, the exposure of personal information complying this way with a certain level of privacy and, in particular, with the collection limitation privacy principle stating that there should be limits to the collection of personal data and that any such data should be obtained by lawful and fair means (Liberty Alliance <http://www.projectliberty.org/>).

Providing Single Sign-On (SSO) between SPs and enabling SPs to share user personal attributes are critical for both users to benefit from a seamless access to their services, and SPs to realize new business opportunities. This requires that federation links are established between (partial) identities. The very nature of these links, and the way they are established and managed are critical with respect to privacy.

In what we call the network-side identity federation model in this paper, the identity federation links are managed by identity providers (IdPs) in the network and the authentication of the user is performed by the IdPs. The identity federation links are used by the IdPs in authentication claims or assertions to reference the user in the namespace of the SPs.

The identities of a user can be federated using a unique identifier shared by the IdPs and all the SPs. In the OpenID 1.x (OpenID <http://openid.net/>) approach, for instance, the users have one or more OpenID identifiers (URI) at one or more IdPs. These identifiers are presented by the users to the SPs and enable the SPs to determine which IdP the authentication must be delegated to. Since the users have to remember their OpenID URI, they usually create only few OpenID identifiers. This brings to a situation where a same OpenID identifier is used to access several SPs. Should these SPs be malicious, they could collude to mass-correlate the users' personal attributes each of them have and compromise this way the collection limitation privacy principle.¹ In this model, the users must fully trust both the IdP and the SPs.

The identity federations can also be established in a more privacy-friendly way. In SAML 2.0 (Maler et al. 2003) for instance, IdPs use pseudonyms

¹The version 2.0 of OpenID enables the users to provide the URL of the IdP instead of their own OpenID URI. Therefore, disclosure of the OpenID URI is not mandatory anymore but the specifications do not give any recommendation on the way an IdP should reference a user in the namespace of the SPs.

or aliases (nameID) to reference the users in the SPs' namespace and these pseudonyms differ from one SP to another. This means that the SPs cannot directly reference this user in the namespace of each other, preventing, this way, malicious SPs from colluding to mass-correlate users' identities. However, in this approach, federation aliases are still stored and managed by the IdPs in the network. This requires that the users put sufficient trust in the IdPs (and the network) to entrust them with the management of the federation links, since this information would enable the IdPs to correlate the partial identities the users have at the SPs.

The identity selector, like CardSpace (<http://netfx3.com/content/windowscardspacehome.aspx>) from Microsoft or Higgins, is a new concept that comes as a software component installed and running on the users' devices. It aims to help users better understand and represent their digital identities and, therefore, to simplify the management and the use of digital identities. The identity selectors support the usual network-side identity federation model. But they also introduce a new model, that we call a client-centric identity model, where the IdPs functions are all transferred to the user's device; Identity federations and authentications are managed and performed at the client side. This approach gives the users a full control on the federation links and enable them to keep their identities separate. But, at the same time, it requires that SPs in the network put sufficient trust in the user's device to accept authentication claims provided by the IdP hosted on the device.

In this paper we propose an intermediary model between the network-side identity federation model where all the IdP's functions are all performed in the network and the client-centric identity approach where these functions are all performed in the user's device. This model is an evolution of the SAML 2.0 identity federation model based on a new partially blind signature scheme. In this model, only the management of the identity federation links are transferred to the user's device and the actual authentication of the user is performed by IdPs in the network. This model gives the users a full control on the identity federation links while preserving the trust relationships established between SPs and IdPs that enable SPs to accept authentication claims provided by the IdPs.

The Section "[Identity federation: towards a Client-side approach](#)" gives an overview of the client-side identity federation approach we propose. This approach relies on blind signature schemes that have been initially introduced by Chaum (1982, 1983). The existing blind signature schemes must be adapted and the Section "[Invariable partially blind signature schemes](#)" presents the adaptation we propose. Note that our new cryptographic building block, called *invariable partially blind signature*, may be of independent interest. In Section "[Client-side identity federation and SSO protocols](#)", we describe in details the client-side identity federation and single sign-on protocols and we finally give some implementation considerations in Section "[Implementation considerations](#)".

Identity federation: towards a Client-side approach

In this section, we first present the network-side identity federation model proposed in the SAML 2.0 specifications and supported by the identity selectors, and the new client-centric identity model introduced with the identity selector concept. This section also clarifies the positioning of our client-side identity federation model with regard to these existing models and shows why these different models are complementary.

SAML 2.0: a full network-side identity federation model

Identity federations occur between identities at SP and identities at IdP, and is based on the use of aliases (or pseudonyms). In the network-side identity federation approach, the federation aliases are stored and managed by the IdP, and can be, like in SAML 2.0, unique on a per-identity-federation basis across all IdP and SPs (see Fig. 1). Each alias corresponds to a federation link between an identity local to a SP and an identity local to an IdP (*e.g.* the pseudonym 123 is the federation link between the identities *user@SP_A* and *user@IdP*). It is a pseudo-random value (generated either by the IdP or the SP) that has no discernible correspondence with the user's identifiers and that represents the user in the namespaces of the IdP and SP.

The first time a user uses a particular IdP to authenticate to a SP, the user may federate her SP's identity to her IdP's identity. This identity federation is done only once for a given user, a given IdP and a given SP. The general principle of identity federation is the following (see Fig. 2).

Fig. 1 Server-side identity federation

Fig. 2 SAML 2.0 identity federation

1. The user accesses to a SP.
2. SP redirects the user to the IdP with an authentication request *AuthnRequest*. This authentication request contains in particular a request identifier, the date of the request and the identity of the SP.
3. IdP verifies the validity of the authentication request and authenticates the user so as to retrieve her local identity at the IdP: *user@idp*. IdP determines that the user *user@idp* has not yet federated her identity with her identity at the SP. It, therefore, generates a new federation alias *alias* and associates it to the local user identity for the SP. IdP finally generates an authentication response *AuthnResponse* and signs it. This response contains the federation alias *alias* and not the local identity *user@idp* of the user.
4. The IdP redirects the user to the SP with the authentication response *AuthnResponse*.
5. SP verifies the IdP's signature on *AuthnResponse* and extracts *alias*. Since there is no local identity corresponding to *alias*, SP asks the user whether she wants to federate his local identity with her IdP's identity. SP then authenticates the user to retrieve the local identity *user@sp* and associates it with the federation alias *alias*.
6. SP finally provides the user with the requested service. The identity *user@idp* of the user at the IdP is now federated to her identity *user@sp* at the SP with the federation alias *alias*.

Once the identity federation is performed, the user can benefit from Single Sign On (SSO) from the IdP to the SP; the user can authenticate to the IdP and access the SP without any additional authentication. The SSO protocol relies on the same general principle as the identity federation. The only differences reside in the following two steps:

- during step 3, the IdP, after authenticating the user, retrieves the alias corresponding to *user@idp* and associates to the requested SP. As before, the IdP finally generates the authentication response *AuthnResponse* including the alias and signs it;
- during step 5, the SP does not need to authenticate the user. It first extracts the alias from the (valid) authentication request and uses this alias to retrieve the identity of the user *user@sp* from his database.

Identity selector

Based on Kim Cameron's laws of Identity (Cameron 2005), the identity selector is a new concept materialized by a client software (e.g. Microsoft's CardSpace or Higgins) running on the user device, and whose goal is to simplify the management of identities by the user. The basic principle relies on the analogy with the billfold: the identity selector is a receptacle for information cards, each of them representing a user identity. The basic idea consists in symbolizing the act of authentication in order to make it more natural: in order to authenticate to a SP and provide it personal information, the user has just to present an information card.

The identity selector concept supports two identity federation models (see Fig. 3). On the one hand, it supports the traditional network-side identity federation model. The card, called managed card in this case, represents an identity managed by an IdP in the network. Authentications and identity federation links are controlled and managed by this IdP and, therefore, at the network side. On the other hand, the identity selector concept introduces a new model, a client-centric identity model, in which the card (called self-issued card in this model) represents an identity created and managed locally on the user's device. The device plays the role of the IdP and the protection of this identity fully relies on the security mechanisms available and (optionally) activated locally. Authentications and identity federation links are, therefore, managed at the client side.

Discussion on SAML 2.0 and the identity selector concept

SAML 2.0 provides a network-side identity federation model where both the authentications and the identity federations are performed by IdPs at the network side. The Identity Provider gives, for a given user, an alias that is different from one SP to another and this alias is used by the IdPs in the authentication claims it provides to the SPs to reference the user in each SPs' namespace. SAML 2.0 provides a first level of protection of the user's privacy

Fig. 3 Infocard model

since there is no way for two corrupted service providers to collude and mass-correlate the identities stored in their database. In most cases, this identity federation approach can be deemed acceptable from a privacy standpoint and the trust the users have in the IdPs in the network can be sufficient to entrust them with the management of the identity federation links. But in this approach, the federation aliases are stored and managed by the Identity Provider in the network. The IdP has, therefore, all the information necessary to correlate users' identities. The network-side identity federation approach is, therefore, more relevant when the partial identities are slightly independent and when the damage that the correlation of these partial identities would cause is limited. But some identities (health, government, bank...) are so different and so critical that we think the user may refuse to entrust the management of the links between these identities to the IdPs in the network.

The identity selector concept supports the network-side identity federation model and introduces a new client-centric identity model where both the authentication and identity federation functions are transferred from the network to the user's device. This new model proposed with the identity selector concept gives more control to the user since the federation aliases are controlled and managed by an identity provider residing in the user's device. It, therefore, provides an even stronger protection of the user's privacy. However, in this radical approach, authentications are performed at the client side and the SP must put sufficient trust in the device to accept the authentication claims it provides.

The client-side identity federation approach we propose (see Fig. 4) is an intermediary approach between the network-side identity federation model

Fig. 4 Client-side federation

and the client-centric identity one. It enables the user to fully control the federation aliases (preventing, this way, the IdP and the SPs to collude and mass-correlate partial user identities they have) but leave the identity management and the user authentication for IdPs in the network. As in the network-based identity federation model, the SPs can, therefore, establish trust relationships with these IdPs and accept the authentication claims they provide. This model is a complementary approach with regard to the network-side and the client-centric ones since it might be relevant in cases where the users could be reluctant to entrust the management of the identity federation links to IdPs in the network and where the SPs could refuse to trust authentication claims provided by the users' device. This could be typically the case if we want to enable authentication and attribute sharing between domains like health, government, bank... for which identities and services are deemed sensitive.

In the client-side identity federation model we propose, the IdP does not know the aliases used to actually reference the users in the namespace of the SPs. This property relies on the use of a so-called blind signature scheme. This scheme enables a requester to ask a signer to sign a message without revealing all parts of the message. The user can, therefore, ask the IdP to sign an authentication response without revealing the alias referencing the user in the SP's namespace. The assumption, here, is that the IdP does not need to know the actual alias used by the SP. The key requirement is that the identifier contained in the authentication response the IdP signs must be stable for a given user and a given SP (it must always be the same).

On the use of blind signature schemes

A blind signature scheme (Chaum 1982, 1983) is a cryptographic protocol that allows a user to obtain a signature without giving the signer any information

about the actual message. Moreover, even if the signer sees one of the documents he has signed later on, he won't be able to determine when and for whom he has signed it.

But this is not exactly what we need in our context since, as said before, the IdP does not only signs the alias but also the link between the alias, the user and the SP. Moreover, the IdP necessarily knows (i) the user's identity since the role of the IdP is to authenticate users and (ii) the requested SP since the IdP has to retrieve the right alias of the user (corresponding to this particular SP).

Consequently, the property of blind signature schemes is somewhat too restrictive. We need that the resulting signed message includes necessary information known by the signer: this is called a partially blind signature. The message to be signed is, therefore, divided into two parts: the first part M will be known by the signer whereas the second part m will be blinded (that is unknown by the signer). In our context, the blinded part of the message the IdP signs corresponds to the alias and may be some information embedded in the authentication request (such as, again, the identifier of the request) whereas the clear part will be in particular the service provider's identity. But again this is not enough in our context.

If we consider the SSO protocol, the IdP needs to sign the link between the alias, the user and the SP several times (at each new authentication request) but the (blind) alias must always be the same for a given couple (user, SP). Moreover, as mentioned before, the authentication request includes a request identifier, the date of the request and the identifier of the SP. The request identifier or the date of request can be used by the IdP and the SP to cross correlate the user's identities. Thus, this information must not be revealed to the IdP and must be different from one authentication request to another. The message must, therefore, be divided into three parts: the first one is known by the signer, the second one is blinded but needs to be invariable and the final one is a "true" blinded message. More precisely, we need a new type of partially blind signature scheme, that we call invariable partially blind signature scheme. Having such building block, we can use it in our context with the message being divided into the three following parts.

1. The first one corresponds to what the IdP needs to know such as the service provider's identity.
2. The second part differs from one authentication request to another and corresponds to information that could be used by the IdP and SP for cross-correlation (the authentication request identifier or the date of the request).
3. The third part corresponds to the information that is invariable from one authentication request to another. It corresponds to the blind alias. Using invariable partially blind signature schemes, we thus obtain the new federation model shown in Fig. 5.

The aim of the next section is to provide such partially blind signature scheme. In Section "[Client-side identity federation and SSO protocols](#)", we will finally give the way to use it in our client-side identity federation solution.

Fig. 5 Client-side identity federation model

Invariable partially blind signature schemes

In this section, we formally introduce the notion of invariable partially blind signature scheme (IPBS for short) and next give two different ways to obtain such scheme.

Model for invariable partially blind signature schemes

An invariable partially blind signature scheme implies three different actors: a user denoted $\mathcal{U}$, a signer $\mathcal{S}$ and a verifier $\mathcal{V}$. Each of them is implied in one or several of the following procedures:

- **SETUP** is an algorithm executed by *e.g.* the signer which takes on input a security parameter λ and outputs the parameters **param**, the signer's public key **spk** and the corresponding private key **ssk**;
- **SIGN** is a protocol between the user taking on input a message $\text{msg} = M\|m\|m^*$, **param** and **spk** and the signer taking on input M , **ssk**, **param** and **spk**. The user outputs a signature σ on the message msg while the signer outputs its view **viewS** of the signing protocol, which includes the blinded version m_{blind} of m .
- **RESIGN** is a protocol between the user taking on input a message $\text{msg} = M\|m\|m^*$, **param**, **spk** and optionally the blind version m_{blind} of the message m and the signer taking on input M , m_{blind} , **ssk**, **param** and **spk**. The user outputs a signature σ on the message $\text{msg} = M\|m\|m^*$ while the signer outputs its view **viewS** of the re-signing protocol, which includes the blinded version m_{blind} of m .

- **VERIF** is an algorithm executed by the verifier which on input a message msg , a signature σ , param and spk outputs 1 if the signature σ on the message msg is valid and 0 otherwise.

We can now focus on the security properties an invariable partially blind signature scheme should verify. There are mainly four properties and if we consider useless to formally describe the correctness one which simply describe the fact that everything goes well with honest actors, it remains the three following definitions.

- **Unforgeability:** the adversary against the unforgeability property is given the signer's public key spk and param and can successfully interact ℓ_1 (resp. ℓ_2) times with the signer in the **SIGN** (resp. **ReSIGN**) protocol. The adversary wins this unforgeability experiment if she successfully outputs $\ell_1 + \ell_2 + 1$ valid signatures. Thus, an invariable partially blind signature scheme is *unforgeable* if for any polynomial-time adversary, her probability of success is negligible.
- **First partial blindness:** the adversary against the first partial blindness property is given the signer's secret key ssk and param and can interact with two honest users $\mathcal{U}_0$ and $\mathcal{U}_1$. At any time of the experiment, the adversary outputs the following different messages $M, m_0 \| m_0^*$ and $m_1 \| m_1^*$. Then, one bit $b \in \{0, 1\}$ is randomly chosen and the adversary engages in two **SIGN** protocols with $\mathcal{U}_0$ (resp. $\mathcal{U}_1$) taking on input $\text{msg}_b = M \| m_b \| m_b^*$ (resp. $\text{msg}_{\bar{b}} = M \| m_{\bar{b}} \| m_{\bar{b}}^*$), param and spk . Next, the adversary is given the two resulting signatures together with the corresponding message (σ_b, msg_b) and $(\sigma_{\bar{b}}, \text{msg}_{\bar{b}})$. Finally, the adversary outputs one bit $b' \in \{0, 1\}$. Thus, an invariable partially blind signature scheme is *first partially blind* if for any polynomial-time adversary, the probability that $b' = b$ differs significantly from $1/2$ is negligible.
- **Second partial blindness:** the second partial blindness is define as the first one, except that the adversary and the honest users $\mathcal{U}_0$ and $\mathcal{U}_1$ interactively play a **ReSIGN** protocol. More precisely, the adversary is given the signer's secret key ssk and param and can interact with two honest users $\mathcal{U}_0$ and $\mathcal{U}_1$. At any time of the experiment, the adversary outputs the following different messages $M, \tilde{m}_0, m_0^*, \tilde{m}_1$, and m_1^* such that $m_{0\text{blind}}$ and $m_{1\text{blind}}$ are the blind version of the messages m_0 and m_1 that have been obtained by $\mathcal{U}_0$ and $\mathcal{U}_1$ respectively in a previously played **SIGN** protocol (we consider that $\mathcal{U}_0$ and $\mathcal{U}_1$ have access to m_0 and m_1 respectively). Then, one bit $b \in \{0, 1\}$ is randomly chosen and the adversary engages in two **ReSIGN** protocols with $\mathcal{U}_0$ (resp. $\mathcal{U}_1$) taking on input $\text{msg}_b = M \| m_0 \| m_b^*$ (resp. $\text{msg}_{\bar{b}} = M \| m_1 \| m_b^*$), param and spk . Next, the adversary is given the two resulting signatures together with the corresponding message (σ_b, msg_b) and $(\sigma_{\bar{b}}, \text{msg}_{\bar{b}})$. Finally, the adversary outputs one bit $b' \in \{0, 1\}$. Thus, an invariable partially blind signature scheme is *second partially blind* if for any polynomial-time adversary, the probability that $b' = b$ differs significantly from $1/2$ is negligible.

Useful tools for our constructions

The basic cryptographic building blocks that will be needed for the design of our IPBS are commitment schemes, signatures of knowledge and special signature schemes (*a.k.a.* Camenisch-Lysyanskaya signature schemes).

Commitment schemes

A commitment scheme allows a party to commit to a tuple $(x_1, x_2, \dots, x_n)$ of (secret) values to another party. A commitment doesn't reveal any (computational) information on the tuple to the other party (hiding property) and prevents the committing party to change the values being committed to at a later stage (biding property). For example if G is a cyclic group in which the *discrete logarithm problem* is assumed to be hard and $g_1, g_2, \dots, g_n$ are n random generators of G , then $C = \text{COMMIT}(x_1, x_2, \dots, x_n) = g_1^{x_1} g_2^{x_2} \dots g_n^{x_n}$ is a commitment on the tuple $(x_1, x_2, \dots, x_n)$.

Signatures of knowledge and zero-knowledge proofs of knowledge

In a zero-knowledge proof of knowledge a prover convinces a verifier that she knows a witness w such that a predicate P is fulfilled without releasing any further information on w . These interactive proofs can also be used non-interactively (*a.k.a.* *signatures of knowledge*) by using the Fiat-Shamir heuristic (Fiat and Shamir 1986). In the sequel, we will use the following notation $\text{NIZK}[(\alpha, \beta, \dots) : P]$ to denote a signature of knowledge proving that the prover knows a tuple $(\alpha, \beta, \dots)$ of secret values satisfying the predicate P . In this notation, *greek letters* will denote the secret knowledge and the others letters will denote public parameters between the prover and the verifier. For example, $\text{NIZK}[\alpha : h = g^\alpha]$ will denote a non-interactive proof of knowledge of the discrete logarithm of h in the base g (see the Schnorr authentication scheme below).

The Schnorr Authentication Scheme The Schnorr authentication scheme has been introduced by Schnorr in Schnorr (1989). It permits a prover to prove that she knows a discrete logarithm of a public key known by the verifier in a group of prime order.

Let p and q be two primes such that $q|(p-1)$. Let G be a subgroup of $\mathbb{Z}/p\mathbb{Z}$ of order q and let $g \in G$. We suppose that the prover knows the discrete logarithm x of h in base g . The authentication scheme is then described in Fig. 6.

This interactive protocol can be turned non-interactively by using e.g. the Fiat-Shamir heuristic (Fiat and Shamir 1986) that consists in first randomly choosing $w \in [0, q-1]$, computing $a = g^w$, then replacing the random choice of c by the output of a secure hash function $\mathcal{H}$ on input g and a . The final component of the non-interactive authentication is then $r = w - cx \pmod{q}$.

Fig. 6 Schnorr authentication scheme

The verification is done by checking that $c = \mathcal{H}(g\|g^r h^c)$. This non-interactive zero-knowledge proof is in the following denoted as $NIZK(x : h = g^x)$.

The Chaum and Pedersen Variant of the Schnorr Authentication Scheme It exists several variants of the Schnorr authentication scheme in the literature. One is due to Brands (1993) and permits someone to prove that she knows the representation $(x_1, \dots, x_n)$ of a public value h in the base $(g_1, \dots, g_n)$. Again, the non-interactive version of this proof, using the Fiat-Shamir heuristic, is denoted by $NIZK((\alpha_1, \dots, \alpha_n : h = \prod_{i=1}^n g_i^{\alpha_i})$. This proof of knowledge is detailed in Brands (1993).

Another one has been proposed by Chaum and Pedersen (1992). In this latter, we consider that there exists a subgroup G of $\mathbb{Z}/p\mathbb{Z}$, where p is a prime number, of order q where q is a prime such that $q|(p-1)$. Let us also consider g and m two elements of G .

The prover owns a secret key $x \in [1, q-1]$ and computes her public keys $h = g^x$ and $z = m^x$. The aim of the Chaum and Pedersen protocol is to provide a proof of knowledge of a secret that corresponds to both the discrete logarithm of h is base g and the discrete logarithm of z in base m : this is a proof of equality of two known discrete logarithms. This is done by choosing at random $w \in [0, q-1]$, computing $a = g^w$ and $b = m^w$ and finally $r = w - cx \pmod{q}$ after receiving a random value c . (a, b, c, r) is valid iff $a = g^r h^c$ and $b = m^r z^c$ (see details in Fig. 7).

An Ad-hoc Version of the Chaum-Pedersen Scheme It exists many ways to modify the above authentication protocols to transform it into a signature scheme. The one we describe in the following is based on the Fiat-Shamir heuristic (Fiat and Shamir 1986). Let p and q be two primes such that $q|(p-1)$. Let G be a subgroup of $\mathbb{Z}/p\mathbb{Z}$ of order q . Let $g, g_1, g_2, g_3 \in G$. Let $\mathcal{H}$ and $\mathcal{H}_1$ be two secure hash functions.

The signer owns a private key $x \in [0, q-1]$ and publishes the corresponding public key $h = g^x$.

- **Signature algorithm.** Let msg be a message of the form $M\|m\|m^*$ where M and m^* are in $\{0, 1\}^*$ and $m \in G$ such that the signer knows the discrete logarithm of m in base g_2 . The signer acts has follow to sign msg . She first

Fig. 7 Chaum-Pedersen authentication scheme

chooses at random $w \in [1, q - 1]$ and computes $z = (g_1^{\mathcal{H}(M)} \cdot m \cdot g_3)^x$, $c = \mathcal{H}(M \| m^* \| m \| z \| g^w \| (g_1^{\mathcal{H}(M)} m g_3)^w)$, $r = w - cx \pmod{q}$ and finally a non-interactive proof of knowledge P of the discrete logarithm of m in base g_2 using the Schnorr protocol (see above). The signature σ on msg is thus (z, c, r, P) .

- Verification algorithm. The verification of the signature (z, c, r, P) of a message $\text{msg} = M \| m \| m^*$ is done by first verifying that P is valid and then by checking that

$$c = \mathcal{H}(M \| m^* \| m \| z \| g^r h^c \| (g_1^{\mathcal{H}(M)} m g_3)^r z^c).$$

This non-interactive zero-knowledge proof is denoted

$$\text{NIZK}(x : h = g^x \wedge z = m^x).$$

Camenisch-Lysyanskaya signature schemes

These special signature schemes are proposed in Camenisch and Lysyanskaya (2004) with in addition some specific protocols.

One of this protocol allows in particular that a signature be issued on messages that are not known to the signer, but to which the signer only knows a commitment. Informally, in a protocol for signing a committed value, we have a signer with public key SPK , and the corresponding secret key SSK , and a user who queries the signer for a signature. The common input to the protocol is a commitment C on secret values $(x_1, x_2, \dots, x_n)$ only known by the user. In the end of the protocol, the user obtains a valid Camenisch-Lysyanskaya signature $\Sigma = \text{CLSIGN}(x_1, x_2, \dots, x_n)$ and the signer learns nothing about $(x_1, x_2, \dots, x_n)$.

Another protocol allows to prove knowledge of a signature on a tuple of messages $(x_1, x_2, \dots, x_n)$ without releasing any information on the corresponding signature. Each message can either be revealed to the verifier, he can know a commitment of it or have no information on it. It is for example possible to prove knowledge of a Camenisch-Lysyanskaya signature on committed values. Using our notation, such a proof would

be denoted by $NIZK[(\alpha_1, \alpha_2, \dots, \alpha_n, \beta) : C = \text{COMMIT}(\alpha_1, \alpha_2, \dots, \alpha_n) \wedge \beta = \text{CLSIGN}(\alpha_1, \alpha_2, \dots, \alpha_n)]$.

A first construction based on the Chaum-Pedersen signature

Several blind signature schemes exist in the literature (Chaum 1982, 1983; Camenisch et al. 1994; Abe 2001) but, as explained in the above section, we need to provide a new construction, based on the model defined previously. For this purpose, we use a generic way which permits to construct a blind signature scheme by using as a primitive a discrete-log based zero-knowledge proof of knowledge (ZKPK).

Description

We can now introduce our invariable partially blind signature we use to improve the privacy of the identity federation of SAML 2.0 as it will be described in Section “[Client-side identity federation and SSO protocols](#)”. For this purpose, we turn the above Chaum-Pedersen signature scheme into a blind version one where M is known by the signer and the blinded parts of the message are $m = g_2^v$ (invariable data) and m^* (variable data).

- **SETUP:** let p and q be two primes such that $q|(p-1)$. Let G be a subgroup of $\mathbb{Z}/p\mathbb{Z}$ of order q . Let $g, g_1, g_2, g_3 \in G$. Let $\mathcal{H}$ and $\mathcal{H}_1$ be two secure hash functions. The secret key ssk of the signer is an integer $x \in [1, q-1]$ and the corresponding public key spk is $h = g^x$.
- **SIGN:** in a nutshell, a user wanting to obtain a blind signature on a message $\text{msg} = M \| m \| m^*$ has first to blind $m = g_2^v$ by choosing at random $s \in [1, q-1]$ and computing $m_{\text{blind}} = m \cdot g^s$. The signer, on input m_{blind} , computes $m_0 = g_1^{\mathcal{H}_1(M)} m_{\text{blind}} g_3$ and $z_0 = m_0^x$ and proves, using the Chaum and Pedersen signature scheme, that the discrete logarithm of h in base g is the same as the discrete logarithm of z_0 in base m_0 . The user finally retrieves a true Chaum-Pedersen signature $\sigma = (z, c, r, P_2)$ on the message msg as described above. The whole protocol is described more precisely in Fig. 8.
- **RESIGN:** this procedure is played similarly to the SIGN one, except that this is not necessary to send m_{blind} anymore.
- **VERIF:** the verification of the validity of a blind signature on a message msg is done as described above for the Chaum and Pedersen signature scheme.

Security features

The security of our invariable partially blind signature scheme is given by the three following propositions. The security proofs are provided in Appendix A [Security analysis of our first IPBS scheme](#).

Fig. 8 The Chaum-Pedersen based IPBS scheme

Theorem 1 (Correctness) *If both parties follow the protocol then $\sigma = (z, c, r, P_2)$ is a valid Chaum-Pedersen signature on $\text{msg} = M \| m \| m^*$.*

Theorem 2 (First and second partial blindness) *For any common information M , the signer's view of the execution of the IPBS protocol is statistically independent of the messages m and m^* and the signature $\sigma = (z, c, r, P_2)$. As a consequence, our IPBS scheme verifies the first and second partial blindness properties.*

Theorem 3 (Unforgeability) *If there exists an adversary $\mathcal{A}$ against the unforgeability of our IPBS scheme, then $\mathcal{A}$ can be used to produce a signature forgery of the Chaum-Pedersen blind signature scheme.*

Another generic construction

Another possible way to design an invariable partially blind signature scheme is to use the so-called Camenisch-Lysyanskaya signature schemes, as introduced above. More precisely, we would have the following description for CL

Fig. 9 The Camenisch-Lysyanskaya based IPBS scheme

signature schemes based on *elliptic curves* and *bilinear maps* (see section 5 of Camenisch and Lysyanskaya (2004)).

- **SETUP:** let G be a group of prime order q and g_1, g_2, g_3, g_4, g_5 five generators in G . These data constitutes the public parameters. Let $\mathcal{H}, \mathcal{H}_1$ and $\mathcal{H}_2$ be three secure hash functions. The signer owns a private key sk of a CL signature scheme. The corresponding public key is denoted by pk .
- **SIGN:** in order to obtain an invariable partially blind signature on a message $\text{msg} = M \| m^*$ where $M, m^* \in \{0, 1\}^*$, the user $\mathcal{U}$ will first choose three random values $a, b, r \in [1, q - 1]$ and computes a commitment C' on a, b and a commitment C'' on r and $\mathcal{H}(m^*)$: $C' = g_1^a g_2^b$ and $C'' = g_3^r g_4^{\mathcal{H}(m^*)}$.² He will then computes two non-interactive proofs P_1 and P_2 in order to prove that he knows the committed values $P_1 = \text{NIZK}[(\alpha_1, \alpha_2) : C' = g_1^{\alpha_1} g_2^{\alpha_2}]$ and $P_2 = \text{NIZK}[(\alpha_3, \alpha_4) : C'' = g_3^{\alpha_3} g_4^{\alpha_4}]$ and send M, C', C'', P_1 and P_2 to the signer $\mathcal{S}$. The signer will verify the proofs P_1 and P_2 and if they are valid, she will choose a random value s and compute the commitment $C = C' C'' g_3^s g_5^{\mathcal{H}_2(M)}$. $\mathcal{S}$ will then use her secret key sk to compute a Camenisch-Lysyanskaya signature Σ on C (which is in fact a signature on the tuple $(a, b, r + s, \mathcal{H}_1(m^*), \mathcal{H}_2(M))$). Finally, $\mathcal{S}$ will send s and Σ to $\mathcal{U}$. $\mathcal{U}$ will then computes $D = g_1^a$ and a non-interactive proof P_3 that he knows a signature on these committed values:

$$P_3 = \text{NIZK}[(\alpha_1, \alpha_2, \alpha_3, \beta) : D = g_1^{\alpha_1} \wedge \beta = \text{CLSIGN}(\alpha_1, \alpha_2, \alpha_3, \mathcal{H}_1(m^*), \mathcal{H}_2(M))].$$

The pair (D, P_3) represents the invariable partially blind signature on the message $\text{msg} = M \| m^*$. For our identity federation protocol D will correspond to the federation alias and C' to the blind alias. The whole protocol is described more precisely in Fig. 9.

²The message M will be known by the signer whereas m^* should be unknown to her.

- **RESign**: this step is simply derived from the above description and will not be detailed in this paper.
- **VERIF**: the verification of a signature is done by checking that the non-interactive proof P_3 is correct, using standard techniques.

Client-side identity federation and SSO protocols

As explained in the above sections, the aim of the identity federation approach we propose is to enable the users to control the federation links between their identities while leaving the actual user authentication to the IdPs in the network.

The identity federation protocol

Again, the first time a user uses a particular IdP to authenticate to a SP, the user may federate her SP's identity to her IdP's identity. In our proposal, the reference given by the IdP to the SP with regard to a user is a blinded alias, as described in Fig. 5. The global identity federation protocol is given in Fig. 10 and works as follow (the IdP plays the role of the signer in the underlying invariable partially blind signature protocol whereas the SP acts as a verifier):

Fig. 10 Our identity federation protocol

1. The user accesses to a SP.
2. SP redirects the user to the IdP with *AuthnRequest*. This authentication request is a standard SAML 2.0 authentication request that contains in particular a request identifier, the date of the request and the SP's identifier. The request identifier and the date of the request should be blinded (so the IdP cannot read them) since this information could be used by the IdP and the SP to establish a link between the identities.
3. IdP verifies the validity of the authentication request and authenticates the user.
4. The IdP determines that the user *user@idp* has not yet federated her identity with her identity local to the SP. The IdP and the user then engage in the invariable partially blind signature protocol given in the previous section (see Fig. 8) with the following inputs.
 - The message M contains the information the IdP is allowed to read including the SP's identity. As mentioned above the IdP will act as a signer in the IPBS.
 - The message m^* is the information in the request/response that can be used to identify the user, such as the request identifier or the date of the request. This message is, therefore, blinded.
 - The message m is the federation alias, computed as $m = g_2^v$ where v is a random value chosen by the user (see Section “[Implementation considerations](#)”). This message is blinded.

The IdP then associates the blinded alias to the local user identity and the SP. At the end of the blind signature protocol, the user obtains an authentication response *AuthnResponse* signed by the IdP (in other words the user obtains from the IdP a valid signature on m^* and the alias m). The IdP only knows the blinded version of the alias ($alias_{blind}$) and has no way, thanks to the property of the IPBS, to compute the unblind version (*alias*).

5. The user sends the signed authentication response to the SP. This authentication response corresponds to a standard authentication response. Only the signature of the authentication response differs from the standard recommendations. The signature is composed of the following information as mentioned previously: z value, c value, r value and P_2 .
6. SP verifies the IdP's signature of the *AuthnResponse* and extracts *alias*. SP then authenticates the user, retrieves the local identity *user@sp* and associates it the federation alias *alias*.
7. Finally, SP gives the user the requested service. The identity *user@idp* of the user at IdP is now federated to her identity *user@sp* at SP with the federation alias *alias*.

Let us now focus on the step 4 dealing with the blind signature scheme interactions between the user and the IdP. This step can be divided as follows.

1. The IdP sends to the user a *Blind Alias Request* to begin the blind signature protocol of Fig. 8.

2. The user creates the alias by computing g^s , blinds it by computing m_{blind} and sends it to the IdP, together with some other values corresponding to m_0 and the proof P_1 that everything has been correctly computed. We can consider here that the part M of the global message can be either sent by the user to the IdP or directly generated by the IdP himself.
3. The next step corresponds to the exchange between the user and the signer of the blind signature protocol until the user receives r_0 from the IdP (i.e. the signer). This is also at the end of this step that the IdP associates the blinded alias m_{blind} with $user@idp$. We could moreover interpret this step as the generation and the signature by the IdP of the *AuthnResponse*. This *AuthnResponse* including the blinded alias and a blind version of the sensible information included in the authentication response is sent to the user.
4. Finally, the user processes the *AuthnResponse* by unblinding the alias and the blinded information in such a way that the signed *AuthnResponse* now includes the actual federation alias.

The single sign on protocol

Once the identity federation is performed, the Single Sign On (SSO) is now possible for this user and this service provider. The new SSO protocol is again different from the identity federation one (Fig. 10) since the service provider does not need to authenticate the user. The service provider uses the alias to retrieve the identity of the user. More precisely, we have the following steps.

1. The user accesses to a SP
2. SP redirects the user to the IdP with *AuthnRequest*.
3. IdP verifies the validity of the authentication request and optionally authenticates the user (authentication is not required if the user has already authenticated and if the IdP maintains an authentication session locally). IdP retrieves from his database the corresponding blinded alias for the requested SP.
4. The IdP and the user then engage in the partially blind signature protocol. More precisely, there is no need for the user to send m_{blind} to the IdP since the IdP already has it in its database. However, in the first part of the blind signature protocol of Fig. 8, since the message M contains the SP's identity, the computation of m_0 and P_1 by the user are necessarily done again. At the end of the protocol, the user obtains an authentication response *AuthnResponse* signed by the IdP and containing the unblinded alias.
5. The user sends the signed authentication response to SP.
6. SP verifies the IdP's signature on *AuthnResponse*, extracts the embedded alias and retrieves the local identity $user@sp$ associated with the federation alias *alias*.
7. The SP finally sends a response to the initial service request from the user.

Implementation considerations

In this section, we first give some details on security aspects that need to be studied for a practical implementation, and we give first results of the performance measurements we have performed on our own implementation.

Details on the implementation

We first focus on implementation considerations by highlighting key aspects that need to be considered in order to ensure that the users' privacy is actually protected and that no security flaw is introduced.

- The blind alias together with the local identity of the user at the IdP side and the related SP should be stored in the user database of the IdP.
- In accordance with the SAML 2.0 specifications, the blind aliases should be different from one SP to another for a particular user (this prevents SPs from linking the partial identities they manage) and from one user to another.

One possible idea is that the user and the IdP jointly generate this alias $m = g_2^v$, such that two aliases are necessarily different. For example, m could be equal to $g_2^{v_1} g_2^{v_2}$ where v_1 is secretly chosen by the user and v_2 is given by the IdP. It is moreover necessary for the user to prove, using standard ZKPK, that the sent m_{blind} is correctly formed using v_1 and v_2 : $NIZK(v_1, s : m_{blind}/g_2^{v_2} = g_2^{v_1} g^s)$.

- For a particular SP and for a given user, the authentication response should always include the same alias except if one of the protagonist (user, SP or IdP) asks for a different one.
- The user should be able to verify that the alias embedded in the authentication response is not the one related to an incorrect SP. This case may happen since this is the IdP who embeds the blinded alias in the blind signature process of the SSO protocol. Note that this verification can be easily done since the user can verify the IdP signature before sending the authentication response to the SP. If the alias is not the right one, the verification will fail since the user will not use the corresponding random values (see below).

It is important to note that, contrary to some SAML 2.0 specifications that are based on standard web browser, the approach we propose requires the installation of a blind signature module at the client side.

Moreover, some information included in the identity federation protocol needs to be used by the user during the SSO process. More precisely, the user needs to reuse the random values implied in the blinding of the alias (this is why we need a particular partially blind signature scheme, as explained in Section “[On the use of blind signature schemes](#)”). Indeed, the random value s used to blind the alias g^v , chosen during the identity federation protocol between the user and the IdP is also used during all future SSO protocols. Thus, the user should have to store this value s . For this purpose, it is possible

to use a single password pwd and a cryptographic function called $HMAC$. Thus this random value can e.g. be $s = HMAC(K, IdP, SP, pwd)$ such that only one password is needed to compute all necessary values. The same type of computations can also be used to pseudo-randomly choose the alias in such a way that two aliases of a same user are different from one SP to another.

One solution to the problem of random generation and storage can be to use a smart card, a dongle or a mobile phone to store the necessary values and make the computations. This can be very useful to address mobility in the client-side identity federation approach we propose.

First performance estimation

A prototype of this system has been implemented in Java, using a 32 bits Intel centrino duo processor at 1.66 GHz and 2 Gbytes RAM for the client side and an Intel(R) Xeon(R) CPU 5110 at 1.60 GHz for the providers.

We first consider the normal federated identity case without the blind signature process, using a 2048-bits RSA signature scheme (with CRT), and we obtain an overall running time of approximatively 300 ms.

We have implemented our first construction based on the Chaum-Pedersen signature scheme, with a 1024-bits modulus. With this configuration, the overall running time of the federation process or the SSO protocol is 1.1 s (measured from the browser). This includes the cryptographic part, the exchanges, the message formatting (in PHP), the javascript execution, etc.

The cryptographic part needs nearly 240 ms and can be divided as follows:

- client-side: 155 ms, including 45 ms for the first part of the protocol, 50 ms for the second part and optionally 60 ms to verify the obtained signature;
- IdP-side: 26 ms in total (the second step is quite immediate);
- SP-side: 60 ms to verify the signature.

Conclusion

In this paper, we have presented a client-side identity federation approach that complements the SAML 2.0 network-based identity federation model and the client-centric identity model introduced with the identity selector concept. In the network-side identity federation model, the users entrust the IdPs with the management of the links between their identities. In the client-side identity federation approach, the links between identities are managed at the client side by the users themselves, providing, this way, a better protection of the users' privacy. Like the network-side approach, it prevents the SPs from mass-correlating their databases. But in addition, it also prevents collusion between IdPs and SPs that would enable a mass-correlation of identity databases. Unlike the client-centric identity model, the authentications are performed by the IdPs in the network which enables the SPs to maintain their trust

relationships with these IdPs and to accept the authentication claims they provide.

Our approach is based on the introduction of a new type of blind signature scheme that we call *invariable partially blind signature*. It enables the user to ask the IdP to sign authentication responses without revealing the federation alias actually used by the SP. We give an example of such blind signature scheme and apply it to our identity federation model.

Our client-side identity federation model complements the existing approaches. The users should be able to decide the most appropriate model depending on the very nature of the identities to federate and on the trust she puts on the IdPs. The network-side identity federation model is probably more relevant for partial identities that are slightly independent (an identity at Orange and an Identity at eBay for instance) and when the users put sufficient trust in the IdPs (and the network) to entrust it with the management of the federation links between these identities. On the contrary, the client-side identity federation model might be chosen when the user wants to keep its identities separate. This might be the case when the identities are such that the damage that the correlation of these identities would cause is deemed important (a federation between a bank identity and a government identity for instance), and when the SPs refuse to trust authentications performed on the users' devices and to accept authentication claims provided by the IdP hosted on these devices.

However, the client-side identity federation model requires that blind signature modules are installed on the user's device. This type of cryptographic primitive has been or should be integrated in client software like Higgins or CardSpace through modules called Idemix (Camenish and Lysyanskaya 2001) and U-Prove (Brands 2000; Brands et al. 2007) respectively to improve user privacy and protect the user personal information. Integrating a client-side identity federation module in such client software seems to be relevant and would provide the users with an even more seamless and consistent user experience.

A Security analysis of our first IPBS scheme

Let us analyze, informally speaking, the security of the invariable partially blind signature scheme described in Section “[A first construction based on the Chaum-Pedersen signature](#)” (see Fig. 8).

A.1 Correctness

If both parties follow the protocol then $\sigma = (z, c, r, P_2)$ is a valid Chaum-Pedersen signature on $\text{msg} = M \| m \| m^*$.

Let $c = \mathcal{H}(M \| m^* \| m \| z \| a \| b)$. We have to prove that

$$g^r h^c = a \text{ and } (g_1^{\mathcal{H}(M)} m g_3)^r z^c = b.$$

The first equality is proved as follows:

$$g^r h^c = g^{ur_0+d} h^{c_0u} = (g^{r_0} h^{c_0})^u g^d = a_0^u g^d = a$$

For the second equality, we have:

$$\begin{aligned} (g_1^{\mathcal{H}(M)} m g_3)^r z^c &= (m_0/g^s)^r (z_0/h^s)^c = (m_0^r z_0^c)/(g^{sr} h^{sc}) \\ &= (m_0^{ur_0+d} z_0^{c_0u})/(g^r h^c)^s \\ &= (m_0^{r_0} z_0^{c_0})^u m_0^d/(g^r h^c)^s \\ &= b_0^u m_0^d/(g^r h^c)^s = b_0^u m_0^d/a^s = b. \end{aligned}$$

Therefore the verification condition

$$c = \mathcal{H}(M \| m^* \| m \| z \| g^r h^c \| (g_1^{\mathcal{H}(M)} m g_3)^r z^c)$$

holds.

A.2 First and second partial blindness

For any common information M , the signer's view of the execution of the IPBS protocol is statistically independent of the messages m and m^* and the signature $\sigma = (z, c, r, P_2)$. As a consequence, our IPBS scheme verifies the first and second partial blindness properties.

To prove that our IPBS protocol satisfies the partial blindness requirement, we have to show that for every possible view and every possible signature (on the same common information M) there exists exactly one triplet $(\tilde{u}, \tilde{d}, \tilde{s})$ of blinding factors which would result in that particular signature and view.

Given any signer's view consisting of $M, m_{blind}, a_0, b_0, z_0, c_0, r_0$ and any signature $\tilde{\sigma} = (\tilde{z}, \tilde{c}, \tilde{r}, \tilde{P}_2)$ on a message $\widetilde{msg} = M \| \tilde{m} \| \tilde{m}^*$ (where $\tilde{m} = g_2^{\tilde{v}}$), let $\tilde{u} = \tilde{c}/c_0 \pmod{q}$, $\tilde{d} = \tilde{r} - \tilde{u}r_0 \pmod{q}$ and $\tilde{s}$ denote the discrete logarithm of $m_{blind}/\tilde{m}$ in the base g (in other words $m_{blind} = g_2^{\tilde{v}} g^{\tilde{s}}$).

Let $a' = a_0^{\tilde{u}} g^{\tilde{d}}$ and $b' = b_0^{\tilde{u}} m_0^{\tilde{d}}/a'^{\tilde{s}}$.

It remains to show that:

$$a' = \tilde{a} = g^{\tilde{r}} h^{\tilde{c}} \quad (1)$$

$$b' = \tilde{b} = \tilde{m}_u^{\tilde{r}} \tilde{z}^{\tilde{c}} \quad (2)$$

where $\tilde{m}_u = g_1^{\mathcal{H}(M)} \tilde{m} g_3 = m_0/g^{\tilde{s}}$.

Let us first observe that since we assume that

$$(M, m_{blind}, a_0, b_0, z_0, c_0, r_0)$$

is a valid view and $\tilde{\sigma} = (\tilde{z}, \tilde{c}, \tilde{r}, \tilde{P}_2)$ a valid signature on $\widetilde{\text{msg}} = M \parallel \tilde{m} \parallel \tilde{m}^*$, we have:

$$\begin{aligned} a_0 &= g^{r_0} h^{c_0} \\ b_0 &= m_0^{r_0} z_0^{c_0} \\ \tilde{a} &= g^{\tilde{r}} h^{\tilde{c}} \\ \tilde{b} &= \tilde{m}_u^{\tilde{r}} \tilde{z}^{\tilde{c}} \\ \tilde{z} &= \tilde{m}_u^x = m_0^x / g^{\tilde{s}x} = z_0 / h^{\tilde{s}} \end{aligned}$$

The last equation holds because the signer actually proves that z_0 equals m_0^x (and indirectly that $\tilde{z} = \tilde{m}_u^x$) when making a blind signature.

Let us show that the first equality (1) holds:

$$a' = a_0^{\tilde{u}} g^{\tilde{d}} = (g^{r_0} h^{c_0})^{\tilde{u}} g^{\tilde{r} - \tilde{u}r_0} = g^{r_0\tilde{u}} h^{c_0\tilde{u}} g^{\tilde{r} - \tilde{u}r_0} = g^{\tilde{r}} h^{\tilde{c}} = \tilde{a}.$$

For the second equality, we have:

$$\begin{aligned} b' &= b_0^{\tilde{u}} m_0^{\tilde{d}} / a'^{\tilde{s}} = (m_0^{r_0} z_0^{c_0})^{\tilde{u}} m_0^{\tilde{d}} / (g^{\tilde{r}\tilde{s}} h^{\tilde{c}\tilde{s}}) \\ &= (m_0^{r_0\tilde{u} + \tilde{d}} z_0^{\tilde{c}}) / (g^{\tilde{r}\tilde{s}} h^{\tilde{c}\tilde{s}}) = m_0^{\tilde{r}} z_0^{\tilde{c}} / (g^{\tilde{r}\tilde{s}} h^{\tilde{c}\tilde{s}}) \\ &= (m_0 / g^{\tilde{s}})^{\tilde{r}} (z_0 / h^{\tilde{s}})^{\tilde{c}} = \tilde{m}_u^{\tilde{r}} \tilde{z}^{\tilde{c}} = \tilde{b}. \end{aligned}$$

Consequently $\tilde{\sigma} = (\tilde{z}, \tilde{c}, \tilde{r}, \tilde{P}_2)$ could have been produced during the execution of the IPBS protocol which led to the view consisting of $M, m_{\text{blind}}, a_0, b_0, z_0, c_0, r_0$.

This completes the proof.

A.3 Unforgeability

If there exists an adversary $\mathcal{A}$ against the unforgeability of our IPBS scheme, then $\mathcal{A}$ can be used to produce a signature forgery of the Chaum-Pedersen blind signature scheme.

The unforgeability of the signatures produced by the IPBS protocol is an immediate consequence of the security of the Chaum-Pedersen blind signature scheme (Chaum and Pedersen 1992). Before showing that a signature forgery of the Chaum-Pedersen blind signature scheme (CPBSS for short) “reduces” to a signature forgery of our IPBS protocol, let us first recall how the CPBSS works. Let p and q be two primes such that $q \mid p - 1$. Let G be a subgroup of $\mathbb{Z}/p\mathbb{Z}$ of order q and let $g \in G$. The secret key of the signer is an integer $x \in [1, q - 1]$ and the corresponding public key is $h = g^x$. To get a blind signature on the message $\text{msg} = (m^*, m)$ where $m^* \in \{0, 1\}^*$ and $m \in G$, the user chooses a random $s \in [1, q - 1]$ and computes $m_0 = mg^s$. Let $z_0 = m_0^x$. The signer then proves that $\log_g h = \log_{m_0} z_0$ as described in Fig. 11 below.

At the end of the protocol, the user retrieves a Chaum-Pedersen signature $\sigma = (z, c, r)$ on the message $\text{msg} = (m^*, m)$. The signature is considered as

Fig. 11 Chaum-Pedersen blind signature scheme

valid iff $c = \mathcal{H}(m^* \| m \| z \| g^r h^c \| m^r z^c)$. Note that if the execution of the protocol is successful then this implies that $z = m^x$.

We will also need for our reduction to slightly modify the **Sign** protocol and the **Verif** algorithm of our IPBS. However, only the computation of the challenge c and consequently the verification of the validity of a signature are modified. The challenge in this variant of our IPBS is computed by the user as follows: $c = \mathcal{H}(M \| m^* \| m \| z \| g_1^{\mathcal{H}_1(M)} mg_3 \| z \| g^r h^c \| (g_1^{\mathcal{H}_1(M)} mg_3)^r z^c)$ (instead of $c = \mathcal{H}(M \| m^* \| m \| z \| g^r h^c \| (g_1^{\mathcal{H}_1(M)} mg_3)^r z^c)$). Note that when the execution of the protocol is successful we have $z = (g_1^{\mathcal{H}_1(M)} mg_3)^x$.

Suppose we have an adversary $\mathcal{A}$ who breaks the unforgeability of our IPBS with non negligible probability. We can construct an algorithm $\mathcal{B}$ using $\mathcal{A}$ as an oracle, which breaks the unforgeability of the CPBSS. $\mathcal{B}$ receives on input from its challenger $(g, h = g^x)$ the public key of the CPBSS and has oracle access to a CPBSS signature oracle (denoted by $\mathcal{O}_{CP}$ in the sequel). It also chooses a secure hash function $\mathcal{H}_1$ as well as three random generators g_1, g_2, g_3 of G and sends its public key $(g, g_1, g_2, g_3, h = g^x, \mathcal{H}_1)$ to $\mathcal{A}$.

$\mathcal{B}$ now will have to answer to the signature requests (**Sign** or **ReSign** requests) of $\mathcal{A}$. For a **Sign** request $\mathcal{B}$ plays as follows (the simulation is similar for **ReSign** requests): when $\mathcal{B}$ will receive m_{blind} , M , P_1 from $\mathcal{A}$, it will compute $m_0 = g_1^{\mathcal{H}_1(M)} mg_3$ and sends this value to $\mathcal{O}_{CP}$. $\mathcal{O}_{CP}$ will send back $a_0 = g^w$, $b_0 = m_0^w$, $z_0 = m_0^x$ to $\mathcal{B}$ that $\mathcal{B}$ will transmit to $\mathcal{A}$.

When $\mathcal{B}$ will receive the challenge c_0 from $\mathcal{A}$, $\mathcal{B}$ will transmit it to $\mathcal{O}_{CP}$. $\mathcal{O}_{CP}$ will send back a value $r_0 = w - c_0 x \pmod{q}$ to $\mathcal{B}$ that it will forward to $\mathcal{A}$. $\mathcal{B}$ therefore perfectly simulates the signer of the IPBS protocol.

Eventually $\mathcal{A}$ outputs $(l_1 + l_2) + 1$ valid distinct signatures (where l_1 corresponds to the number of **Sign** requests and l_2 to the number of **ReSign**

requests): $(\text{msg}_1 = (M_1, m_1^*, m_1), \sigma_1), (\text{msg}_2 = (M_2, m_2^*, m_2), \sigma_2), \dots, (\text{msg}_{l+1} = (M_{l+1}, m_{l+1}^*, m_{l+1}), \sigma_{l+1})$ where $l = (l_1 + l_2)$.

Then $(\text{MSG}_1 = (M_1 \| m_1^* \| m_1 \| g_1^{\mathcal{H}_1(M_1)} m_1 g_3), \sigma_1), (\text{MSG}_2 = (M_2 \| m_2^* \| m_2 \| g_1^{\mathcal{H}_1(M_2)} m_2 g_3), \sigma_2), \dots, (\text{MSG}_{l+1} = (M_{l+1} \| m_{l+1}^* \| m_{l+1} \| g_1^{\mathcal{H}_1(M_{l+1})} m_{l+1} g_3), \sigma_{l+1})$ represent $l + 1$ valid signatures of the CPBSS although only l signatures have been requested to $\mathcal{O}_{CP}$. $\mathcal{B}$ has therefore broken the unforgeability property of the CPBSS.

We have shown that if there exists an adversary $\mathcal{A}$ that can produce $l + 1$ signatures after having requested only l signatures to the signer of our IPBS scheme (one-more unforgeability) then $\mathcal{A}$ can straightforwardly be used to produce a signature forgery of the Chaum-Pedersen blind signature scheme. As the later scheme is conjectured to be one-more unforgeable (see Brands 1993) such an adversary cannot exist. A detailed security analysis of the Chaum-Pedersen blind signature scheme can be found in Brands (1993).

References

- Abe M. A secure three-move blind signature scheme for polynomially many signatures. In: EUROCRYPT '01. Lecture Notes in Computer Science. vol. 2045. New York: Springer; 2001. p. 136–151.
- Brands SA. An efficient off-line electronic cash system based on the representation problem. 1993. Technical report, Amsterdam, The Netherlands, The Netherlands.
- Camenisch J, Lysyanskaya A. Signature schemes and anonymous credentials from bilinear maps. In: CRYPTO 2004. Lecture Notes in Computer Science. vol. 3152. New York: Springer; 2004. p. 56–72.
- Camenisch J, Piveteau J-M, Stadler M. Blind signatures based on the discrete logarithm problem. In: EUROCRYPT'94. 1994. p. 428–432.
- Chaum D. Blind signatures for untraceable payments. In: CRYPTO. 1982. p. 199–203.
- Chaum D. Blind signature system. In: CRYPTO. 1983. p. 153.
- Chaum D, Pedersen TP. Wallet databases with observers. In: CRYPTO'92. Springer; 1992. Lecture Notes in Computer Science. vol. 740, p. 89–105.
- Cameron K. The laws of identity. Redmond: Microsoft Corporation; 2005.
- Fiat A, Shamir A. How to prove yourself: practical solutions to identification and signature problems. In: CRYPTO. 1986. p. 186–194.
- Maler E, Mishra P, Philpott R. Assertions and protocol for the oasis security assertion markup language (saml). 2003. OASIS Standard. September.
- Schnorr C-P. Efficient identification and signatures for smart cards. In: CRYPTO. 1989. p. 239–252.
- Camenisch J, Lysyanskaya A. Efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In: EUROCRYPT 2001. LNCS. vol. 2045. New York: Springer; 2001. p. 93–118.
- Camenisch J, van Herreweghen E. Design and implementation of the idemix anonymous credential system. In: Proceedings of the 9th ACM conference on Computer and communications security. New York: ACM; 2002. ISBN 1-58113-612-9, p. 21–30.
- Brands S. Rethinking public key infrastructures and digital certificates. 1st edition, ISBN 0-262-02491-8, August. Cambridge: MIT; 2000.
- Brands S, Demuyneck L, De Decker B. A practical system for globally revoking the unlinkable pseudonyms of unknown users, in ACISP 2007. LNCS. vol. 4586. New York: Springer; 2007. p. 400–415.