A Delegation Solution for Universal Identity Management in SOA

Yang Zhang and Jun-Liang Chen

Abstract—The relationship-focused and credential-focused identity managements are both user-centric notions in Service-oriented architecture (SOA). For composite services, pure user-centric identity management is inefficient because each subservice may authenticate and authorize users and users need participate in every identity provisioning transaction. If the above two paradigms are unified into the universal identity management where identity information and privileges are delegatable, user centricity will be more feasible in SOA. The credential-focused system is a good starting point for constructing a universal identity management system. However, how to implement a practical delegation scheme is still a challenge although there are some delegatable anonymous credential schemes that were theoretically constructed. This paper aims to propose a practical delegation solution for universal identity management. For this, a pseudonym-based signature scheme is first designed where pseudonyms are self-generated and unlinkable for realizing user's privacy. Next, a proxy signature is presented with the pseudonyms as public keys where delegation can be achieved through certificate chains. Finally, the security of our scheme is analyzed and proved in the random oracle model.

Index Terms—Privacy concerns of service-oriented solutions, identity management, privacy governance methods and tools, privacy management in data dissemination, service-oriented architecture.

1 Introduction

1.1 Motivation

In Service-oriented architecture (SOA), individuals often use identity providers to provide their identity information and the identities are represented by a set of attributes [1], [2]. Based on the user-centricity philosophy, identity management systems [3], [4] are classified into the relationship-focused and credential-focused systems [5]. In the relationship-focused system, identity providers play an important role and are involved in each transaction that conveys identity information to a service provider. The users only adopt identity providers to provide identity information and have control over his attributes. Therefore, the users participate in every identity provisioning transaction. On the contrary, in the credential-focused system, the users obtain long-term credentials from identity providers and store them locally. Then, these credentials are used to provide identity information without involving the identity provider. The users are still involved in every identity transaction as in the relationship-focused approach.

The above two paradigms provide user-centric privacy management in personal data dissemination. The apparent major advantage of user centricity is that users control through their involvement in each transaction. In fact, it is also the major drawback of user centricity because they cannot handle delegations. A universal identity management system incorporates the advantages of these user-centric systems and provides the delegation capability. The credential-focused system is suitable for constructing a universal identity management system. However, constructing a universal identity management system from credential-focused identity management systems is nonintuitive and complex, because the specific properties of anonymity, minimal data disclosure, and various anonymity revocation capabilities involve multiparty transactions. These render delegation a formidable task to tackle.

In SOA, the users may access more than one service in any given time. When they use the same pseudonym to access different services, all their transactions are linkable. It would be ideal if the users can self-generate different pseudonyms based on their credentials without interacting with identity providers. On the other hand, the users should use the same pseudonym to link different actions in one transaction and delegate their privilege to others in order to improve the runtime performance of composite services. Therefore, the features of self-generation of pseudonyms and delegation of identity information are essential in the universal identity management system. For instance, let it be assumed that a user by name Liming visits a hospital called People-Health. Following the visit, he has to get certain clinical tests done in some examination centers several times, the results of which are required for proper comprehensive diagnosis. Owing to privacy restricts, most of the test centers do not reveal this data to anyone but the user. Therefore, Liming himself has to retrieve the data every time it is needed by the People-Health. As this exercise is cumbersome, Liming would look for a method that enables the People-Health to directly retrieve the desired data from the various centers and to generate the pseudonyms without online identity providers. Such a capability, besides being more efficient, is vital to handle cases of emergency.

1.2 Related Works and Research Questions

There is no straightforward transformation of anonymous credential schemes without delegation into delegatable

• *The authors are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Room 640, Teaching Building 3, No. 10, Xi Tu Cheng Road, Haidian District, Beijing 100876, China. E-mail: {yangzhang, chjl}@bupt.edu.cn.*

Manuscript received 4 May 2009; revised 28 Oct. 2009; accepted 12 Mar. 2010; published online 24 Mar. 2010
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSCSI-2009-05-0103. Digital Object Identifier no. 10.1109/TSC.2010.9.

schemes. Classic anonymous credential systems were introduced by Chaum [6] in 1985, as a way of allowing the user to work effectively, but anonymously, with multiple organizations. The works of [7], [8], [9], [10] developed the model and implementation of anonymous credential systems. Camenisch and Lysyanskaya [11], [12], [13] proposed anonymous credential systems, which are more efficient than the earlier ones, by constructing a signature scheme with efficient protocols. All the above systems use interactive zero-knowledge protocols to prove the possession of credentials without optimizing the rounds of interaction. Belenkiy et al. [14] introduced noninteractive anonymous credentials in 2007 to solve this problem. However, the scheme could not be directly transformed into a delegatable anonymous credential scheme.

Delegatable anonymous credential schemes were proposed in [15], [16] where users can obtain credentials from identity providers and delegate their credentials to other users. If an identity provider issues user A a credential for his given pseudonym Nym_A, user A can prove to user B that Nym_A has a credential from the identity provider. Credentials received directly from the identity provider are level 1 credentials, those that have been delegated once are level 2 credentials, and so on. User A can also delegate his credential to user B, and user B can then prove that he has a level 2 credential from the identity provider where user B proves to others his possession of credential without involving any identity information of A. However, the size of the possession proof increases with increase in delegation level and cannot be bounded to a constant number by aggregating proofs. Furthermore, in those schemes, identity providers are involved in issuing credentials when a new pseudonym is generated, and these schemes are not efficient either as far as network resources are concerned.

Camenisch et al. proposed a general certification framework for SOA where identity information can be privacy-enhanced [17]. They claim that their framework can be integrated into today's Public Key Infrastructure (PKI) on the Internet. The framework includes cryptographic primitives for realizing the functionality, definition of protocol interfaces for the *CertificateIssuance* and *CertificateProof* protocols, and a powerful specification language with well-defined semantics that allows defining the data to be released in a transaction. However, the framework and implementation do not specify how to realize delegation. The solution offered here not only utilizes the advantage of the general certification framework, but also provides a new way of implementing the *CertificateIssuance* and *CertificateProof* protocols to realize delegation.

In this paper, a pseudonym-based signature scheme is proposed to construct practical delegation solutions for universal identity management where users can self-generate pseudonyms based on their credentials. The self-generated pseudonyms are used as public keys. The privacy is ensured by the unlinkability between different pseudonyms. According to this idea, we get a natural solution to the delegation problem. A conventional signature scheme often immediately allows for (nonanonymous) delegatable credentials: A, who has a public signing key and a certification chain of length L can sign B's public key, giving B a certification chain of length $L+1$. Therefore, the delegation solution consists of two signature schemes in which the pseudonyms are used as public keys: a pseudonym-based signature scheme and a conventional proxy signature scheme [18], [19], [20], [21]. The pseudonym-based signature scheme provides anonymous proof of possession of credentials to protect user's privacy where service providers verify the signature to decide whether the signer has the rights to access the services. In the conventional proxy signature scheme, the original signers delegate their signing capability to proxy signers without divulging their private keys. Then, the proxy signer creates a valid signature on behalf of the original signer. The receiver of the signature verifies the signature and the original signer's delegation together. Our proxy signature scheme has the delegation capability by warrant. A warrant explicitly states the signers' identity, delegation period, and the qualification of the message on which the proxy signer can sign, etc.

Zero-knowledge proofs of credential possession in classic anonymous credential systems can be converted to signatures via Fiat-Shamir heuristic [22]. Compared with pseudonym-based signature schemes, the converted signature schemes have four undesirable features: First, the pseudonyms in the converted signature schemes are not self-generated; second, an identity provider must provide online issuing service; third, the ability to sign under the pseudonyms is lacked; fourth, the signature size is often too long. Compared with group signature schemes [23], [24], [25], [26], [27], the pseudonym-based signature scheme requires self-generated pseudonyms that can be used as public keys, while group signature schemes do not need any pseudonym, nor they specify how to generate them. Therefore, besides self-generation feature, the scheme proposed here has the desirable feature that the pseudonyms are used as temporal public keys for signatures. This feature can be compared to that of the ID-based signature schemes where signers have the ability to sign with their identities. Unlike the identity in ID-based signature schemes, pseudonyms are not directly bound to real identities or certificates because pseudonyms are self-generated. Thus, the security notion is a bit different from that of the ID-based signature schemes. To prove the unforgeability of pseudonym-based schemes, the challenger should distinguish between forged pseudonyms and valid randomized pseudonyms generated by adversaries who have obtained some valid credentials.

Our pseudonym-based scheme is similar to the Direct Anonymous Attestation (DAA) scheme [28], [29] which coordinates a Trust Platform Module (TPM) and a host together to generate pseudonyms and signatures. The scheme was adopted by the Trusted Computing Group as the method for remote authentication of a TPM, while preserving the privacy of the user of the platform that contains the module. Compared with DAA, this scheme does not require a TPM to work online, which improves the performance of the scheme. Although DAA adopts group signature schemes [23], [24], [25], [26], [27] to generate pseudonyms that link transactions, it is not clear how it can be used to build a delegation solution for universal identity management. While DAA uses pseudonyms to link transactions, ours uses pseudonyms to achieve privacy because one pseudonym is unlinkable to the other. Moreover, ours uses pseudonyms as public keys for signatures.

1.3 Contributions

The contribution of this paper is threefold. Our first contribution is that a signature-based natural approach is adopted to construct the delegation solution for universal identity management in SOA. Our second contribution is that the novel concept of pseudonym-based signature scheme is introduced where pseudonyms are self-generated and messages can be bound to the self-generated pseudonyms which are used as the public keys for signatures. Based on this, the novel pseudonym-based signature scheme is constructed. Our third contribution is the application of our scheme to universal identity management systems where the delegation of privilege to access services is realized by adopting warrant proxy signature schemes [18], [19], [20], [21] based on time-varying pseudonyms.

1.4 Organization of the Paper

The remainder of the paper is structured as follows: Section 2 gives a description on preliminaries. Section 3 contains our constructions. Section 4 focuses on analysis. Section 5 presents our deployment. Finally, conclusions are drawn in Section 6.

2 Preliminaries

Suppose we have groups G_1 and G_2 of the same prime order p and security parameter κ. Assume that the discrete logarithm problem is hard in both groups. Then, we need a cryptographic bilinear map $e : G_1 \times G_1 \rightarrow G_2$ to satisfy the following properties [30], [31]:

1. Bilinearity: $\forall a, b \in Z_p^*$, $P, Q \in G_1$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. No-degeneracy: For any point $P \in G_1$, $e(P, P) \neq 1_{G_2}$.
3. Computability: There exists an efficient algorithm to compute $e(P, Q)$ for $\forall P, Q \in G_1$.

To give the security proof of our scheme, we introduce a problem which is slightly different from the one proposed by Mitsunari et al. [32] and is called Collusion Attack Algorithm with k Traitors and n Examples ((k,n)-CAA).

Definition 1 ((k,n)-CAA). *Let (G_1, G_2, e) be as above, k, n be integers, $P, P_1, \ldots, P_n \in G_1, x \in Z_p$. Given*

$$\{P, P_j, a_i \in Z_p, xP, xP_j, 1/(x + a_i)P \mid 1 \leq i \leq k1 \leq j \leq n.\},$$

to compute $1/(x + a)P$ for some

$$aP_j \notin \{a_i P_j \mid 1 \leq i \leq k, 1 \leq j \leq n\}.$$

The (k,n)-CAA is considered to be hard in the literature. That is, the success probability of any probabilistic, polynomial-time, 0/1 valued algorithm in solving (k,n)-CAA problem is negligible. A function $F(y)$ is said to be negligible if it is less than $1/y^l$ for every fixed $l > 0$ and sufficiently large integer y.

3 Constructions

Participants in a delegation solution for universal identity management comprise identity providers (who grant credentials), user u (who obtains credentials), user v (who is delegated by u to access services, which can be a service provider), and service providers. Our solution is different from the delegatable anonymous credential systems. In the latter, the user first registers a pseudonym with the identity provider, and then, the identity provider grants to the user a credential associated with the registered pseudonym. Our solution, on the other hand, allows the users to first get a credential from the identity provider, and then, to generate multiple new pseudonyms as needed, while the identity provider does not participate. Without interacting with the identity provider, the users can show to service providers that they possess the right credentials. Compared with relationship-based delegation scheme, in our solution, the users can self-generate different pseudonyms and directly delegate privileges to others without the identity provider's participation.

The delegation solution can be modeled by the following six subprotocols:

Setup. The identity provider generates system parameter and system public/private key pairs.

Credential issuing. The identity provider grants a secret credential to user u.

Pseudonym generation. User u generates a new pseudonym in current time slot according to its secret credential and time stamp. Two pseudonyms are unlinkable and only loose time synchronization is required.

Signing-warrant. User u signs a warrant that contains delegation period, delegated pseudonyms, and the services to be accessed, etc. User v obtains a proxy key according to his private key and the signature of the warrant.

Delegation-signing. User v generates proxy signatures on behalf of user u.

Delegation-verification. Service providers verify proxy signatures from v together with u's delegation.

The relationship among the six subprotocols is illustrated in Fig. 1. If the service provider trusts the identity provider (IdP) and a secret credential is issued to the delegator u by the IdP, u can grant or delegate the delegatee v the access privilege to the service when v does not have the privilege. u uses the pseudonym generation protocol to protect its privacy and the signing-warrant protocol to grant privilege.

Our delegation solution adopts two signature schemes to implement these subprotocols. The first one is pseudonym-based signature scheme which provides anonymous proof of possession of credentials to protect user's privacy. Service providers verify the signature to decide whether the signer has the rights to access the services. The second one is a warrant proxy signature scheme where the original signer u delegates his signing capability to the proxy signer v without leaking his private key, and then, the proxy signer creates a valid signature on behalf of the original signer. The pseudonym-based signature scheme mainly implements *Credential Issuing*, *Pseudonym Generation*, and *Signing-warrant*, and the warrant proxy signature scheme mainly implements *Delegation-Signing* and *Delegation-Verification*.

When v again delegates his signing capability to other users, he adopts the warrant proxy signature scheme, instead of pseudonym-based signature scheme. Compared with certificate chain approaches to delegate signing capability, our solution can aggregate signatures for the warrant m_w, verifies the aggregated signature only once, and avoids verifying signatures one by one down the certificate chain. According to the general certification framework for SOA [17], our signature-based delegation solution has

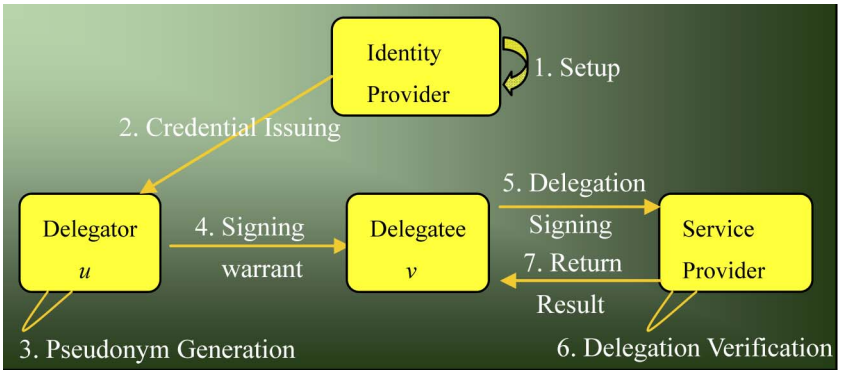

Fig. 1. Relationship in the delegation solution.

potential to be integrated into today's PKI where *Signing-warrant* can realize the *CertificateIssuance protocol*, and *Delegation-Signing* and *Delegation-Verification* can realize the *CertificateProof* protocol.

In this section, the pseudonym-based scheme Π_{sig} is first presented. Then, the warrant proxy signature scheme Π_{psig} is given. In order to provide the administrator of a service domain with a means to control the generation of pseudonyms, we propose a pseudonym-controlled variation of Π_{sig} in the last part of this section.

3.1 Pseudonym-Based Signature Scheme Π_{sig}

In the pseudonym-based signature scheme Π_{sig}, unlike in an identity-based signature scheme [33], [34], the user can noninteractively renew public/private key pairs. If the renewed public key is viewed as a pseudonym, then a pseudonym can be self-generated.

In Π_{sig}, an identity provider (or a domain's manager, or an organization management center) generates the credential Cre for the user u. The user u generates, by accessing Cre, a pseudonym $(P_u, \bar{P}_u)$ that is unlinkable to other pseudonyms. Without the identity provider reissuing Cre, u can renew $(P_u, \bar{P}_u)$ by accessing Cre. The user u uses different pseudonyms to prevent adversaries from linking different transactions and analyzing their traffic patterns.

Π_{sig} is modeled by five algorithms as in Table 1.

TABLE 1
Pseudonym-Based Signature Scheme

PGen	*It generates the system parameters param and the master-key* ms = s .
Gen	*Executed by the identity provider, it generates the credential Cre for the user u .*
$\Delta - Gen$	*Executed by u , it generates the pseudonym* $(P_u, \bar{P}_u)$ *and corresponding secret value* μ' .
Sign	*It takes as input the user's private key* (μ, μ', S_u) *and the message m to return the signature of m under* (μ, μ', S_u) .
Verify	*It takes as input u 's pseudonym* $(P_u, \bar{P}_u)$, *the organization public key* (W, W_i) , *the message m , and the signature sig to return either 1 or 0.*

In our construction, the identity provider periodically publishes a set of public restriction keys $\{W_i, \ldots, W_j\}$ which correspond to the time slots $\{slot_i, \ldots, slot_j\}$. A new public restriction key begins to work when a new time slot starts. The public restriction keys are used to enable the user to update his pseudonyms. The function $T(time)$ takes $time$ as input and outputs a time slot where only loose time synchronization is required.

Definition 2. Π_{sig} *is made up of the following five algorithms:*

$PGen(1^\kappa)$
Setup G_1, G_2, e *and* $P \in G_1$,
pick cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$,
compute $Q_i = H_1(T(time_i))$,
choose a master-key $s \in_R Z_p$,
compute organization's public keys $W = sP, W_i = sQ_i$,
return $ms = s, param = (G_1, G_2, e, P, W, W_i, H_1, H_2)$.

$Gen(ms, param, u)$
$\mu \in_R Z_p$
$Cre = (\mu, S_u) = (\mu, 1/(s+\mu)P)$
return Cre.
User u *can verify the correctness by checking*
$\mathrm{e}(\mu P + W, S_u) = e(P, P)$.

$\Delta - Gen(Cre, param, time_i)$
$Q_i = H_1(T(time_i))$, $\mu' \in_R Z_p$
$P_u = (\mu' + \mu)Q_i$, $\bar{P}_u = \mu' S_u$
return $((P_u, \bar{P}_u), \mu')$
The key pair $(P_u, \bar{P}_u)/(\mu, \mu', S_u)$ *of the user* u *satisfy*
$e(P_u + W_i, S_u) = e(Q_i, P)e(Q_i, S_u)^{\mu'} = (Q_i, P)e(Q_i, \bar{P}_u)$.

$Sign(m, (P_u, \bar{P}_u), (\mu, \mu', S_u), param, time_i)$
$r, r' \in_R Z_p$
$R_{G_1} = rQ_i$, $R = [e(Q_i, P)e(Q_i, \bar{P}_u)]^{r'}$
$c = H_2(m || \mathrm{R}_{G_1} || R || P_u || \bar{P}_u || T(time_i))$
$z_1 = c(\mu' + \mu) + r$, $z_2 = c\mu' + r'$
return $sig = (\mathrm{c}, z_1, z_2)$.

$Verify(m, sig, (P_u, \bar{P}_u), param, time_i)$
parse $sig = (\mathrm{c}, z_1, z_2)$
$Q_i = H_1(T(time_i))$
$\hat{R}_{G_1} = z_1 Q_i - cP_u$
$\hat{R} = [\mathrm{e}(Q_i, P)e(Q_i, \bar{P}_u)]^{z_2} / e(P_u + W_i, \bar{P}_u)^c$
$\hat{c} = H_2(m \parallel \hat{R}_{G_1} \parallel \hat{R} \parallel P_u \parallel \bar{P}_u \parallel T(time_i))$
return $c \stackrel{?}{=} \hat{c}$.

The correctness of Π_{sig} is illustrated below. Assume that the elements in the tuple $sig = (c, z_1, z_2)$ are given as in $Sign$. Then,

$$\begin{aligned}\hat{R}_{G_{1i}} &= z_1 Q_i - cP_u = (c(\mu' + \mu) + r)Q_i - c(\mu' + \mu)Q_i \\ &= rQ_i = R_{G_1},\end{aligned} \tag{1}$$

$$\begin{aligned}\hat{R} &= [e(Q_i, P)e(Q_i, \bar{P}_u)]^{z_2} / e(P_u + W_i, \bar{P}_u)^c \\ &= [e(Q_i, P)e(Q_i, \bar{P}_u)]^{c\mu'} [e(Q_i, P)e(Q_i, \bar{P}_u)]^{r'} / \\ &\quad [e(P_u + W_i, \bar{P}_u)]^c,\end{aligned} \tag{2}$$

where

$$\begin{aligned}[e(P_u + W_i, \bar{P}_u)]^c &= [e((\mu' + \mu)Q_i + sQ_i, \mu'/(s + \mu)P)]^c \\ &= [e(Q_i, \mu' P)e(\mu' Q_i, \mu'/(s + \mu)P)]^c \\ &= [e(Q_i, P)e(Q_i, \bar{P}_u)]^{c\mu'}, \\ \Rightarrow \hat{R} &= [e(\mathrm{Q_i}, \mathrm{P})e(\mathrm{Q_i}, \bar{P}_u)]^{c\mu'} [e(Q_i, P)e(\mathrm{Q_i}, \bar{P}_u)]^{r'} / \\ &\quad [e(Q_i, P)e(Q_i, \bar{P}_u)]^{c\mu'} \\ &= [e(Q_i, P)e(\mathrm{Q_i}, \bar{P}_u)]^{r'} \\ &= R.\end{aligned}$$

That is, $c = \hat{c}$ according to (1) and (2).

This signature scheme is converted from a zero-knowledge proof via Fiat-Shamir heuristic [22]. The prover u and verifier v undertake a proof of knowledge values $(\mu + \mu', \mu')$ satisfying the following equation:

$$e(P_u + W_i, \bar{P}_u) = e((\mu' + \mu)Q_i + sQ_i, \mu'/(s + \mu)P)$$

and

$$P_u = (\mu' + \mu)Q_i.$$

That is,

$$e(P_u + W_i, \bar{P}_u) = [e(Q_i, P)e(Q_i, \bar{P}_u)]^{\mu'}$$

and

$$P_u = (\mu' + \mu)Q_i.$$

The protocol for proving knowledge of the discrete logarithm is as follows:

1. Prover u: chooses $r, r' \in_R Z_p$
   computes $R_{G_1} = rQ_i$, $R = [e(Q_i, P)e(Q_i, \bar{P}_u)]^{r'}$
   sends (R_{G_1}, R) to the verifier v.
2. Verifier v: receives (R_{G_1}, R)
   chooses $c \in_R Z_p$
   sends c to the prover u.
3. Prover u: receives c
   computes $z_1 = c(\mu' + \mu) + r$, $z_2 = c\mu' + r'$
   sends (z_1, z_2) to the verifier v.
4. Verifier v: receives (z_1, z_2)
   verifies the correctness by checking
   $z_1 Q_i = cP_u + R_{G_1}$ and
   $e(P_u + W_i, \bar{P}_u)^c R = [e(Q_i, P)e(Q_i, \bar{P}_u)]^{z_2}$.

In our pseudonym-based signature scheme, the value c is noninteractively obtained by computing the hash value of $(m||R_{G_1}||R||P_u||\bar{P}_u||T(time_i))$ according to the Fiat-Shamir heuristic [22]. If u proves possession of the knowledge of the discrete logarithm $(\mu', \mu + \mu')$ satisfying $e(P_u + W_i, \bar{P}_u) = [e(Q_i, P)e(Q_i, \bar{P}_u)]^{\mu'}$ and $P_u = (\mu' + \mu)Q_i$, the verifier v will believe that the credential $Cre = (\mu, 1/(s + \mu)P)$ is issued to $(P_u, \bar{P}_u)$ by the identity provider. When the user u signs a message under $(P_u, \bar{P}_u)$, the identity provider can track the transcripts by iteratively computing $e(P_u + W_i, S_u) = e(Q_i, P)e(Q_i, \bar{P}_u)$ according to its stored private keys $\{S_1, S_2, \ldots\}$.

3.2 Proxy Signature Scheme Π_{psig}

Assume that the user v has a public/private key pair $(PK_v = \nu Q_i, \nu)$ and the user u has some rights to access one service sev with a public/private key pair $(PK_u = (P_u, \bar{P}_u), (\mu, \mu', S_u))$. The user u can grant the user v to delegate himself to access the service sev. It does not matter whether the user v has the rights to access sev. The service provider of sev verifies the proxy signature from v, and then, know whether v is indeed delegated by PK_u. If v is delegated by PK_u and PK_u has rights to access sev, v will be allowed to access sev. u creates the warrant m_w which contains related information such as $\bar{P}_u$, a part of the pseudonym of u, the delegation period, etc. Also, u generates the signature α for m_w and conveys both the signature and m_w to v. v creats a proxy key from α and m_w. The following proxy signature scheme is from the work of [35] where the key generation algorithm of the original signer is slightly different.

Definition 3 (Π_{psig} scheme). *Π_{psig} is made up of the following five algorithms:*

$Setup(1^\kappa)$
Generate G_1, G_2, e and $Q_i \in G_1$,
pick cryptographic hash functions $H_1, H_2 : \{0,1\}^* \to G_1$,
return $param = (G_1, G_2, e, Q_i, H_1, H_2)$.

$KGen(param, u, v)$
The key of the original signer u:
$\tau = \mu' + \mu, PKey_u = P_u = \tau Q_i, SK_u = \tau$
The key of the proxy signer v:
$PKey_v = \nu Q_i, SK_v = \nu, \nu \in_R Z_p$

$PKGen(param, \tau, \nu, PKey_u)$
Generate a proxy key for v:
Create the warrant m_w.
u signs m_w: $\alpha = \tau H_1(m_w)$,
v checks whether (m_w, α) satisfies
$e(\alpha, Q_i) = e(H_1(m_w), PKey_u)$.
If true, v gets a proxy key (α, ν)

$PSign(m, param, m_w, (\alpha, \nu))$
v generates proxy signatures:
$\sigma = \alpha + \nu H_2(m||m_w)$
return $sig = (m, \sigma)$.

$PVerify(m, sig, param, m_w, PKey_u, PKey_v)$
Return $e(\sigma, Q_i) \stackrel{?}{=} e(H_1(m_w), PKey_u)e(H_2(m||m_w), PKey_v)$

The security proof of this scheme can be found in [35].

When u delegates his signing capability to the user v, v will possess u's privilege to access the services. u adopts the pseudonym-based signature scheme Π_{sig} to generate time-varying pseudonyms for privacy and the proxy signature

scheme Π_{psig} to realize delegation. Therefore, the delegation solution consists of Π_{sig} and Π_{psig}. When v again delegates the privilege to another user x, Π_{psig} is executed as follows:

$Setup(1^\kappa)$
Generate G_1, G_2, e *and* $P \in G_1$,
pick cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$,
return $param = (G_1, G_2, e, P, H_1, H_2)$.

$KGen(param, u, v)$
The key of the original signer v:
$PKey'_v = PKey_u + PKey_v$, $SK_v = \nu$
The key of the proxy signer x:
$PKey_x = \chi Q_i, SK_x = \chi$, $\chi \in_R Z_p$

$PKGen(param, \nu, \chi, PKey'_v)$
Generate a proxy key for x:
v *signs* m_w: $\alpha' = \alpha + \nu H_1(m_w)$, *where* m_w *is created by* u *and* α *is the signature for* m_w *produced by* u.
x *checks whether* (m_w, α') *satisfies* $e(\alpha', Q_i) = e(H_1(m_w), PKey'_v)$.
If true, x *gets a proxy key* (α', χ)

$PSign(m, param, m_w, (\alpha', \chi))$
x *generates proxy signatures*:
$\sigma = \alpha' + \chi H_2(m||m_w)$
return $sig = (m, \sigma)$.

$PVerify(m, sig,\ param, m_w, PKey'_v, PKey_x)$
return $e(\sigma, Q_i) \stackrel{?}{=} e(H_1(m_w), PKey'_v)e(H_2(m||m_w), PKey_x)$

Compared with certificate chain approaches to delegate signing capability, our solution can aggregate signatures for the warrant m_w, verify the aggregated signature only once, and thus, avoids verifying signatures one by one down the certificate chain. That is to say, if $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_n$ is the user chain for delegation and $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3 \rightarrow \cdots \rightarrow \alpha_n$ is the signature chain for the warrant m_w which are, respectively, produced by $\{x_1, x_2, x_3, \ldots, x_n\}$, then x_n obtains the aggregated signature $\alpha = \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n$ and the proxy key $(\alpha, SK_{x_n} = \chi_{x_n})$. x_n generates the proxy signature for the message m as follows:

$$\sigma = \alpha + \chi_{x_n} H_2(m||m_w).$$

Service providers compute $PKey = PKey_{x_1} + PKey_{x_3} + PKey_{x_4} + \cdots + PKey_{x_n}$ and verify

$$e(\sigma, Q_i) \stackrel{?}{=} e(H_1(m_w), PKey)e(H_2(m||m_w), PKey_{x_n}).$$

If the signature σ is valid and x_1 has proved the possession of issued credentials, the service providers will allow x_n to access the services. The service providers verify only one signature and not the signatures produced by each delegation user in order to show that the last delegatee has the privilege for accessing the services. In addition, the conventional certificate chain can also be used in our solution.

Reverting to the healthcare example cited in Section 1, Liming can adopt Π_{sig} to generate the new pseudonym $(P_{Li\min g}, \bar{P}_{Li\min g})$, and signs the warrant m_w that contains $\bar{P}_{Li\min g}$, time period, and the names of those health examination services. Then, Liming conveys the signature α and the warrant m_w to People-Health to enable them to obtain the proxy key (α, ν). Therefore, People-Health can directly request the results of health examination from the test centers by signing the request messages under the proxy key (α, ν) according to the algorithm $PSign$ in Π_{psig}. The concerned test centers or examination centers have stored the test results of the user $(P_{Li\min g}, \bar{P}_{Li\min g})$ because Liming adopted the pseudonym as his identity and proved by using Π_{sig} that $(P_{Li\min g}, \bar{P}_{Li\min g})$ has the rights to consume the examination services and access the results, which protects the privacy of Liming by not disclosing his real identity, exact age, and other identity information. When the request messages signed by People-Health are received, the examination centers verify the corresponding signature according to the algorithm $PVerify$ in Π_{psig}. If the signature is valid, the stored results will be conveyed to People-Health.

In some settings, the identity provider of a service domain may want to control the generation of pseudonyms. For example, when users require adequate protection of their personal information, anonymous communication services are often used to deliver the consumer services [2]. If the users are just in an ad hoc network, they will be not only service consumers but also service providers. In this case, the pseudonym generation approach is required to have the pseudonym-uniqueness property for a period of time. Otherwise, if adversaries have controlled one node, they can forge different pseudonyms according to different neighbors, that is, they can forge false topology. They can also use a different pseudonym for a different instance of anonymous communication services and infer their traffic patterns by distinguishing messages relating to different pseudonyms, which often expose the VIP private information or their action characteristics. Therefore, the identity providers should have the means to manage how to self-generate pseudonyms besides delegation of identity information. We propose a variation of the pseudonym-based signature scheme to satisfy the requirement in the next section.

3.3 Pseudonym-Controlled Variation of Π_{sig}

In order to provide control over self-generation properties of pseudonyms, the algorithm $\Delta - Gen$ of Π_{sig} is modified such that the new pseudonyms do not take effect until the start of new time slot and publishing of new restriction keys. Identity providers manage pseudonym generation by adopting restriction keys and time slot.

The identity provider periodically publishes a set of public restriction keys $\{W_i, \ldots, W_j\}$ which correspond to time slots $\{slot_i, \ldots, slot_j\}$. A new public restriction key begins to work when a new time slot starts. The public restriction keys are used to ensure uniqueness of the pseudonym of a user in a single time slot and make it possible for the user to generate his pseudonyms. The pseudonym-controlled variation Π_{vsig} of pseudonym-based signature is as follows:

Definition 4 (Π_{vsig} scheme). Π_{vsig} *is made up of five algorithms as follows:*

$PGen(1^\kappa)$
Setup G_1, G_2, e *and* $P \in G_1$,
pick cryptographic hash functions $H_1, H_2 : \{0,1\}^* \rightarrow G_1$,
compute $Q_i = H_1(T(time_i))$,
choose a master-key $s \in_R Z_p$,
compute organization's public keys $W = sP, W_i = sQ_i$,
return $ms = s, param = (G_1, G_2, e, P, W, W_i, H_1, H_2)$.

$Gen(ms, param, u)$
$\mu \in_R Z_p$
$Cre = (\mu, S_u) = (\mu, 1/(s+\mu)P)$
return Cre.
User u *can verify the correctness by checking* $e(\mu P + W, S_u) = e(P, P)$.

$\Delta - Gen(Cre, param, time_i)$
$Q_i = H_1(T(time_i))$
$P_u = \mu Q_i$,
return P_u
The key pair $P_u/(\mu, S_u)$ *of the user* u *satisfy* $e(P_u + W_i, S_u) = e(Q_i, P)$.

$Sign(m, P_u, (\mu, S_u), param, time_i)$
$\alpha, r, r' \in_R Z_p$
$T = \alpha S_u$, $R_{G_1} = rQ_i$, $R = e(Q_i, P)^{r'}$
$c = H_2(m||T||R_{G_1}||R||P_u||T(time_i))$
$z_1 = c\alpha + r'$, $z_2 = c\mu + r$
return $sig = (T, c, z_1, z_2)$.

$Verify(m, sig, P_u, param, time_i)$
parse $sig = (T, c, z_1, z_2)$
$Q_i = H_1(T(time_i))$
$\hat{R}_{G_1} = z_2 Q_i - cP_u$
$\hat{R} = e(Q_i, P)^{z_1}/e(P_u + W_i, T)^c$
$\hat{c} = H_2(m||T||\hat{R}_{G_1}||\hat{R}||P_u||T(time_i))$
return $c \stackrel{?}{=} \hat{c}$.

The correctness of Π_{vsig} is illustrated below. Assume that the elements in the tuple $sig = (c, z_1, z_2)$ are given as in $Sign$. Then,

$$\begin{aligned}\hat{R}_{G_1} &= z_2 Q_i - cP_u = (c\mu + r)Q_i - c\mu Q_i = rQ_i \\ &= R_{G_1},\end{aligned} \tag{3}$$

$$\begin{aligned}\hat{R} &= e(Q_i, P)^{z_1}/e(P_u + W_i, T)^c \\ &= [e(Q_i, P)]^{c\alpha}[e(Q_i, P)]^{r'}/[e(P_u + W_i, T)]^c,\end{aligned} \tag{4}$$

where

$$\begin{aligned}[e(P_u + W_i, T)]^c &= [e(\mu Q_i + sQ_i, \alpha/(s+\mu)P)]^c \\ &= [e(Q_i, \alpha P)]^c = [e(Q_i, P)]^{c\alpha},\end{aligned}$$

$$\begin{aligned}\Rightarrow \hat{R} &= [e(Q_i, P)]^{c\alpha}[e(Q_i, P)]^{r'}/[e(Q_i, P)]^{c\alpha} \\ &= [e(Q_i, P)]^{r'} = R.\end{aligned}$$

That is, $c = \hat{c}$ according to (3) and (4).

This signature scheme is also converted from a zero-knowledge proof via Fiat-Shamir heuristic. The prover u and verifier v undertake a proof of knowledge values (μ, α) satisfying the following equation:

$$e(P_u + W_i, T) = e(\mu Q_i + sQ_i, \alpha/(s+\mu)P),$$

and

$$P_u = \mu Q_i,$$

That is,

$$e(P_u + W_i, T) = e(Q_i, P)^{\alpha},$$

and

$$P_u = \mu Q_i.$$

When the identity provider in the domain publishes some pseudonyms $\{P_u^i, \ldots, P_u^j\}$ of user u in its certificate revocation list, the credential of u is revoked in time slots $\{slot_i, \ldots, slot_j\}$. This revocation solution is simple and attractive because the computation is efficient and the pseudonyms of u that are not in these time slots are still unlinkable. The signatures produced by u are also traceable since the identity provider can compute all the users' pseudonyms in all the time slots according to $\{\mu_1, \mu_2, \ldots\}$.

When Π_{vsig} is adopted to construct delegation solutions combined with Π_{psig}, the warrant m_w will contain related information such as the delegation period, service name, and so on. Unlike the Π_{sig}-based scheme, it does not include $\bar{P}_u$ which is part of the public key of u. The security proof of Π_{vsig} is similar to that of Π_{sig} and is not given in this paper.

4 Proofs and Analysis

Our solution is based on two signature schemes and adopts the natural signature chain to accomplish delegation. So, it is important to prove the security of the two signature schemes. As the proxy signature scheme had already been proved secure in the other work [35], the pseudonym-based signature scheme remains to be proved secure. Compared with convenient signature schemes, in order to prove the security of pseudonym-based schemes, the challenger should distinguish forged pseudonyms and valid randomized pseudonyms generated by adversaries who may have obtained some valid credentials. Therefore, we first define the security model of pseudonym-based signatures, and then, prove our pseudonym-based signature scheme is secure.

Definition 5 (Unforgeability security notion for Π_{sig}). *A pseudonym-based signature scheme is said to be strongly existentially unforgeable under adaptive-chosen message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in this game:*

1. *The challenger runs the setup algorithm to generate the system's parameters and sends them to the adversary A.*
2. *The adversary A performs a series of queries:*
   a. *Key queries: A impersonates the user u to query keys. The challenger accesses Gen oracle and returns the credential Cre.*
   b. *Pseudonym queries: A uses time and user u to query pseudonyms. The challenger accesses $\Delta - Gen$ oracle and returns pseudonym $(P_u, \bar{P}_u)$ as public key and S_u as the user's private key.*
   c. *Signature queries: A sends the message m, a time, and the pseudonym $(P_u, \bar{P}_u)$, and then, receives a signature on m that was generated by the signature oracle using the secret key corresponding to the pseudonym $(P_u, \bar{P}_u)$.*
3. *After a polynomial number of queries, A produces the signature sig on the message m for $(P_u, \bar{P}_u)$ whose corresponding private key and credential were never asked during stage 2. The message, signature pair*

(m,sig) was not returned either by the signature oracle during stage 2.

The adversary A wins the game if the signature verification algorithm outputs 1 when it is run on the tuple $((P_u,\bar{P}_u), m,sig,T(time))$. The adversary's advantage is defined to be its probability of producing a forgery taken over the coin-flippings of the challenger and A.

Theorem 1. *If the (k,n)-CAA assumption is true, then Π_{sig} is existentially unforgeable under adaptive-chosen message attacks in the random oracle model.*

Proof. The correctness of the scheme is straightforward. So, we prove that it is unforgable. If there is an adversary A that succeeds in attacking the scheme, then we can construct a probabilistic polynomial time algorithm B to solve the (k,n)-CAA problem. Let k,n be integers, and $\{P,P_j,a_i\in Z_p,xP,xP_j,1/(x+a_i)P|1\le i\le k,1\le j\le n\}$ be a random instance of the (k,n)-CAA problem taken as input by B. B solves (k,n)-CAA problem by adopting A as a subalgorithm. B initializes A as follows:

PGen. B sets up (G_1,G_2,e), $P,P_1,\ldots,P_n\in G_1$, and H_1,H_2. Let $W=xP$, $Q_i=\lambda_iP_i$ for the ith time slot where the λ_iP_i is the response of the random oracle H_1, and $W_i=x\lambda_iP_i$. Send to A the parameter $param=(G_1,G_2,e,Q_i,W,W_i,H_1,H_2)$.

The adversary A then starts performing queries such as those described under Definition 5. These queries are answered by B as follows:

Queries on oracle H_1. B prepares q_{H_1} responses $\{q_1,q_2,\ldots,q_{q_{H_1}}\}$ of the hash oracle queries and $P_1,\ldots,P_n$ are distributed randomly in this set. When the ith time slot $T(time)$ is submitted to the random oracle H_1, B picks q_i if q_i is equal to some element in $\{P_1,\ldots,P_n\}$ where the element is denoted by P_i without loss of generality. Otherwise, the simulation halts and fails. If no halt, B chooses $\lambda_i\in_R Z_p$, computes λ_iP_i as the hash value $H_1(T(time_i))$, and memorizes $(T(time),\lambda_i,P_i)$.

Queries on oracle H_2. When the random oracle H_2 has a string $(m||R_{G_1}||R||P_u||\bar{P}_u||T(time_i))$ as input, B picks a random $c\in_R Z_p$, and returns c.

Key queries. B prepares q_{key} responses $\{w_1,w_2,\ldots,w_{q_{key}}\}$ of the key oracle queries and $a_1,a_2,\ldots,a_k$ are distributed randomly in this set. When a user u_i is submitted to the Gen oracle, B picks $(w_i,1/(s+w_i)P)$ if w_i is equal to some element in $\{a_1,a_2,\ldots,a_k\}$ where the element is denoted by a_i without loss of generality, and returns $cre=(a_i,A_i=1/(s+a_i)P)$. Otherwise, the simulation halts and fails.

$\Delta-Gen$ queries. When A queries the pseudonym of u in the time slot $T(time)$, B randomly chooses $a'_i\in_R Z_p$ and computes $P_u=(a'_i+a_i)\lambda_iP_i$ and $\bar{P}_u=a'_iA_i$, then returns $((P_u,\bar{P}_u),a'_i)$. $(P_u,\bar{P}_u)$ *and* (a_i,a'_i,A_i) satisfy $e(P_u+W_i,A_i)=e(Q_i,P)e(Q_i,A_i)^{a'_i}=e(Q_i,P)e(Q_i,\bar{P}_u)$.

Signature queries. When A queries the signature oracle on the message m for the pseudonym $(P_u,\bar{P}_u)$, B chooses $c,z_1,z_2\in_R Z_p$ and computes $R_{G_1}=z_1Q_i-cP_u$ and $R=[e(Q_i,P)e(Q_i,\bar{P}_u)]^{z_2}/e(P_u+W_i,\bar{P}_u)^c$. Finally, B lets the answer of the random oracle H_2 is c when it takes as input $(m||R_{G_1}||R||P_u||\bar{P}_u||T(time_i))$. If this causes a collision, i.e., if B previously set the oracle at this point to some other $c\prime$, the simulation halts and fails. (c,z_1,z_2) is returned to A and appears as a valid signature from A's point of view.

Eventually, A produces a fake signature $sig=(c,z_1,z_2)$ for the tuple $(m,(P_u,\bar{P}_u),time)$ with non-negligible advantage. B checks by iteratively calculating $e(P_u+W_i,A_x)=e(Q_i,P)e(Q_i,\bar{P}_u)$ according to its stored $\{A_x=1/(x+a_x)P|1\le x\le k.\}$ whether μ', Cre, and $(P_u,\bar{P}_u)$ are queried by A. If the answer is yes, the simulation declares failure and exits. If no, B continues the successive procedure. From the Forking Lemma [36], after a polynomial replay of the attacker A, B obtains another valid signature $sig=(c',z'_1,z'_2)$ for the same $(m,(P_u,\bar{P}_u),time)$ with $c\ne c'$, $\hat{R}_{G_1}=R_{G_1}$, and $R=R'$. Then, B knows that

$$\begin{aligned}
&R_{G_1}=z_1Q_i-cP_u=z'_1Q_i-c'P_u=R'_{G_1} &\Rightarrow\\
&(z_1-z'_1)Q_i=(c-c')P_u &\Rightarrow\\
&P_u=\frac{z_1-z'_1}{c-c'}Q_i
\end{aligned}$$

$$\begin{aligned}
&R=[e(Q_i,P)e(Q_i,\bar{P}_u)]^{z_2}/e(P_u+W_i,\bar{P}_u)^c\\
&=[e(Q_i,P)e(Q_i,\bar{P}_u)]^{z'_2}/e(P_u+W_i,\bar{P}_u)^{c'}=R' &\Rightarrow\\
&e(Q_i,P)^{z_2-z'_2}e(Q_i,\bar{P}_u)^{z_2-z'_2}=e(P_u+W_i,\bar{P}_u)^{c-c'} &\Rightarrow\\
&e(Q_i,P)^{(z_2-z'_2)/(c-c')}=e\left(P_u-\frac{z_2-z'_2}{c-c'}Q_i+W_i,\bar{P}_u\right) &\Rightarrow\\
&e\left(P_u-\frac{z_2-z'_2}{c-c'}Q_i+W_i,\frac{c-c'}{z_2-z'_2}\bar{P}_u\right)=e(Q_i,P) &\Rightarrow\\
&e\left(\frac{(z_1-z'_1)-(z_2-z'_2)}{(c-c')}P_i+xP_i,\frac{c-c'}{z_2-z'_2}\bar{P}_u\right)=e(P_i,P).
\end{aligned}$$

B gets

$$S=\frac{c-c'}{z_2-z'_2}\bar{P}_u \text{ for } aP_i=\frac{(z_1-z'_1)-(z_2-z'_2)}{(c-c')}P_i.$$

If $aP_i\notin\{a_xP_y|1\le x\le k,1\le y\le n.\}$, B gets a solution (aP_i,S) to the (k,n)-CAA instance. If $aP_i\in\{a_xP_y|1\le x\le k,1\le y\le n.\}$ and (aP_i,S) is not queried by A (otherwise, the simulation declares failure and exits), B gets a solution (aP_i,S) to the (k − 1,n)-CAA instance or to the (k,n − 1)-CAA instance [37].

Assume that A has an advantage ε in forging a signature, we assess B's advantage in solving the (k,n)-CAA problem. The probability for B not to fail in key queries is $(k/q_{key})^{q_{key}}$ (q_{key} is the number of key queries by A). The probability for B not to fail in the random oracle H_1 queries is $(n/q_{H_1})^{q_{H_1}}$ (q_{H_1} is the number of random oracle H_1 queries by A). So, the probability for B to succeed in simulating is $(k/q_{key})^{q_{key}}(n/q_{H_1})^{q_{H_1}}$ which is non-negligible. From the Forking Lemma [36], B gets two valid signatures with non-negligible probability. It comes that B's advantage is non-negligible. □

The unlinkablity of different pseudonyms of user u is achieved by randomizing the pseudonyms [38]. Because μ' is randomly picked from Z_p, the signing keys $(\mu',\mu'+\mu)$ are indistinguishable from random elements in Z_p. As a result, the pseudonym $\bar{P}_u=\mu'S_u$ and $P_u=(\mu'+\mu)Q_i$ are undistinguishable from random elements in G_1. Therefore, the pseudonyms are unlinkable.

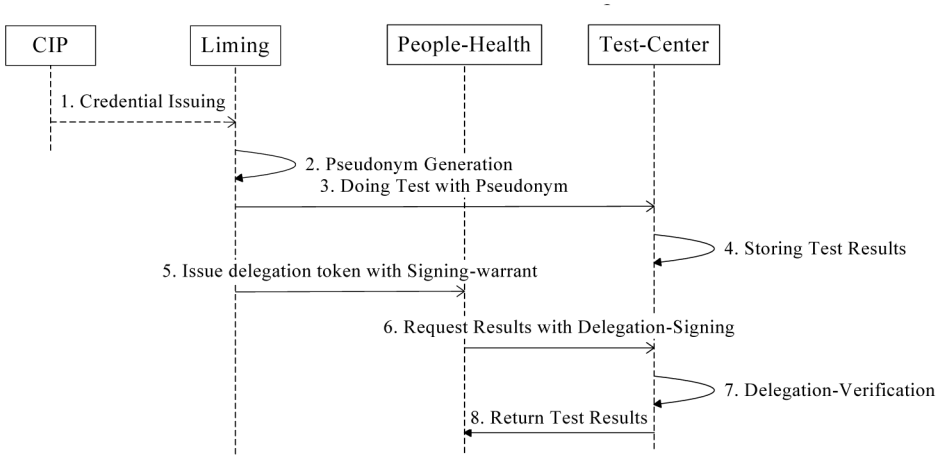

Fig. 2. Delegation solution for healthcare example.

5 Deployment

SOA is a very popular paradigm for system integration and interoperation. Web service is the current standard for SOA. Therefore, industry pursues the deployment of identity management systems in distributed different security domains, called Federated Identity Management (FIM) [3], [4], to build one cornerstone of the Web service security. Current user-centricity FIM systems are almost relationship-focused, and can enhance the user's privacy by following the data minimization and transaction unlinkability principles. Multiple industry products [39], [40] embrace this paradigm. However, the bottleneck effect will become more serious for identity providers if the delegation function is implemented only based on the relationship-focused model. Some credential-focused systems are also developed to achieve FIM. The example is idemix [41], [42], but it does not support self-generation and efficient delegation. This section describes how to deploy our solution in the relationship-focused paradigm by integrating the credential-focused paradigm to realize universal identity management.

We adopt the concept from WS-Federation [4] to describe how to deploy the delegation solution for universal identity management. As a public specification, WS-Federation defines a framework to allow different security domains to federate, such that authorized access to Web services can be realized in distributed realms. That includes mechanisms for brokering of identity, attribute, authentication, and authorization assertions between domains, and privacy of federated claims. Based on WS-Trust [43], WS-Federation supports delegation by using identity providers to issue appropriate security tokens for entities in different security domains. As a relationship-focused framework, it also has the undesirable features of general relationship-focused paradigms. Our scheme can be deployed in this relationship-focused framework with enhanced credentials to achieve universal identity management. The basic entities mapping between the delegation solution and WS-Federation are provided as follows:

IdP. An Identity Provider is an entity that acts as an authentication service to end-requestors and as a data origin authentication service to service providers. IdPs are trusted third parties to maintain the requestor's some identity information. The original IdP is enhanced to issue secret credentials by adding ***Credential Issuing*** interface.

Requestor. An end user, an application, or a machine is typically represented by a digital identity and may have multiple valid digital identities. The original requestor is enhanced to self-generate pseudonyms and delegate privileges by adding the issued party interface of ***Credential Issuing*** and a ***Signing-warrant*** interface.

Resource. A Web service, service provider, or any valuable things. Sometimes, it can act as another requestor. The original resource is enhanced to sign messages and verify signatures by adding ***Delegation-Signing*** and ***Delegation-Verification*** interfaces. When it acts as a requestor, it delegates privileges to other resources by adding a ***Signing-warrant*** interface.

Before describing general deployment in typical scenarios, we revert to the healthcare example cited in Section 1. Assume the Citizen Identity Provider (CIP) issues Liming a digital identity credential which can be used to prove that he is a citizen in that city. Fig. 2 illustrates the application of the delegation solution with privacy preserving in this example, where People-Health can directly retrieve the desired data from the test centers. Such a capability, besides being more efficient, is vital to handle cases of emergency.

The example runs as follows:

Step 1. A secret credential is issued to Liming by the Citizen Identity Provider where the Gen of Π_{sig} is used. That is to say, the ***Credential Issuing*** subprotocol in the solution is executed between the CIP and Liming.

Step 2. Liming adopts the $\Delta - Gen$ of Π_{sig} to generate the new pseudonym $(P_{Li\min g}, \bar{P}_{Li\min g})$. That is to say, the ***Pseudonym Generation*** subprotocol in the solution is executed by Liming.

Step 3. Liming uses the pseudonym $(P_{Li\min g}, \bar{P}_{Li\min g})$ as his identity to do health examination in the test centers, where he may prove the citizenship by using the $Sign$ of Π_{sig} (this is not illustrated in the figure).

Step 4. The concerned test center stores the test results under the user $(P_{Li\min g}, \bar{P}_{Li\min g})$ because Liming adopted the pseudonym as his identity.

Step 5. Using the $Sign$ of Π_{sig}, Liming signs the warrant m_w that contains $\bar{P}_{Li\min g}$, time period, and the names of

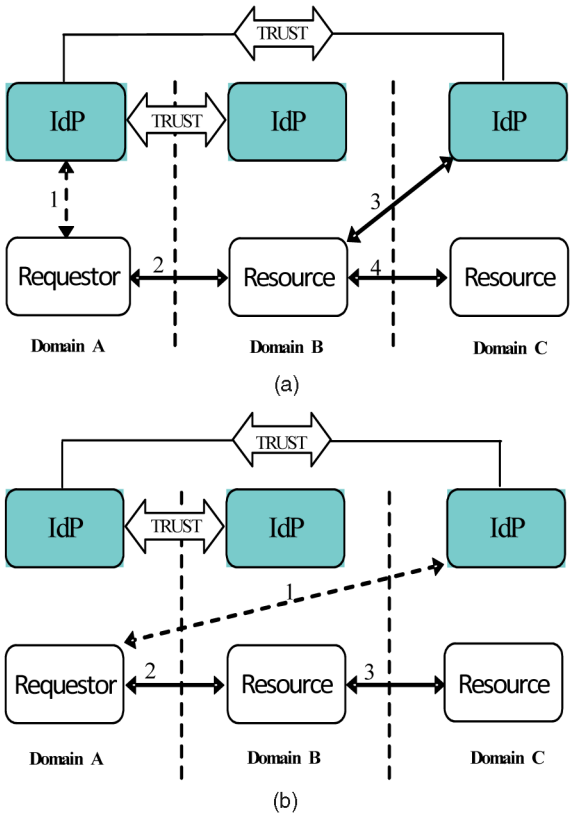

Fig. 3. The universal delegation scenarios. (a) The requestor IdP issues credentials. (b) Target service IdP issues credentials.

those test centers. Then, Liming conveys the signature α and the warrant m_w to People-Health to enable them obtain the proxy key (α, ν). That is to say, the ***Signing-warrant*** subprotocol in the solution is executed between People-Health and Liming.

Step 6. People-Health can directly request the results of health examination from the test centers by signing the request messages under the proxy key (α, ν) according to the algorithm $PSign$ in Π_{psig}.

Step 7. When the request messages signed by People-Health are received, the concerned test center verifies the corresponding signature according to the algorithm $PVerify$ in Π_{psig}.

Step 8. If the signature is valid, the stored results will be conveyed to People-Health.

For general deployment, some typical scenarios are used to illustrate delegation model. In Fig. 3, the requestor has stored some long-term credentials which may be from his IdP or C's IdP. Credentials from the IdP in other domains means that if that IdP issues the requestor a long-term credential in one transaction, it can be used latterly in other transactions for efficiency and convenience. Each arrow represents a possible communication path between the participants. Each dashed arrow represents that the possible communication can be executed offline or has been completed between the participants.

In Fig. 3a, a secret credential is issued to the requestor by the IdP in the requestor's trust realm (Domain A) (see 1). The requestor self-generates identity security tokens (pseudonyms) based on its credential and sends a delegation token to the resource/service in domain B (see 2). The delegation signing tokens are then proved to the IdP in domain C and an access security tokens are returned from C (see 3). Resource B uses the access token to access resource C. In Fig. 3b, a secret credential to access resource C is issued to the requestor from the IdP in domain C (see 1). The requestor self-generates pseudonyms based on its credential and sends a delegation token to the resource/service in domain B (see 2). Resource B uses the delegation signing token access resource C. Unlike the classic delegation process in WS-Federation, no IdP is involved when new identity and delegation tokens are needed if the requestor has stored the credentials, and the privacy of the requestor is protected using unlinkable pseudonyms. In both of the cases, whether IdP B and IdP C trust each other does not matter.

Fig. 4 illustrates the message sequence of our delegation solution in these typical scenarios where a resource accesses data from another resource on behalf of the requestor. In Fig. 4a, the delegation solution runs as follows:

Step 1. A secret credential is issued to the requestor by the IdP in domain A if the requestor has not stored the credential. That is to say, the ***Credential Issuing*** subprotocol in the solution is executed between the IdP and the requestor.

Step 2. The requestor self-generates pseudonyms based on its stored credential using the ***Pseudonym Generation*** subprotocol in the solution. Then, it uses the ***Signing-warrant*** subprotocol to issue delegation token with respect to the self-generated pseudonym.

Step 3. The requestor makes a composite service request to service B.

Step 4. The service B uses the ***Delegation-Signing*** subprotocol to sign an access-token request to the IdP in the domain C.

Step 5. The IdP in the domain C uses the ***Delegation-Verification*** subprotocol to verify the request. If the verification is successfully, it returns an access token to the service in domain B.

Step 6. Service B makes a service request to service C using the access token.

Steps 7 and 8. Service C returns the service response to service B, and service B returns a composite service response to the requestor.

In Fig. 4b, the delegation solution runs on almost the same lines as those of Fig. 4a, which is corresponding to Fig. 3b, and is as follows:

Step 1. A secret credential is issued to the requestor by the IdP in domain C if the requestor has not stored the credential to access resource C and is trusted. That is to say, the ***Credential Issuing*** subprotocol in the solution is executed between the IdP in domain C and the requestor.

Step 2. The requestor self-generates pseudonyms based on its stored credential using the ***Pseudonym Generation*** subprotocol in the solution. Then, it uses the ***Signing-warrant*** subprotocol to issue delegation token with respect to the self-generated pseudonym.

Step 3. The requestor makes a composite service request to service.

Step 4. The service B uses the ***Delegation-Signing*** subprotocol to sign a service request to the resource in the domain C. Because the credentials are issued by the IdP in domain C, service C can directly verify the signature

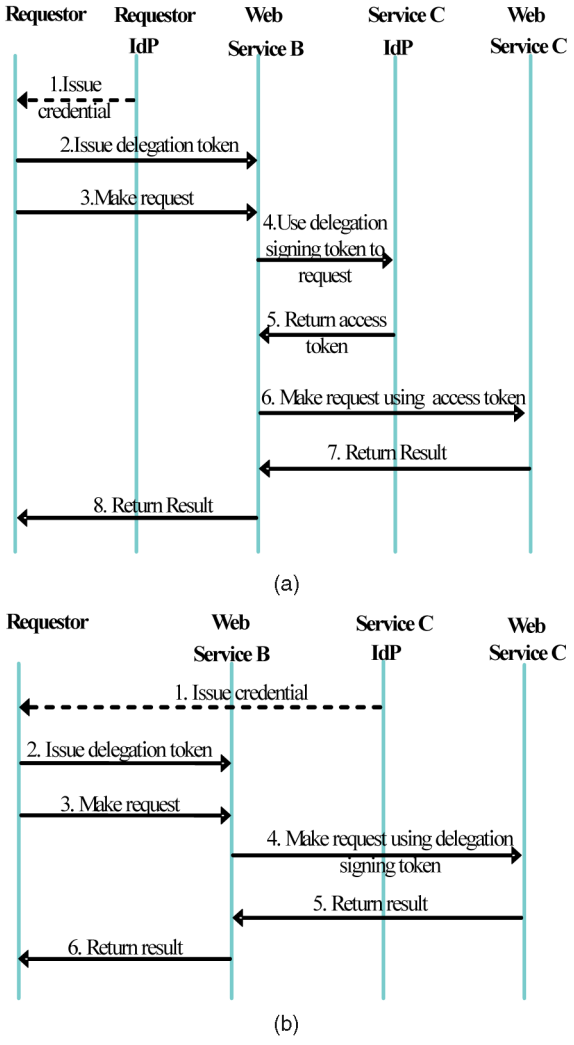

Fig. 4. The universal delegation processed in Web service environments. (a) The requestor IdP issues credentials. (b) Target service IdP issues credentials.

produced by service B, where IdP C may not involve the transactions and service C knows the public keys of its IdP.

Step 5. Service C uses the ***Delegation-Verification*** subprotocol to verify the request. If the verification is successful, it returns a request result to service B.

Step 6. Service B forms composite results and returns them to the requestor.

For a delegation chain $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_n$ where the requestor is denoted by x_1, resource A is denoted by x_2, and so on, the delegation solution works as above by iterating delegation token issuance. Therefore, we describe in detail the delegation token in the chain as follows:

$$\{\alpha, KSet\},$$

where $\alpha = \alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_n$ is the signature and $\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3 \rightarrow \cdots \rightarrow \alpha_n$ is the signature chain for the warrant m_w which are, respectively, produced by $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \cdots \rightarrow x_n$; KSet is a list of pseudonyms, i.e.,

$$KSet = \{PKey_{x_1}, PKey_{x_{n-1}}, PKey = PKey_{x_1} + PKey_{x_3} + PKey_{x_4} + \cdots + PKey_{x_{n-1}}\},$$

where middle pseudonyms can be hidden.

In the above deployment, we use the relationship-focused framework and trust model from WS-Federation. Requestors use stored credentials to reduce interaction rounds, and enhance controllability of identity transactions with self-generating identities and self-issuing credentials.

6 Conclusions

In this paper, a practical delegation solution for universal identity management, as well as a novel notion of pseudonym-based signature scheme, is introduced. In our proposal, the users prove the possession of valid credentials without interacting with identity providers. Beyond user centricity, our delegation solution also allows for privilege delegation to improve the runtime performance of composite services. As the natural certificate chains, the solution consists of two signature schemes: a pseudonym-based signature scheme and a conventional warrant proxy signature scheme. The pseudonym-based signature scheme can be used to self-generate pseudonyms, to prove the possession of credentials, and to achieve privacy based on time-varying pseudonyms. The proxy signature scheme is used to delegate the signing capability where the proxy key includes the signature of warrants produced by the signing protocol in the pseudonym-based signature scheme. Therefore, the model and constructions will provide strong building blocks for the design and implementation of universal identity management systems.

For future work, it is of interest to investigate how to compose delegation credentials for composite services, which are probably issued for different attributes. Making one composite credential from any subset of all credentials will improve system performance and operate conveniently in different scenarios [44].

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No. 60432010, the National Grand Fundamental Research 973 Program of China under Grant No. 2007CB307103, and the National Science Foundation for Postdoctoral Scientists of China under No. 20090450335.

References

[1] K. Cameron, Laws of Identity, http://www.identityblog.com, 2005.
[2] PRIME CONSORTIUM, Privacy and Identity Management for Europe (PRIME), http://www.prime-project.eu, 2010.
[3] Identity-Management, Liberty Alliance Project, http://www.projectliberty.org, 2010.
[4] C. Kaler and A. Nadalin, *Web Services Federation Language*, 2003.
[5] A. Bhargav-Spantzel and J. Camenisch, "User Centricity: A Taxonomy and Open Issues," *Proc. Second ACM Workshop Digital Identity Management (DIM '06)*, pp. 493-527, 2007.
[6] D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete," *Comm. ACM*, vol. 28, no. 10, pp. 1030-1044, 1985.
[7] D. Chaum and J.H. Evertse, "A Secure and Privacy-Protecting Protocol for Transmitting Personal Information between Organizations," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '86)*, pp. 118-167, 1986.
[8] I.B. Damgard, "Payment Systems and Credential Mechanisms with Provable Security against Abuse by Individuals," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '88)*, pp. 328-335, 1988.

[9] L.D. Chen, "Access with Pseudonyms," *Lecture Notes in Computer Science,* pp. 232-243, Springer, 1995.
[10] A. Lysyanskaya, R. Rivest, and A. Sahai, "Pseudonym Systems," *Proc. Sixth Ann. Int'l Workshop Selected Areas in Cryptography (SAC '99),* pp. 184-199, 1999.
[11] J. Camenisch and A. Lysyanskaya, "Efficient Non-Transferable Anonymous Multi-Show Credential System with Optional Anonymity Revocation," *Proc. Advances in Cryptology (EUROCRYPT '01),* B. Pfitzmann, ed., pp. 93-118, 2001.
[12] J. Camenisch and A. Lysyanskaya, "A Signature Scheme with Efficient Protocols," *Proc. Third Conf. Security in Comm. Networks (SCN '02),* pp. 268-289, 2002.
[13] J. Camenisch and A. Lysyanskaya, "Signature Schemes and Anonymous Credentials from Bilinear Maps," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '04),* pp. 56-72, 2004.
[14] M. Belenkiy, M. Chase, and M. Kohlweiss, "Non-Interactive Anonymous Credentials," *Proc. Theoretical Cryptography Conf. (TCC),* http://eprint.iacr.org/2007/384, 2008.
[15] M. Chase and A. Lysyanskaya, "On Signatures of Knowledge," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '06),* C. Dwork, ed., pp. 78-96, 2006.
[16] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, *Delegatable Anonymous Credentials,* http://eprint.iacr.org/2008/428, 2010.
[17] J. Camenisch, D. Sommer, and R. Zimmermann, "A General Certification Framework with Applications to Privacy-Enhancing Certificate Infrastructures," *Proc. IFIP Int'l Federation for Information Processing,* pp. 25-37, 2006.
[18] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures: Delegation of the Power to Sign Messages," *IEICE Trans. Fundamentals,* vol. E79-A, no. 9, pp. 1338-1354, 1996.
[19] S. Kim, S. Park, and D. Won, "Proxy Signatures, Revisited," *Proc. Int'l Conf. Information and Comm. Security (ICICS '97),* pp. 223-232, 1997.
[20] T. Okamoto, M. Tada, and E. Okamoto, "Extended Proxy Signatures for Smart Card," *Proc. Information Security Workshop '99,* pp. 247-258, 1999.
[21] J. Herranz and G. Saez, "Revisiting Fully Distributed Proxy Signature Schemes," *Proc. Int'l Conf. Cryptology in India (Indocrypt '04),* pp. 356-370, 2004.
[22] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '86),* A.M. Odlyzko, ed., pp. 186-194. Aug. 1986.
[23] D. Chaum and E. van Heyst, "Group Signatures," *Proc. Advances in Cryptology (Eurocrypt '91),* D.W. Davies, ed., pp. 257-265, 1991.
[24] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions," *Proc. Advances in Cryptology (Eurocrypt '03),* pp. 614-629, 2003.
[25] D. Boneh and X. Boyen, "Short Signatures without Random Oracles," *Proc. Advances in Cryptology (Eurocrypt '04),* pp. 56-73, 2004.
[26] M. Bellare, H. Shi, and C. Zhang, "Foundations of Group Signatures: The Case of Dynamic Groups," *Proc. Topics in Cryptology (CT-RSA '05),* pp. 136-153, 2005.
[27] C. Delerablee and D. Pointcheval, "Dynamic Fully Anonymous Short Group Signatures," *Proc. Progress in Cryptology (VIETCRYPT '06),* pp. 193-210, 2006.
[28] E. Brickell, J. Camenisch, and L.Q. Chen, "Direct Anonymous Attestation," *Proc. ACM Conf. Computer and Comm. Security,* pp. 132-145, 2004.
[29] J. Camenisch, "Protecting (Anonymous) Credentials with the Trusted Computing Group's Trusted Platform Modules Vo1. 2," *Proc. 21st IFIP Int'l Information Security Conf. (SEC '06),* 2006,
[30] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '01),* vol. 2139, pp. 213-229, 2001.
[31] P. Barreto, H. Kim, B. Bynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '02),* pp. 354-368, 2002.
[32] S. Mitsunari, R. Sakai, and M. Kasahara, "A New Traitor Tracing," *IEICE Trans. Fundamentals,* vol. E85-A, no. 2, pp. 481-484, 2002.
[33] F. Hess, "Efficient Identity Based Signature Schemes Based on Pairings," *Proc. Workshop Selected Areas in Cryptography (SAC '02),* pp. 310-324, 2002.
[34] F. Zhang and K. Kim, "ID-Based Blind Signature and Ring Signature from Pairings," *Proc. Advances in Cryptology (Asiacrypt),* 2002.
[35] X. Huang, Y. Mu, W. Susilo, F. Zhang, and X. Chen, "A Short Proxy Scheme: Efficient Authentication in the Ubiquitous World," *Proc. Embedded and Ubiquitous Computing (EUC) Workshops,* pp. 480-489, 2005.
[36] D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," *J. Cryptology,* vol. 13, no. 3, pp. 361-396, 2000.
[37] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signatures Using Strong Diffie-Hellman," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO),* pp. 41-55, 2004.
[38] R. Canetti, "Universally Composable Signature, Certification, and Authentication," *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW),* pp. 219-245, 2004.
[39] Microsoft, A Technical Reference for InfoCard v1.0 in Windows, 2005.
[40] Higgins Trust Framework, http://www.eclipse.org/higgins, 2006.
[41] J. Camenisch and E.V. Herreweghen, "Design and Implementation of the Idemix Anonymous Credential System," *Proc. Ninth ACM Conf. Computer and Comm. Security,* pp. 21-30, 2002.
[42] J. Camenisch, T. Gross, and D. Sommer, "Enhancing Privacy of Federated Identity Management Protocols," *Proc. Fifth ACM Workshop Privacy in Electronic Soc.,* pp. 67-72, 2006.
[43] IBM, Microsoft, Actional, BEA, Computer Associates, Layer 7, Oblix, Open Network, Ping Identity, Reactivity, and Verisign. Web Services Trust Language (WS-Trust), Feb. 2005.
[44] A. Segev and E. Toch, "Context-Based Matching and Ranking of Web Services for Composition," *IEEE Trans. Service Computing,* vol. 2, no. 3, pp. 210-222, June 2009.

Yang Zhang received the PhD degree in computer applied technology from the Institute of Software, Chinese Academy of Sciences in 2007. His research interests include security and privacy of anonymous systems. His published papers concern anonymous routing protocols, anonymous authentication protocols, design and implementation of anonymous systems, and pseudonym systems. He currently works in the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His team makes scientific research on Mobile Service Platform (National Natural Science Foundation of China under Grant No. 60432010).

Jun-Liang Chen graduated from the Department of Telecommunications, Shanghai Jiaotung University in 1955 and received the Doctor of Engineering degree from the Moscow Institute of Electrical Telecommunications, former Soviet Union, in 1961. He was a visiting scholar in the University of California at Berkeley and also at Los Angeles from 1979 to 1981. He has been working in the Beijing University of Posts and Telecommunications (BUPT) since 1955. Currently, he is a professor and director of the Research Institute of Networking Technologies of BUPT. He is both a member of the Chinese Academy of Sciences and the Chinese Academy of Engineering. He has published a book and more than 250 papers. His current research interests include network intelligence and services, switching technology, communications software, and fault-tolerant computing.