SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment*

Sherman S.M. Chow[1], Yi-Jun He[2,**], Lucas C.K. Hui[2], and Siu Ming Yiu[2]

[1] University of Waterloo, Ontario, Canada
smchow@math.uwaterloo.ca
[2] University of Hong Kong, Hong Kong
{yjhe,hui,smyiu}@cs.hku.hk

Abstract. Identity security and privacy have been regarded as one of the top seven cloud security threats. There are a few identity management solutions proposed recently trying to tackle these problems. However, none of these can satisfy all desirable properties. In particular, *unlinkability* ensures that none of the cloud service providers (CSPs), even if they collude, can link the transactions of the same user. On the other hand, *delegatable authentication* is unique to the cloud platform, in which several CSPs may join together to provide a packaged service, with one of them being the source provider which interacts with the clients and performs authentication while the others will be transparent to the clients. Note that CSPs may have different authentication mechanisms that rely on different attributes. Moreover, each CSP is limited to see only the attributes that it concerns.

This paper presents SPICE – the first digital identity management system that can satisfy these properties in addition to other desirable properties. The novelty of our scheme stems from combining and exploiting two group signatures so that we can randomize the signature to make the same signature look different for multiple uses of it and hide some parts of the messages which are not the concerns of the CSP. Our scheme is quite applicable to cloud systems due to its simplicity and efficiency.

Keywords: Cloud Computing, Digital Identity Management, Interoperability, Delegation, Privacy, Unlinkability.

1 Introduction

Cloud computing is a new computing platform where thousands of users around the world can have network access of a shared pool of computing resources

* The work described in this paper was partially supported by the General Research Fund from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. RGC GRF HKU 713009E), the NSFC/RGC Joint Research Scheme (Project No. N_HKU 722/09), HKU Seed Fundings for Applied Research 201102160014, and HKU Seed Fundings for Basic Research 201011159162 and 200911159149.

** Corresponding author.

F. Bao, P. Samarati, and J. Zhou (Eds.): ACNS 2012, LNCS 7341, pp. 526–543, 2012.

(e.g., storage, applications, services, *etc.*) with minimal requirement for the users' computers, and minimal management effort from the service provider. Various popular cloud computing services have emerged including Amazon's Simple Storage Service (S3), Box.net, CloudSafe and Internap XIPCloud Storage. On the other hand, there are quite a number of security and privacy concerns in this computing platform that are yet to be resolved [20,21]. In particular, user identity security was regarded as one of the top seven cloud security threats by the Cloud Security Alliance in 2010 [3].

1.1 Extended DIM Framework in the Cloud

Existing solutions for tackling the identity security problem in the cloud are usually based on the framework of a digital identity management (DIM) system [4,6,19], which allows the cloud service providers to provide services only to authenticated users. A typical DIM system in a cloud environment consists of four components: *cloud service providers* (CSPs), *identity providers* (IdPs), *registrars*, and *users* (cloud clients). IdPs are responsible for assigning attributes to users. Registrars are able to verify the attributes given by an IdP to a user, and then issue a certificate to the user. Based on these certificates, users can authenticate themselves to CSPs and gain access to the authorized services.

In an *extended* DIM framework depicted in Figure 1, CSPs may not work alone as independent providers. Several CSPs may join together to provide a packaged service to the clients. One of them, called the *source CSP*, acts as the interface to the clients while the others, called the *receiving CSPs*, can be transparent to the clients and provide services in the back-end. A CSP can be both a source and a receiving CSP, and a receiving CSP can also interact with different source CSPs providing different services.

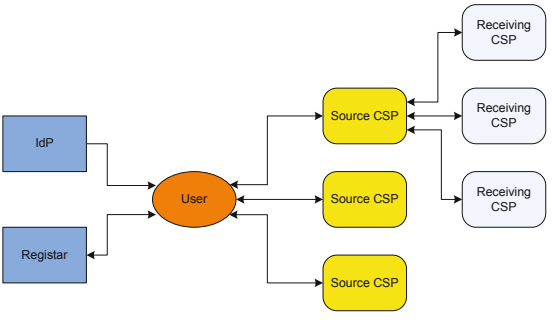

Fig. 1. Extended DIM Structure

1.2 Desirable Properties for DIM in the Cloud

A DIM system should have the following desirable security/privacy and functional properties for authentication.

- **Unlinkability**. In cloud computing, a user may purchase multiple services associated with the same or different CSPs. Unlinkability ensures that no CSPs, even if they collude, can link different transactions, whether they are of the same service or different services, of the same user.
- **Delegatable Authentication**. Each CSP would like to authenticate the user in its own way before providing their packaged services. The authentication should be *delegatable* such that the receiving CSP can authenticate a user without a direct communication with either the user[1] or the registrar, and without fully trusting the source CSP.[2]
- **Anonymity.** The users should be able to anonymously authenticate themselves, as authorized users to the CSP, without letting the CSP know about their real identity or exact attributes.
- **Accountability**. The users may abuse their anonymity. If needed, a trusted party can revoke the anonymity so the users can be held accountable for their malicious action.
- **User Centric Access Control.** Users should be able to control what information they want to reveal about themselves over the cloud or to a CSP, and to control who can access that information, and how this information would be used in order to minimize the risk of identity theft and fraud.
- **Single registration**. The users only need to register themselves once for getting the credentials without the need of contacting the registrar every time authentication is needed.

1.3 Existing Solution and Limitations

In Bertino *et al.* [6]'s DIM system, the registrar stores a set of signatures, each signing on a commitment of a user's attribute. To authenticate to a CSP, the user first retrieves from the registrar a set of signatures on the attributes that the CSP requires (matching of names may be needed to determine the naming of the attributes). Then, the user executes a zero-knowledge proof (ZKP) protocol to prove to the CSP that the signatures presented by the user signs on the committed attributes. Finally, a new credential will be generated to this user by the CSP upon its verification of the registrar's signatures. This credential can be presented to another CSP by the user to show that there was a CSP which verified the required set of attributes.

Different CSPs may require a different set of attributes for authentication, so many different credentials will be issued. In their scheme, the registrar should remain (practically always) online to return different signature sets to a user whenever the user subscribes to new services. The number of times that the

[1] It is more convenient to the users if all receiving CSPs are transparent. On the other hand, the source CSP may want to keep the operation private and may not be willing to let the users know the CSPs which it collaborated for providing indirect services.

[2] The receiving CSP cannot fully trust the source CSP since it is also regarded as a big client to the receiving CSP. E.g., PayPal (providing a payment gateway between the merchant and the customer) does not rely on eBay (providing shopping "service") for authenticating consumers although they are partners.

registrar responds to a user is linear to the total number of services that the user wants to invoke. Apart from the scalability issue, this also increases the risk of exposing the "master secret key" (used by the registrar to certify attributes) from a variety of network attacks.

The second non-negligible consequence is the linkability issue that the attacker may link registrar's signatures of the same user to compromise the privacy of that user. Even though their authentication mechanism uses zero-knowledge proof of knowledge (ZKPoK) of a "correct"[3] attribute (that is contained in the commitment signed by the registrar) to avoid showing the attributes in clear, the user simply shows a signature (which is an aggregation of the signatures given by the registrar) on the commitments of the attributes, which makes different signatures easily linkable. One may wonder if we can just also use ZKPoK to hide the signature. Looking ahead, our solution leverages the re-randomizability of the signature, which is a very lightweight approach when compared with ZKPoK.

The third consequence is that it is hard to delegate authentications (not to say making the delegatable authentications unlinkable) based on this framework. We want that a source CSP who just verified the credential of a user can convince a receiving CSP that the user in question indeed has (more than) what the receiving CSP expects, without requiring the receiving CSP to directly interact with the registrar or the user. This feature is termed as delegatability in Bertino *et al.*'s work and was remarked as a future research direction. Finally, we remark that simultaneously addressing all the desirable requirements is more involved as they may conflict with each other.

There is an open source reference implementation called Shibboleth [1], which can be regarded as using a similar (in a high-level way) authentication mechanism. Their single sign-on feature allows service providers (SPs) in a federation, which must be explicitly configured *a priori*, to provide a packaged service to a user, which seems to be the delegatable authentication feature. However, the user needs to authorize a source SP to access a receiving SP's service on behalf, so the process is *not transparent*. Most importantly, their IdP (which plays the role of registrar in the terminologies in [6] and in this paper) needs to be online during the access and two certificates will be generated for both SPs. Unlinkability is also not a concern in Shibboleth.

There are other related works which try to provide extensions of the basic notion such as [4,11,16,19]. However, up to now, it is fair to say there still does not exist any scheme that can satisfy both the property of *unlinkability* and *delegatable authentication*, in an efficient yet cryptographically secure manner.

1.4 Our Contributions

We present SPICE – a DIM system in the cloud environment that can satisfy all the aforementioned six desirable properties for identity management, which is the first such system to our knowledge. In our scheme, the registrar issues only

[3] It is not clear that how the CSP can verify if an attribute is correct if it is hidden by the commitment and the zero-knowledge proof.

one credential to each user no matter how many services that the user subscribes or will subscribe later. For authentication, the user generates from the credential many different versions of certificates for proving the possession of (different sets of) attributes required by different service providers, without asking registrar for issuing new certificates each time.

The novelty of our proposed system stems from extending two group signature schemes [8,9] to their "full" potential with the help of the nice properties of Waters' signature [22] and Groth-Sahai proof system [15], both are implicitly used as building blocks in [8,9]. Group signatures are privacy-oriented signatures where a group manager can issue signing keys to many group members who in turn can sign on behalf of the group. Anyone can be convinced that the signature is indeed from the group, but not exactly who.

To satisfy the unlinkability property, we apply randomization to the signatures (which acts as the certificates in our DIM system) to make them look different and hence unlinkable for multiple requests. Both [15] and [22] are known to be re-randomizable (see Section 4 for the technical details). On the other hand, to control which attributes can be revealed to which CSPs, we want the messages being signed (correspond to the attributes in our DIM system) to be partitioned into different blocks. We thus need to extend the message space from a single message to many blocks of messages. As a result, some of the attributes can be hidden from the CSPs if they are not related to CSPs' verification. Finally, for delegated authentication, some of the attributes certified by a group signature can be hidden by the source CSP by a sanitization process.

The idea of using sanitizable and re-randomizable group signature for a privacy-preserving DIM is conceptually simple, and the run-time performance of our prototype is practically efficient, hence we believe our system is applicable in cloud environment nowadays.

2 Related Work

2.1 Identity Management Systems

After Bertino *et al.*'s work, Celesti *et al.* [11] constructed an InterCloud Identity Management Infrastructure focusing on the InterCloud scenario where clouds cooperate with other federated ones with the purpose to enlarge their computing and storage capabilities. In InterCloud, the users' home cloud should be able to perform a single sign-on in order to gain access to the resources offered by another cloud that participates in an InterCloud formation; each home cloud should be able to authenticate itself with foreign clouds using its digital identity guaranteed by a third party.

The PhD thesis of Hussain [16] studied secure anonymous authentication with Personas. Personas are defined as sets of statements, where each statement asserts the status of an individual's set of attributes. The main advantage of [16] is that the Personas of an individual are distributed among a number of servers which mitigate the damage of a single server compromise. However, they did not pick up the challenge to support unlinkability and delegation.

Angin *et al.* proposed an entity-centric approach for privacy and identity management in cloud computing [4]. It aims at achieving three requirements: (i) authenticating users without disclosing personal identity information (PII); (ii) ability to provide identity management services on untrusted hosts, such as public hosts; and (iii) not using trusted third parties. Their approach is based on an entity-centric DIM which uses active bundles scheme to protect PII from untrusted hosts; and anonymous identification which involves using zero-knowledge proofs for authentication of an entity without disclosing its identifier. However, the proposed approach could have a large communication overhead since each anonymous identification would involve a good number of communications between a CSP and the user. Further, an active bundle (a token) which is a container with a payload of sensitive data, metadata, and a virtual machine (VM), needs to be passed between a CSP and a user during a communication session, but no details mention about how to pass the token effectively. The token could have a large size. So, we are not certain about the practicality of the proposed approach.

Ranchal *et al.* [19] proposed another scheme to address the same privacy problem [4] by using the active bundles scheme and computing predicate over encrypted data for giving answer about PII. Their approach uses identity data on untrusted hosts without TTP. However, when users asking for a service from the CSP, the overhead is high since they need to compute a token for a CSP using secure multiparty computation involving the cooperation with a number of parties. Also, they did not mention how to pass effectively a big active bundle between a user and a CSP.

2.2 Other Credential-Based Authentication Systems

A group signature is essentially a ZKPoK of the signatures. Indeed, that is exactly the mechanism behind most cryptographic anonymous credentials. But many such systems often have their own specific features (e.g., ensuring a credential can only be used for at most k times [12]) and do not perfectly fit with our needs.

One may consider asking the user to "delegate" a credential to a source CSP using "delegatable anonymous credential", then the source CSP can show the credential to a receiving CSP on the user's behalf. Not every anonymous credential systems come with this property. A recent system (e.g., [5]) supports delegation of the credential itself, but it is different from delegating the verification process. In more details, the credential system in [5] requires that each user to have their own private key for exculpability reason, which is not a concern in our scenario but results in a much more complicated signing mechanism (an interactive protocol for obtaining a signature on a message hidden in the commitment) and the corresponding proof (the private key corresponding to the message being signed at the i-th level is used to sign the message at the $(i+1)$-th level). Besides, their application does not require any anonymity revocation. We stressed that their scheme is more on delegating the credential (i.e. the signing process) than delegating the verification. To conclude, delegatable anonymous

credentials are too heavyweight for our usage on one hand, but still lack the anonymity management mechanism required in our scenario.

Finally, we remark that there are extensions of the basic group signature idea in terms of anonymity management mechanism, such as tracing and signature claiming or denial considered in traceable signature [2,12].

3 Overview of SPICE

3.1 Framework of SPICE

We will describe the main entities in our system, and different phases (corresponding to different tasks) of a typical run-time scenario. We will list the actions performed by different entities, the objects created, and how these objects flow between them.

In our framework, the registrar will first create a user-specific credential, which is then used to generate a *source certificate* certifying a number of attributes. That is the *user enrollment* phase.

In the *authentication phase*, a user uses the source certificate obtained from the registrar to create an *authentication certificate*, or simply a *certificate* according to the set of attributes expected by a CSP. Upon successful *verification*, the CSP either provides the service to the user, or contacts other receiving CSP(s) for help in providing a packaged service.

In order to perform *delegatable authentication*, the CSP makes use of the certificate obtained to generate a *"new" certificate* depending on the set of attributes expected by a receiving CSP.

3.2 Framework of Basic Group Signatures

A group signature scheme, GS, consists of five algorithms. The first two and the last one are used by the group manager (GM), the third is used by group members (U), and the fourth one is used by any verifiers.

- $\mathsf{GS.Setup} : (\mathsf{gpk}, \mathsf{gsk}, \mathsf{gok}) \leftarrow \mathsf{GS.Setup}(1^\kappa)$ is an algorithm that generates a group public (verification) key gpk, certificate issuing key gsk and opening-key gok.
- $\mathsf{GS.UG} : \mathsf{sk} \leftarrow \mathsf{GS.UG}_{\mathsf{gpk}}(\mathsf{gsk}, \mathsf{id})$ is an algorithm that generates a user signing key sk to the user having identity id.
- $\mathsf{GS.Sign} : \sigma \leftarrow \mathsf{GS.Sign}_{\mathsf{gpk}}(\mathsf{sk}, M)$ is a probabilistic algorithm that generates a signature σ for message M by using signing-key sk.
- $\mathsf{GS.Ver} : 1/0 \leftarrow \mathsf{GS.Ver}_{\mathsf{gpk}}(\sigma, M)$ is a deterministic algorithm that decides the correctness of signature σ on M. It outputs 1 for any input created through $\mathsf{GS.Setup}$, $\mathsf{GS.UG}$ and $\mathsf{GS.Sign}$, 0 otherwise.
- $\mathsf{GS.Open} : \mathsf{id}/\perp \leftarrow \mathsf{GS.Open}_{\mathsf{gpk}}(\mathsf{gok}, \sigma)$ is an algorithm that identifies the signer of a valid signature σ by using the opening-key gok.

3.3 The Key Ideas

The authentication mechanism in SPICE will be instantiated by a group signature scheme with special properties denoted by GS. The registrar will act as the group manager in GS. As revealed previously in Section 1.4, there are multiple attributes associated with a source certificate (which is a signature on $(\ell - 1)$ attributes given by member signing key in GS). These attributes (hosted in the message blocks associated with GS) will be accompanied by certificate created by the credential under the normal circumstance.

For accessing a service, the user creates a certificate by the signing algorithm of GS, signing on one more block of message, which is the session's information such as the time of access, the description of the service, *etc.* When needs arise, the opening mechanism of GS can be used by the registrar to revoke the anonymity of a certificate. From now on, we will use the terminologies in the context of GS.

One of the key concepts in our scheme is randomization of the group signatures. It is either applied on the signature so that multiple requests issued by using the same signature are unlinkable, or applied on the message-component of the signature, such that the attribute being signed can be hidden. In order to design such a group signature scheme, we combine and extend two group signature schemes from [8,9]. First, we extend [9] by adding two algorithms: randomization algorithm (GS.Rand), and hiding algorithm (GS.Hide) to make the signature re-randomizable and some blocks of the signatures can be hidden (sanitized). Second, we extend the idea of two-level hierarchical signature in [8] to make our group signature feasible to be signed on blocks of messages instead of a single message.

Authentication is done by presenting a randomized copy of the certificate in general, as depicted in Figure 2. Take a user A with only three attributes as a simple example. User A's certificate is a signature of attributes A_1, A_2 and A_3. Every time A randomizes the signature before authenticating himself to CSPs. We use dark color to represent the original signature, and light color for the randomized signature. The re-randomizable property prevents the certificates from the same user from being linked.

SPICE classifies attributes into the following types for possible signature randomization and sanitization.

I) Sensitive personal information. This class of attributes, depicted by A_1 in Figure 2 is relatively stable and has a common representation across different CSPs. There is no special treatment for this class of attributes in our scheme.[4]

II) Service-specific attributes. Some attributes may be of interest to CSPs who are providing a similar kind of services (e.g., whether HTML or attachment is allowed in out-going e-mail) or providing a collective / coupled service (e.g., geographic information for various social networks or other location-based services).

[4] We defer to Section 5.2 for a possible strengthening of privacy concern when getting a credential from the registrar.

This class is depicted by A_2 in Figure 2. The CSPs may share similar authentication policy, but they may employ heterogeneous naming when establishing their authentication policies.

A nice feature considered in Bertino *et al.*'s work is *naming heterogeneity*, i.e., the existence of variations between attributes associated with the user's credential and attributes specified in the policies required by different CSPs. They used a matching technique based on look-up tables, dictionaries (WordNet 2.1 English Lexical Database[5]), and ontology mapping (Falcon-AO[6]) to resolve syntactic (e.g., "ID" vs. "Identity"), terminological (e.g., "email address" vs. "email account") and semantic (e.g., "privacy level" vs. "sharing setting") variations. SPICE will also use the same tools.

This class of attributes will be certified by a group signature, comes from another instance of GS, i.e., instead of the group of cloud clients, now we consider a group of CSPs. Source CSPs and receiving CSPs are group members. Being another instance of GS, the group signature generated is supposed to be certificates on the class of attributes that may need naming resolution. In particular, they cannot be used as typical user certificates. Whenever there is a variation of attributes representing the same concept, the source CSP generates a group signature certifying the same attribute with a different naming. These signatures may be sent to the client or stored by the source CSP for future usage (and re-randomization can also be done to ensure the unlinkability). In this way, the only thing that the registrar needs to know is who are concerned with a selected subset of attributes, instead of knowing the ontology mapping between all these CSPs, which has been offloaded to the CSPs effectively. The operation is depicted in Figure 2(b).

We remark that we just take a simple approach of appending another group signature to the original one. Updating the attributes associated with a cryptographic key is a tricky problem (except a recent work [10] that works with a particular group signature scheme [7]), which hinders the mismatch resolution considered by Bertino *et al*, and especially difficult when we also want to satisfy unlinkability.

III) Irrelevant attributes. There are many other attributes, which are unrelated to a specific (class) of services in question, depicted by A_3 in Figure 2. On the other hand, even when the CSPs are sharing authentication information, it is likely that some of the attributes are irrelevant for some of these CSPs (e.g., when sharing geographic information of a user logged into a social network, it is inappropriate to share other information such as the posting privilege of the user in the social network to other location-based services).

The privacy of this class of attributes can be taken care of by the hiding algorithm of our group signature scheme. The operation depicted in Figure 2(c) is hiding attribute A_3 which is not required by the receiving CSP, and Figure 2(d) is re-randomizing a sanitized signature.

[5] `http://wordnet.princeton.edu`

[6] `http://iws.seu.edu.cn/projects/matching`

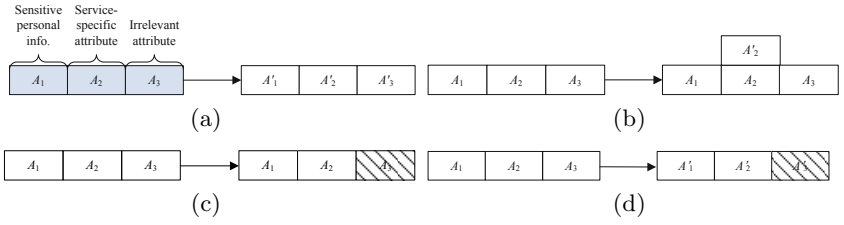

Fig. 2. (a) Re-randomization (b) Naming conversion (c) Attribute hiding (d) Hide-and-randomize

4 Randomizable Group Signatures

This section provides the details of instantiating the group signature scheme GS, which is the main building block in SPICE as described in previous section.

4.1 Design of Randomizable Group Signatures

Recall that hierarchical signatures with non-interactive zero knowledge (NIZK) proofs can be used to construct group signatures [8]. Two schemes given by Boyen and Waters [8,9] fall into this paradigm, with Groth-Sahai proofs [15] as the NIZK proofs. The basic idea is that, the message at the first level of the hierarchy will serve as a group member's identity, and the lower level corresponds to the message that a group member is going to sign. To hide the identity, an (extractable) NIZK proof is used to hide the message at the first level.

We use a group signature scheme which can be seen as a combination of these two schemes [8,9]. Indeed, one can just use the scheme from [8] which uses the hash functions of the same form (instantiated with different parameters) for both the group members' identities and the actual messages to be signed. As [9], the scheme to be presented here is a hybrid of two schemes, one at each level. For our application, we added three extra properties.

1. We want the group signatures can be re-randomizable by a public algorithm, so we add the algorithm GS.Rand. For this we require two properties from our building blocks. Recall that the group signature here is just an NIZK proof of a "regular" signature. First, the implicit regular signatures should be re-randomizable. This property can be satisfied by Waters' signatures – the signing mechanism used for signing the actual messages. Second, the NIZK proofs should be re-randomizable. The Groth-Sahai proof system [15] being used here is known to be re-randomizable (e.g., [5,14]), hence re-randomization of the group signatures can be done.
2. We want the group signatures to be able to sign blocks of messages instead of a single message. For this we rely on the fact that one may add a hierarchy of messages to be signed by extending the public key with a new set of parameters for the Waters-hashes [22] for other levels, as illustrated in the

two-level hierarchical signatures schemes in [8]. This technique has been utilized in other applications such as [13,18].

3. For multiple blocks of message, it is desirable if some blocks can be hidden from the verifiers when they only concern the other blocks. For this we introduce another algorithm GS.Hide which transforms a given group signature into one with some selected blocks hidden, in the same spirit as how the identity at the first level of the hierarchy is being hidden.

4.2 Concrete Construction

Below we supply the details of GS. In our DIM structure, registrar takes the role of the group manager (GM), users and the CSPs are group members (U).

Basic Algebraic Settings. Let $(\mathbb{G} = \langle g \rangle, \mathbb{G}_T)$ be a bilinear group pair of composite order $n = pq$ where p and q are random primes of bit size at least κ. Let $\mathbb{G}_q = \langle h \rangle$ be the cyclic subgroup of $\mathbb{G}$ of order q.

GS.Setup. This algorithm is used by the registrar. It generates the bilinear groups parameter listed above. The opening-key is $\mathsf{gok} = q$. Then, it picks two random exponents $\alpha, \omega \in \mathbb{Z}_n$, defines $\hat{A} = \hat{e}(g,g)^{\alpha}$ and $\Omega = g^{\omega}$. It also picks a random generator $u \in \mathbb{G}$. The certificate issuing key is $\mathsf{gsk} = \langle g^{\alpha}, \omega \rangle$.

Suppose the number of message blocks to be supported is ℓ, where each of them is m bit. It picks ℓ vectors, each vector $\boldsymbol{v}_j$ consists of $(m+1)$ random generators, $v_j, v_{j,1}, \cdots, v_{j,m} \in \mathbb{G}$. Define $F_j(M_j) = v_j \prod v_{j,i}^{\mu_{j,i}}$ where $M_j = \mu_{j,1} \cdots \mu_{j,m}$. The group public key is $\mathsf{gpk} = \langle \mathbb{G}, \mathbb{G}_T, n, g, u, \Omega, h, \hat{A}, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_{\ell} \rangle$.

GS.UG. This algorithm is used by the registrar to generate signing keys for group members (users and CSPs). The system supports 2^k members and each of their identities are mapped to integer id where $0 \leq \mathsf{id} < 2^k < p$, a unique value $s_{\mathsf{id}} \in \mathbb{Z}_n$ is assigned to each member and will be stored for later opening purpose. This value must be chosen so that $(\omega + s_{\mathsf{id}})$ lies in $\mathbb{Z}_n^{\times}$, the multiplicative group modulo n. The member signing key $\mathsf{sk}_{\mathsf{id}}$ is in the form of $\langle K_1, K_2, K_3 \rangle = \langle (g^{\alpha})^{\frac{1}{\omega + s_{\mathsf{id}}}}, g^{s_{\mathsf{id}}}, u^{s_{\mathsf{id}}} \rangle$, which is of the same form as [9].

GS.Sign. This algorithm has three usages. When it is used by the registrar to generate a source certificate, it will sign on $(\ell - 1)$ attributes. When it is used by a user, it will sign on the information for the authentication session (by appending the source certificate with a last block of message, i.e. the ℓ-th block). If used by a CSP, it will generate a sanitized certificate for solving naming conflict.

I) For signing on blocks of message $\{M_j\}_{j=1}^{\ell-1}$ where $M_j = \mu_{j,1} \cdots \mu_{j,m}$, it first picks $r_1, \cdots, r_{\ell-1} \in \mathbb{Z}_n$ and creates an ordinary signature by

$$\begin{aligned} \theta &= \langle \theta_1, \theta_2, \theta_3, \{\theta_{j'}\}_{j'=4}^{\ell+2}, \{\theta'_{j'}\}_{j'=4}^{\ell+2} \rangle \\ &= \langle K_1, K_2, K_3 \prod_{j=1}^{\ell-1} F_j(M_j)^{r_j}, \{g^{-r_j}\}, \{h^{r_j}\} \rangle. \end{aligned}$$

This step extends [9] in two ways. First, instead of signing on a single message M_1, a product of $F_j(M_j)^{r_j}$'s is attached to sign on multiple messages, similar to the 2-level structure of [8]. Second, h^{r_j} is also attached for future hiding of the component $F_j(M_j)^{r_j}$, similar to how the "first-level" message is being hidden in [8].

Then, it turns θ into a group signature with s_{id} hidden by randomly picking $t_1, t_2, t_3, \{t_{j'}\}_{j'=4}^{\ell+2} \in \mathbb{Z}_n$, computing $\{\sigma_j = \theta_j \cdot h^{t_j}\}$ for $j = 1, \cdots, \ell+2$ and generating the corresponding proofs.

$$\pi_1 = h^{t_1 t_2} \cdot (\theta_1)^{t_2} \cdot (\theta_2 \Omega)^{t_1},$$
$$\pi_2 = u^{t_2} \cdot g^{-t_3} \cdot (\prod_{j=1}^{\ell-1} (F_j(M_j))^{-t_{j+3}})$$

This is the exact same step in [9], apart from the fact that a product of $F_j(M_j)^{r_j}$'s is attached.

The final signature is $\langle \{\sigma_j\}_{j=1}^{\ell+2}, \{\theta'_{j'}\}_{j'=4}^{\ell+2}, \pi_1, \pi_2 \rangle$.

II) When given a source certificate / signature in the above form signed on $(\ell-1)$ message blocks, one can generate a certificate / signature on ℓ blocks by first appending the last message M_ℓ following the format of θ, then followed by a re-randomization using GS.Rand to be described. Due to the lack of space we omit an explicit description of this procedure, but that can be easily deduced from the above description and follows a similar signing procedure in the hierarchical signature schemes [22,8] based on Waters' signature.

GS.Ver. This algorithm is used to verify a source certificate. The verifier outputs 1 if and only if both of the following equations hold, 0 otherwise.

$$\hat{e}(h, \pi_1) = \hat{A}^{-1} \cdot \hat{e}(\sigma_1, \sigma_2 \Omega),$$
$$\hat{e}(h, \pi_2) = \hat{e}(\sigma_2, u)\hat{e}(\sigma_3, g)^{-1} \prod_{j=1}^{\ell} (\hat{e}(\sigma_{j+3}, F_j(M_j))^{-1})$$

(One may defer the checking of $\{\theta'_{j'}\}_{j'=4}^{\ell+2}$ from GS.Ver to GS.Hide since they are used when the $(j'-3)$-th message block $M_{j'-3}$ should be hidden.)

GS.Open. This algorithm is used by the registrar to ensure accountability. Given a valid signature, this algorithm takes out its σ_2 component, and tests whether $(\sigma_2)^q = (g^{s_{\mathsf{id}_i}})^q$ for each suspected id_i. We note that $(g^{s_{\mathsf{id}_i}})^q$ can be precomputed and stored when each user secret key is generated, as suggested in [9].

GS.Hide$(\sigma, M_j, j) \rightarrow \sigma'$. This algorithm is used by users and source CSPs to hide some attributes of the certificate. Given a valid signature σ that may have been re-randomized and may have some blocks already hidden, one may hide the j-th block of message $M_j = \mu_{j,1} \cdots \mu_{j,m}$ from being required in GS.Ver as follows. For simplicity the below description assumes none of the blocks have

been hidden, but it can be easily seen that the hiding action of the j-th block is independent of whether any other block $j' \neq j$ was hidden or not.

1. Abort if $\hat{e}(\sigma_{j+3}, h) \cdot \hat{e}(g, \theta'_{j+3}) \neq 1$.
2. Randomly pick $\tau_1, \cdots, \tau_m \in \mathbb{Z}_n$.
3. Create the commitments $c_{j,k} = v_{j,k}^{\mu_{j,k}} \cdot h^{\tau_k}$ for $k = 1, \cdots, m$.
4. Construct the proofs $\pi_{j,k} = (v_{j,k}^{2\mu_{j,k}-1} \cdot h^{\tau_k})^{\tau_k}$ for $k = 1, \cdots, m$.
5. Define $\tau = \sum_{k=1}^{m} \tau_k$.
6. Set $c_j = v_j \prod_{k=1}^{m} c_{j,k} = F_j(M_j) \cdot h^{\tau}$.
7. Randomly pick $\tilde{r} \in \mathbb{Z}_n$.
8. Compute $\sigma'_3 = \sigma_3 \cdot (\theta'_j)^{\tau} \cdot (c_j)^{\tilde{r}}$.
9. Compute $\sigma'_{j+3} = \sigma_{j+3} \cdot g^{-\tilde{r}}$.
10. Output $\mathsf{GS.Rand}(\langle \{\sigma_1, \sigma_2, \sigma'_3, \sigma_4, \cdots, \sigma_{j+2}, \sigma'_{j+3}, \cdots, \sigma_{\ell+3}, \{\theta'_{j'}\}_{j'=4}^{\ell+2}, \pi_1, \pi_2, \{c_{j,k}\}, \{\pi_{j,k}\}\rangle)$.

In essence, that is the conceptual step (for hiding the group member's identity) in [8] which changes a regular signature to a proof of signature on a hidden message. Broadly speaking, that is the technique from Groth-Sahai system [15].

GS.Rand$(\sigma) \rightarrow \sigma'$. This algorithm is used by user to randomize the certificate in order to provide certificate unlinkability. Given a valid signature σ, anyone can output its randomized version σ' as follows. For simplicity, we first shown how to randomize when σ has no hidden message component.

1. First randomly pick $t'_1, t'_2, t'_3, \{t'_{j'}\}_{j'=4}^{\ell+2} \in \mathbb{Z}_n$.
2. Randomize the commitment parts of the signature by computing $\sigma'_j = \sigma_j \cdot h^{t'_j}$.
3. Then, for the proof component π_1, compute $\pi'_1 = \pi_1 \cdot \sigma_1^{t'_2} \cdot \sigma_2^{t'_1} \cdot h^{t'_1 t'_2} \cdot \Omega^{t'_1}$.
4. Update the corresponding proof $\pi'_2 = \pi_2 \cdot u^{t'_2} \cdot g^{-t'_3} \cdot (\prod_{j=1}^{\ell} (F_j(M_j))^{-t'_{j+3}})$.

Now the signature with message block hidden can also be randomized in a similar manner, by returning $\pi'_{j,k} = \pi_{j,k} \cdot c_{j,k}^{2\tau'_k} \cdot v_{j,k}^{-\tau'_k} \cdot h^{\tau'_k \tau'_k}$ for randomly picked $\tau'_k \in \mathbb{Z}_n$. The commitment can be updated correspondingly by $c'_{j,k} = c_{j,k} \cdot h^{\tau'_k}$. However, the proof π'_2 should be updated in a slightly different manner. For each hidden message block j, $F'_j(\cdot) = v_j \prod_{k=1}^{m} c_{j,k}$ (which is a constant function) should be used instead of $F_j(M_j)$.

Essentially, this algorithm just re-randomizes the commitments and the proofs using the re-randomization procedures [5,14] of the Groth-Sahai systems [15].

Finally, it is trivial to re-randomize $\{\theta'_{j'}\}_{j'=4}^{\ell+3}$ and hence they are omitted.

GS.Ver'. This algorithm is used to verify a randomized and sanitized certificate.

1. For all hidden message block j:
   (a) Check whether the proofs are correct, abort if $\hat{e}(c_{j,k}, c_{j,k} \cdot v_{j,k}^{-1}) = \hat{e}(h, \pi_{j,k})$.
   (b) Define $F'_j(\cdot) = v_j \prod_{k=1}^{m} c_{j,k}$ (which is a constant function).
2. Define $F'_j(M_j) = F_j(M_j)$ for $F'_j(\cdot)$ which has not been defined previously.

3. Output 1 if and only if both of the following equations hold, 0 otherwise.

$$\hat{e}(h, \pi_1) = \hat{A}^{-1} \cdot \hat{e}(\sigma_1, \sigma_2 \Omega)$$

$$\hat{e}(h, \pi_2) = \hat{e}(\sigma_2, u)\hat{e}(\sigma_3, g)^{-1} \prod_{j=1}^{\ell} (\hat{e}(\sigma_{j+3}, F'_j(M_j))^{-1})$$

The hybrid definition of $F'_j(\cdot)$ just corresponds to the hybrid signing methods in [8] – one on a known message and another on a hidden message.

The scheme described above is a natural combination and/or extension of existing techniques and hence its analysis will be largely based on the existing works. Correctness of verification follows from those in [8,9] and is trivial to see. Security analysis of our scheme will be deferred to the full version of this paper.

5 Privacy-Preserving Identity-Management

5.1 SPICE for Web Authentication

Here we show a typical run-time scenario of our system for a typical web authentication scenario. The whole system architecture is shown in Figure 3. The white color boxes represent the components from existing technologies (e.g., [6,17]), and the boxes with diagonals represent our new components. There are five components: *registrar*, *source CSP*, *receiving CSP*, *user* and *web browser*. The web browser interacts with the CSPs on behalf of the user to perform online authentication. The registrar is a trusted third party, which generates source certificate for each user. It has no interaction with the web browser, and will not involve in the authentication process. It can remain offline unless there may be new users joining the system.

The registrar manages user attributes, and uses the algorithm GS.UG to generate users' credentials. Then a source certificate is generated from the credentials for the user by registrar. Each user keeps the source certificate locally. When user accesses the browser to request for a service from a source CSP, the CSP will send back an authentication request, which includes the attributes names to be authenticated. These requests can be sent using an HTTP GET packet for example. When a user receives the authentication request, the functions GS.Rand and GS.Hide will be used to randomize and sanitize a certificate. The user uploads the randomized certificate to source CSP through web browser. The source CSP uses function GS.Ver' to authenticate the certificate. If successful, source CSP provides the service to user.

If a source CSP and a receiving CSP join together to provide a packaged service, the receiving CSP needs to authenticate user too. Firstly, the source CSP uses the *vocabulary conflicts handler module* [6] to check if there are attributes naming mismatch. If so, the CSP may generate another certificate using GS.Sign accompanying this certificate. Then, the source CSP generates another certificate by sanitizing (i.e., using GS.Hide to hide) the attributes that receiving CSP does not concern and re-randomizing (i.e., applying GS.Rand on) the certificate. The

sanitized certificate is sent to receiving CSP. A receiving CSP authenticates the certificate using GS.Ver'.

Additionally, the *naming management service* stores the mappings with other service provider ontologies, and the sets of synonyms (Synsets). *Policy repository* stores attributes verification policies. The *request manager component* asks the users for the attributes necessary for verification. The vocabulary conflicts handler checks if there are receiving CSP required attributes names that match the Synsets of itself.

A receiving CSP should have same modules as a source CSP. However, in order to distinguish between their roles, we omit some of the components which are only useful when they take another side of role.

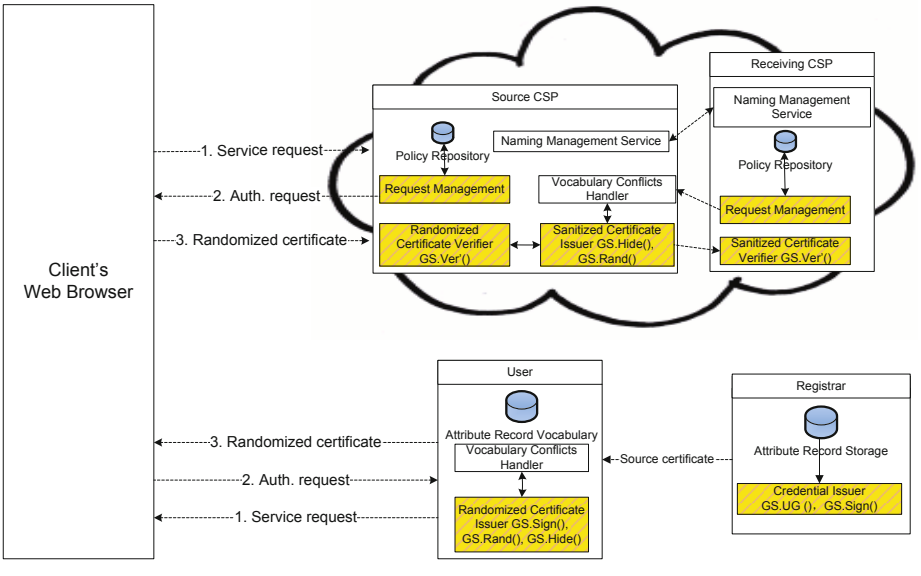

Fig. 3. System Architecture based on Randomizable Group Signatures

5.2 Security, Privacy and Functional Requirements

Unlinkability. The source certificate will be randomized by user for each request, so that the CSPs cannot obtain a certificate of a special form. If any CSP can link these randomized certificates to the same source certificate, this breaks the perfect unlinkability provided by GS.Rand which leads to a contradiction.

Delegatable Authentication. In order to allow a receiving CSP to authenticate a user without directly contacting with the user, firstly re-randomization is done, then sanitization is done by issuing a new group signature which is also re-randomizable and unlinkable.

Anonymity. CSPs are not able to see the clear values of the user attributes from the certificate whenever the user chose to hide them using GS.Hide, which is guaranteed by the hiding property of the commitment and the zero-knowledge property of the proof implicitly used by GS.

Registrar is the trusted party who controls which user has which set of attributes, so we do not consider anonymity against this party. In case the registrar is not allowed to see this sensitive personal information, one may actually use a group signature scheme which supports signing committed messages to realize a signing process similar to that of a blind signature scheme. The hiding algorithm presented in the previous section essentially changes a signature signing directly on a message to one signing on a commitment of that message, but for simplicity of our presentation, an explicit description of this variant is omitted.

Accountability. From the identifiability of GS, the registrar can trace which user (or which source CSP in the case of naming mismatch) issued the certificate, by the GS.Open algorithm using its group secret key.

User Centric Access Control. In our system, the user can choose which attributes to reveal when presenting a certificate, the disclosure of user identity or attributes is no longer arbitrarily or at the will of CSP.

Single Registration. The users only need to contact the registrar once in our system, instead of contacting the registrar every time authentication is about to occur for preserving the unlinkability in previous systems.

Efficiency. For unlinkable authentication, re-randomization is needed and its time complexity is in the total number of attributes a user possesses. However, it only involves a small number of exponentiations for each attribute. In particular, no pairing operation is involved on the user side. Also, we note that the randomization can be pre-computed. The only online operation for authentication is signing appending the certificate with the last block of the message, which takes constant time (in the number of attributes). We will report a performance evaluation of our prototype implementation for different settings in the full version of this paper.

6 Conclusion

Privacy and security have become a critical concern with the immense growth in the popularity of cloud computing, and digital identity management (DIM) being one of the critical components. We proposed a privacy-aware interoperable DIM system for the cloud based on the simple use of randomizable group signatures, which solved two open problems (unlinkability and delegatable authentication) left by Bertino *et al.* [6]. Our scheme relies on the conceptually simple use of extended group signatures. Most part of the operations in our system can be

performed offline and we remove the need of contacting the registrar before every authentication or storing a large amount of certificates. We believe the overhead is quite minimal for the privacy concern and we leave the design of more efficient anonymous credential systems as our future work.

References

1. Shibboleth, `http://shibboleth.internet2.edu`
2. Abe, M., Chow, S.S.M., Haralambiev, K., Ohkubo, M.: Double-Trapdoor Anonymous Tags for Traceable Signatures. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 183–200. Springer, Heidelberg (2011)
3. Alliance, C.S.: Top Threats to Cloud Computing V1.0 (March 2010), `https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf`
4. Angin, P., Bhargava, B.K., Ranchal, R., Singh, N., Linderman, M., Othmane, L.B., Lilien, L.: An Entity-Centric Approach for Privacy and Identity Management in Cloud Computing. In: IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 177–183 (2010)
5. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
6. Bertino, E., Paci, F., Ferrini, R., Shang, N.: Privacy-preserving Digital Identity Management for Cloud Computing. IEEE Data Eng. Bull. 32(1), 21–27 (2009)
7. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
8. Boyen, X., Waters, B.: Compact Group Signatures Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
9. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
10. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving Revocation with Efficient Update of Anonymous Credentials. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 454–471. Springer, Heidelberg (2010)
11. Celesti, A., Tusa, F., Villari, M., Puliafito, A.: Security and Cloud Computing: InterCloud Identity Management Infrastructure. In: WETICE, pp. 263–265. IEEE Computer Society (2010)
12. Chow, S.S.M.: Real Traceable Signatures. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 92–107. Springer, Heidelberg (2009)
13. Chow, S.S.M., Phan, R.C.-W.: Proxy Re-signatures in the Standard Model. In: Wu, T.-C., Lei, C.-L., Rijmen, V., Lee, D.-T. (eds.) ISC 2008. LNCS, vol. 5222, pp. 260–276. Springer, Heidelberg (2008)
14. Fuchsbauer, G., Pointcheval, D.: Proofs on Encrypted Values in Bilinear Groups and an Application to Anonymity of Signatures. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 132–149. Springer, Heidelberg (2009)
15. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

16. Hussain, M.: The Design and Applications of a Privacy-Preserving Identity and Trust-Management System. PhD thesis, School of Computing, Queen's University, Canada (2010), http://hdl.handle.net/1974/5520
17. Kalfoglou, Y., Schorlemmer, W.M.: If-map: An ontology-mapping method based on information-flow theory. J. Data Semantics 1, 98–127 (2003)
18. Paterson, K.G., Schuldt, J.C.N.: Efficient Identity-Based Signatures Secure in the Standard Model. In: Batten, L.M., Safavi-Naini, R. (eds.) ACISP 2006. LNCS, vol. 4058, pp. 207–222. Springer, Heidelberg (2006)
19. Ranchal, R., Bhargava, B.K., Othmane, L.B., Lilien, L., Kim, A., Kang, M.H., Linderman, M.: Protection of Identity Information in Cloud Computing without Trusted Third Party. In: IEEE Symposium on Reliable Distributed Systems (SRDS), pp. 368–372 (2010)
20. Rocha, F., Correia, M.: Lucy in the Sky without Diamonds: Stealing Confidential Data in the Cloud. In: DSN Workshop – Dependability of Clouds, Data Centers and Virtual Computing Env., pp. 129–134 (2011)
21. Takabi, H., Joshi, J.B.D., Ahn, G.-J.: Security and Privacy Challenges in Cloud Computing Environments. IEEE Security & Privacy 8(6), 24–31 (2010)
22. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)