PAPER

Efficient Consistency Achievement of Federated Identity and Access Management Based on a Novel Self-Adaptable Approach

Shi-Cho CHA[†a)], *Member and* Hsiang-Meng CHANG[†], *Nonmember*

SUMMARY Federated identity and access management (FIAM) systems enable a user to access services provided by various organizations seamlessly. In FIAM systems, service providers normally stipulate that their users show assertions issued by allied parties to use their services as well as determine user privileges based on attributes in the assertions. However, the integrity of the attributes is important under certain circumstances. In such a circumstance, all released assertions should reflect modifications made to user attributes. Despite the ability to adopt conventional certification revocation technologies, including CRL or OCSP, to revoke an assertion and request the corresponding user to obtain a new assertion, re-issuing an entirely new assertion if only one attribute, such as user location or other environmental information, is changed would be inefficient. Therefore, this work presents a self-adaptive framework to achieve consistency in federated identity and access management systems (SAFIAM). In SAFIAM, an identity provider (IdP), which authenticates users and provides user attributes, should monitor access probabilities according to user attributes. The IdP can then adopt the most efficient means of ensuring data integrity of attributes based on related access probabilities. While Internet-based services emerge daily that have various access probabilities with respect to their user attributes, the proposed self-adaptive framework significantly contributes to efforts to streamline the use of FIAM systems.

key words: *federated identity and access management, federated identity and access management, single sign-on*

1. Introduction

Recent advances in cloud computing, service oriented architecture (SOA), and social network services have enabled various organizations to provide federated services collaboratively. Users can thus elicit a federated service that consists of services provided by different service providers without knowing the actual one providing the service. For instance, members of the social network Facebook can adopt services provided by Facebook-allied organizations without registering new accounts and logging-in to the services [1]. Such services can decide privileges for the Facebook members based on their personal profiles and other confidential information provided by Facebook. In this case, federated identity and access management (FIAM) systems enable users to securely access services seamlessly as well as allow service providers to coordinate user privileges within the scope of their services [2], [3].

Among several online service providers offering a federated service collaboratively, a service provider may not know in advance what users will use its services. Therefore, current FIAM systems normally adopt a ticket-based or assertion-based approach [2]. In such an approach, services can request that their users obtain assertions to use the services from trusted identity providers. An assertion contains user identity-related information and is signed by an identity provider. The associated service can then authenticate the user based on that assertion. Although a service can determine user privileges based on identity and origin of the user, the service may have more advanced authorization requirements. For instance, a service may require a user's location and other environmental information to determine the user privileges. To fulfill such requirements, FIAM systems normally enable attribute authorities[*] to embed attribute-related information, such as age, gender, and group membership, into assertions. Moreover, services can request that attribute authorities provide relevant user information for authorization and access control purposes.

As user privileges to access a service are determined based on attributes in user assertions, the integrity of such attributes is a relevant issue in certain services. In this case, conventional certification revocation technologies, such as certificate revocation list (CRL) [4] or online certificate status protocol (OCSP) [5], can be adopted to determine the assertion status. For an invalid assertion, a new assertion containing the most recent information can be requested from its allied identity provider. However, the approach may be inefficient in certain services. For instance, assume that the content of certain data is seldom updated and the number of services requiring the data is small. Requesting identity providers to send the latest content to the services is a more efficient approach than requesting the service to obtain the status information of the data each time that the services must use the data.

Therefore, this work presents a novel self-adaptive framework to achieve strong consistency in federated identity and access management (SAFIAM) systems. SAFIAM selects an appropriate scheme automatically to ensure strong consistency between data in issued assertions and data stored in identity providers or attribute authorities based on current circumstances, including access probabilities and the number of services that may require such attributes. Comparatively, current studies only provide an interface or standards for services and attribute authorities to maintain

Manuscript received August 16, 2010.
Manuscript revised August 10, 2011.

[†]The authors are with the Information Management Department, National Taiwan University of Science and Technology, Taipei, 106, Taiwan (R.O.C.).
a) E-mail: csc@cs.ntust.edu.tw

DOI: 10.1587/transinf.E95.D.577

[*]Identity providers may also play the role of attribute authorities.

Copyright © 2012 The Institute of Electronics, Information and Communication Engineers

data consistency with respect to user attributes. For instance, attribute authorities can inform services that a specified user attribute of a user is invalidated in the framework proposed by Shafiq, Bertino, and Ghafoor [6]. Additionally, services can obtain the most recent user profiles in SAML [7], OpenSocial [8], or OAuth [9] standards. While SAFIAM can select the scheme that uses the least bandwidth to ensure data consistency with respect to user attributes among services, identity providers, and attributes authorities for different users and services, the proposed self-adaptive framework significantly contributes to efforts to increase the efficiency of FIAM systems.

The rest of the paper is organized as follows. Section 2 introduces pertinent literature. Section 3 then reviews the proposed framework. Next, Sect. 4 describes how the proposed framework selects the optimal strategy to achieve consistency. Section 5 illustrates the algorithm for the proposed approach. Additionally, Sect. 6 summarizes the simulation results to demonstrate the effectiveness of the proposed framework. Conclusions are finally drawn in Sect. 7, along with recommendations for future work.

2. Pertinent Literature

2.1 SSO, FIM, and FIAM

Single sign-on (SSO) systems, such as remote authentication dial in user service (RADIUS) [10] and Kerberos [11], were developed to reduce overhead costs and security risks in order to enable users to manage their ID/passwords in various systems. An SSO system allows individuals to authenticate their identity once and use multiple services that an organization offers. Further details regarding SSO can be found in articles written by Clercq [12] and Grubb and Carter [13] and, therefore, are omitted here for brevity.

Often viewed as a special-purposed SSO system, a FIM system provides a standard means of individual access to resources shared among various organizations with a single identity, enabling an organization to easily form a partnership with others [14]. In state-of-the-art, there are several FIM standards, such as ID-FF [15], SAML [7], Akenti [16], WS-Federation [17], OpenID [18], CardSpace [19], and so forth, have been proposed. Additionally, several FIM systems, including Shibboleth [20], GridShib [21], MAMS [22], SWIFT [23], are currently adopted in Web and grid-based applications. Because there are various different kinds of standards and protocols, several studies, such as [3], [24], [25] and so on, address the interoperability issues among the standards.

In addition to the functions of SSO and FIM systems, FIAM systems further consider access control and authorization processes. As mentioned earlier, current FIAM systems normally adopt a ticket-based or assertion-based scheme. Because a service determines user privileges based on the user assertion, a trustworthy party should issue the assertion. Therefore, trust among services, identity providers, and attribute authorities has received considerable interest. For instance, identity providers and attribute authorities can embed assurance levels based on NIST SP800-63 in the assertions that they issue. Therefore, a service can determine the trustworthiness of an assertion based on its assurance level [26]. Additionally, a service can either obtain a dynamic trust list from a trusted third party [27] or request a trusted third party [28] to authenticate identity providers and attribute authorities to identify trustworthy identity providers and attribute authorities dynamically. Moreover, an individual may have several accounts and associated attributes in different identity providers or attribute authorities. Therefore, Chadwick and Inman [29] and Dabrowski and Pacyna [30] developed schemes to link an individual's attributes in different identity providers or attribute authorities for authorization purposes. However, current FIAM standards have not emphasized how to achieve consistency among user attributes in assertions and in identifying providers or attribute authorities. Therefore, this work develops a self-adaptive framework to achieve consistency in federated identity and access management systems (SAFIAM).

2.2 Strong Consistency in Distributed Systems

In distributed systems, a strong consistency among different data copies can be achieved in several ways. Because only a specified identity provider can update user attributes, user attributes in the identity provider can be viewed as the original copy of the objects. Moreover, consistency between identity providers and service providers can be maintained using cache-coherence protocols. Cache-coherence protocols can be classified into the following schemes [31]–[33].

- Client validation – Each client caching a copy of an object can query the server that holds the original object whether the object is updated since it is obtained. If the object has been updated, the client can request the latest copy from the server; otherwise, the client uses the object directly.
- Server invalidation – The server informs each client caching a copy of an object when the server updates the object. After receiving the message, the client deletes the stale copy. Then, when an individual wishes to use the object in the future, the client obtains a new copy from the server.
- Server pushing – Instead of notifying clients that the cached objects are updated as in a server invalidation scheme, a server can transfer updated objects to the clients that hold copies of the object.

Rodriguez and Sibal [31] further proposed SPREAD based on the schemes. SPREAD can determine the most efficient scheme to achieve strong consistency of Web contents between Web servers and Web proxies based on the characteristics of Web contents. SAFIAM adopts a similar approach to selecting a means to achieve strong consistency in FIAM systems. However, as Web proxies can only obtain Web contents from Web servers, service providers can request user attributes from both identity providers and users' assertions in FIAM systems. Therefore, this work modified the problem model to reflect the difference.

3. Framework Overview

Figure 1 depicts the framework of SAFIAM, in which several identity providers can issue assertions for users to use different services[†]. In SAFIAM, each service is assumed to have a local repository to store user attributes, i.e. an assumption that reflects actual circumstances. For instance, in the scenario of OpenID, a user can register as a member of an identity provider, provide personal profiles to the identity provider, and obtain a global unique identity. When the user requests for the first time to view a web page from a Web site that supports OpenID, the web site instructs the user to provide the identity and then use that identity to locate the identity provider. The web site can also instruct the identity provider to authenticate the user and forward the user profiles to the site. Therefore, the site can generate a local account based on the data. When the user desires to view web pages from the site again, the site can simply instruct the identity provider to authenticate the user and obtain pertinent data in a local repository based on the user's identity [34].

For users desiring to use a service, the service requires that users display assertions issued by their trusted identity providers. If lacking a valid assertion, users send requests to one of the identity providers. Upon receiving the user requests, the *Request Logging* mechanism logs the requests. The request contains information on what attributes that the assertions should include. After verifying the identity of users, the *Request Processor* informs the *Assertion Issuer* to generate assertions based on *User Profiles* and sends the assertions to the users.

When a service receives requests from users, the *Assertion Verification* mechanism verifies the associated assertions and filters out the invalid attributes. The *Assertion Processor* then extracts the user identity and other attributes from the assertions and, finally, sends the data to the *Attribute Assembler*.

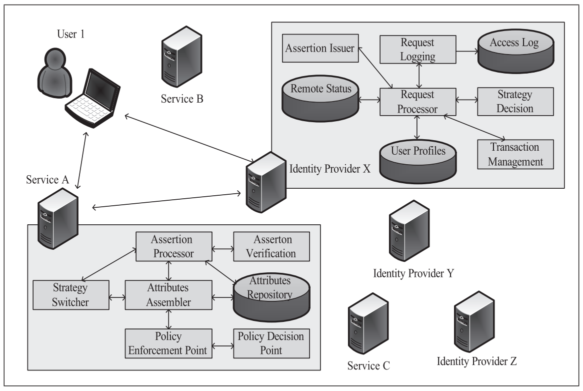

Fig. 1 Overview of SAFIAM.

If users have not used the service previously, the service creates an entry in *Attribute Repository* and stores the data. Moreover, the Attribute Assembler retrieves missing attributes from the identity provider. Otherwise, the Attribute Assembler gathers the attributes for privilege decision based on the strategy set by the *Strategy Switcher*. The current SAFIAM adopts the following strategies based on the cache-coherence schemes discussed in Sect. 2.2:

- Service validation (SV) strategy: According to Fig. 2 (a), services can obtain user identities and assertions (Step Q_1^1). The services then query related identity providers whether user attributes in their local repository are updated (Step Q_2^1). If the data have been updated, the identity providers or IdPs send the updated attributes to the services (Step Q_{3a}^1). Alternatively, the identity providers inform the service that they can use the attributes in a local repository directly (Step Q_{3b}^1). However, when users or corresponding identity providers wish to update user data, the identity providers can update the data directly without notifying the services that have the user data (Step U_1^1 in Fig. 2 (b)).
- IdP invalidation (II) strategy: Figure 3 (a) and (b) provide an overview of this strategy. After user identities are obtained (in Step Q_1^2), the services verify whether the relevant data in their local repository have been marked as invalid. Services without a valid copy of the user data request the data from related identity providers (Step Q_2^2 and Step Q_3^2). Otherwise, the services use local data directly. In contrast with the service validation strategy, an identity provider records which service have user data in the *Remote Status* database. When the identity providers receive a request to update user data (Step U_1^2, the identity providers obtain a list of services containing the user data. Next,

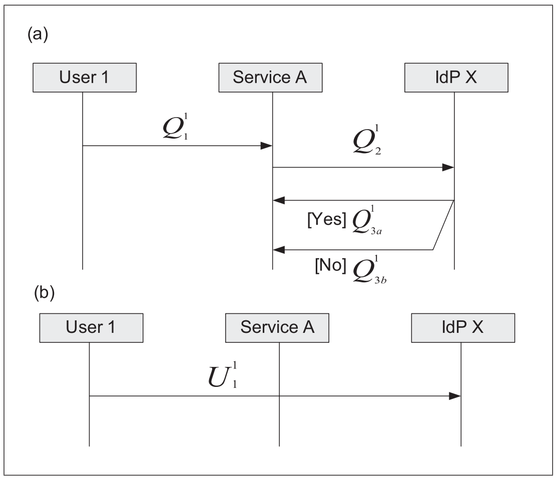

Fig. 2 Overview of service validation strategy.

[†]Because identity providers can cover the function of attribute authorities, identity providers are used in this work to represent both identity providers and attribute authorities for simplicity.

the *Transaction Management* component in the identity providers instructs the services to mark the updated data as invalid (Step U_2^2). Upon receiving notification, the services return messages to the identity providers (Step U_3^2). To enable identity providers to analyze access probabilities of user data, apart from sending an acknowledgment, the services inform the identity providers the times that the user data are used since the last time that the systems were notified. The identity providers store the information in their *Access Log* database.

- IdP pushing (IP) strategy: Identity providers send the updated data to services containing user data when the user data are updated (Step U_1^3 and U_2^3 in Fig. 4 (b)). Similar to the IdP invalidation strategy, the services send acknowledgments along with the access information with respect to the user data back to the identity provider after receiving the updated requests (Step U_3^3). Because user data in services are always updated, services can use their local data to determine user privileges immediately (Step Q_1^3 in Fig. 4 (a)).

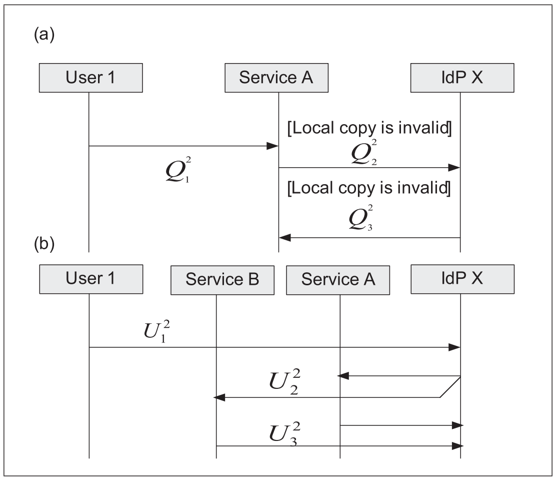

Fig. 3 Overview of IdP invalidation strategy.

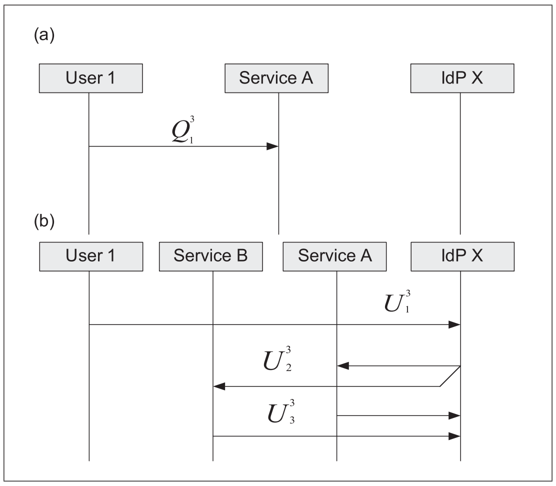

Fig. 4 Overview of IdP pushing strategy.

After the Attribute Assembler collects necessary attributes using the above strategies, the *Policy Decision Point* determines the user privileges based on the attributes with subsequent enforcement by the *Policy Enforcement Point*. Finally, the *Strategy Decision* component in identity providers monitors and analyzes the access logs continually and, then, identifies the most appropriate strategy based on the logs.

4. Strategy Decision

Table 1 lists the notations in this work. Communication costs of the strategies are compared by classifying requests to FIAM systems according to the following four situations:

- RR Situation: Services receiving user requests may require a set of attributes A to determine the user privileges to gain access to the services. This work classifies the request as RR situation if the attributes are not modified (or there is no update requests to the attributes) since the services requests to obtain the attributes previously. Communication costs of the RR situation for each strategy can be summarized as follows:
  - Service validation strategy

$$C_{SV}^q = C_T + C_A^q + C_c \tag{1}$$

Table 1 Glossary of notations.

C_X^q	Communication costs for services to obtain the latest attributes regarding users with strategy X.
C_X^u	Communication costs to update users' attributes with strategy X.
C_T	Communication costs of transferring an assertion to request a service.
C_A^q	Communication costs of transferring a message to query the status of a specified attribute set or requesting an identity provider to transfer updated attributes back.
C_c	Communication costs of notifying services that attributes in A are invalid or transferring a message to inform services that data have not been updated.
C_A^u	Communication costs of passing a message to update a set of attributes A or a message to inform services values of attributes in A.
N_A	Number of services that subscribe to the attribute set A.
N_A^c	Number of services that subscribe to the attribute set A and marked A as up-to-date.
N_S	Number of services that received service requests from users.
$\hat{p}_i$	Expected probability to update attributes in i_{th} period.
$\hat{q}_i$	Expected probability to access service in i_{th} period.
TC_X	Total communication costs in the interval between two updates when using strategy X.
w	Parameter to indicate how many periods are considered for predicting $\hat{p}_i$.
x_i	Number of requests from the request after $(i-1)_{th}$ update request to receive i_{th} update request.

- IdP invalidation strategy

$$C^q_{II} = C_T \tag{2}$$

- IdP pushing strategy

$$C^q_{IP} = C_T \tag{3}$$

- UR Situation: This work classifies user requests in the UR situation if users update related attributes after the services obtain the attributes the last time. Communications cost of UR situation for each strategy can be shown as follows:
  - Service validation strategy

$$C^q_{SV} = C_T + C^q_A + C^u_A \tag{4}$$

  - IdP invalidation strategy

$$C^q_{II} = C_T + C^q_A + C^u_A \tag{5}$$

  - IdP pushing strategy

$$C^q_{IP} = C_T \tag{6}$$

- RU Situation: This work classifies a request to update a set of attributes A into the RU situation if a service obtains the attributes since the attributes are updated the last time. Communication costs of the RU situation for each strategy can be listed as follows:
  - Service validation strategy

$$C^u_{SV} = C^u_A \tag{7}$$

  - IdP invalidation strategy

$$C^u_{II} = C^u_A + N^c_A * C_c \tag{8}$$

  - IdP pushing strategy

$$C^u_{IP} = C^u_A + N_A * C^u_A \tag{9}$$

- UU Situation: This work classifies a request to update a set of attributes A into the UU situation if no service obtains the attributes since the attributes have been updated the last time. Communication costs of the UU situation for each strategy can be shown as follows:
  - Service validation strategy

$$C^u_{SV} = C^u_A \tag{10}$$

  - IdP invalidation strategy

$$C^u_{II} = C^u_A \tag{11}$$

  - IdP pushing strategy

$$C^u_{IP} = C^u_A + N_A * C^u_A \tag{12}$$

According to Eqs. (1)–(12), a user desiring to use a service must transfer an assertion with size C_T to the service, regardless of the strategy adopted. Similarly, the user must send attributes with size C^u_A to update the attributes in all strategies. Therefore, C_T and C^u_A can be neglected in the above equations. Table 2 summarizes the communication costs under different situations by the strategy.

Table 2 Communication costs under different situations based on the strategy adopted.

	Service Validation	**IdP Invalidation**	**IdP Pushing**
C_{RR}	$C^q_A + C_c$	0	0
C_{UR}	$C^q_A + C^u_A$	$C^q_A + C^u_A$	0
C_{RU}	0	$N^c_A * C_c$	$N_A * C^u_A$
C_{UU}	0	0	$N_A * C^u_A$

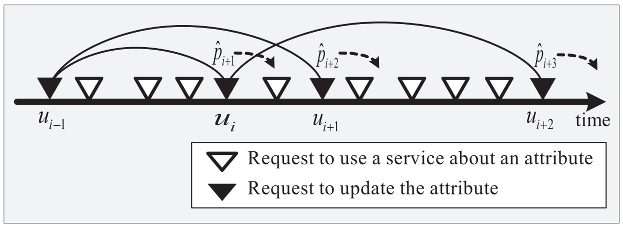

Fig. 5 Overview of prediction model.

As mentioned earlier in Sect. 3, an identity provider monitors the access probabilities of its user attributes. Therefore, the identity provider can use the most efficient means of achieving consistency for each attribute based on the access probabilities. This work demonstrates first how SAFIAM estimates future access probabilities with respect to an attribute or a set of attributes of users based on previous access probabilities. This work then demonstrates how SAFIAM calculates cumulative communication costs for an attribute when using different strategies based on associated estimated access probabilities and the communication costs to handle user requests in different strategies, as shown in Eqs. (1)–(12).

According to Fig. 5, the interval between two update requests to users' attributes is treated as a prediction period. An identity provider predicts the probability $\hat{p}_i$ that an attribute is updated in a period based on the number of users allowed to access services related to the attribute in previous w periods. Upon receiving user requests, the services must query and obtain contents of the attribute. Therefore, in this work, query requests are used simply to represent the situations. Where x_i denotes the number of requests from the request after $(i-1)_{th}$ update request to receive i_{th} update request (the i_{th} update request is included in x_i). Therefore, $x_i - 1$ query requests are related to an attribute in i_{th} period. Eqs. (13) and (14) show the probability that x_i equals an integer a and the expected value of x_i, respectively.

$$P(x_i = a) = \hat{p}_i \cdot \hat{q}_i^{a-1} \tag{13}$$

$$E(x_i) = \frac{1}{\hat{p}_i} \tag{14}$$

$$\begin{aligned} TC_X = {} & P(x_i = 1) \cdot C_{UU} \\ & + P(x_i > 1) \cdot [E(N_S) \cdot C_{UR} \\ & + (E(x_i) - 1 - E(N_S)) \cdot C_{RR} + C_{RU}] \end{aligned}$$

$$= \hat{p}_i \cdot C_{UU} + \hat{q}_i \cdot [E(N_S) \cdot C_{UR} + (E(x_i) - 1 - E(N_S)) \cdot C_{RR} + C_{RU}] \quad (15)$$

Consequently, an identity provider can estimate the cumulative communication costs involving user attributes in a specific period by summing up $E(x_i)$ user requests related to the attribute and the successive update to the attribute. According to Eq. (15)[†], the equation consists of two parts: The first part represents the expected communication costs when no service request involving an attribute in a specific period is available. Under this circumstance, only C_{UU} must be considered. The second part reveals the expected communication costs when x_i exceeds 1. Assume that $E(N_S)$ services require the attribute. This work determines the expected communication costs when x_i is greater than 1 by summing up the following three components:

- $E(N_S)$ times the communication costs of an update request to the attribute in the UR situation.
- $E(x_i) - 1 - E(N_S)$ times the communication costs between the IdP and a service when the service receives the user request consecutively (or in the RR situation)[††]
- The communication costs of the succeeding update request to the attribute (or in the RU situation).

Assume that N_A services may need the attribute. The expected number of services that receive service requests $E(N_S)$ can be estimated based on solutions to the conventional occupancy problem [35] in Eq. (16). For brevity, this work omits the inference details of the equation.

$$E(N_S) = N_A \cdot \left[1 - \left(\frac{N_A - 1}{N_A}\right)^{E(x_i)-1}\right] \quad (16)$$

Finally, Eqs. (17)–(19) summarize the estimated cumulative communication costs with respect to an attribute in different strategies for a specific period[†††]. An IdP can estimate $\hat{p}_i$ first and use $\hat{p}_i$, evaluate communication costs in a specific period based on the equations and, then identify the best strategy. The identity provider that does not currently adopt the best strategy notifies related services to switch to the strategy.

$$\begin{aligned} TC_{SV} &= \hat{p}_i \cdot 0 + \hat{q}_i \cdot [E(N_S) \cdot (C_A^q + C_A^u) \\ &\quad + (E(x_i) - 1 - E(N_S)) \cdot (C_A^q + C_c) + 0] \\ &= \hat{q}_i \cdot [E(N_S) \cdot (C_A^q + C_A^u) \\ &\quad + (E(x_i) - 1 - E(N_S)) \cdot (C_A^q + C_c)] \quad (17) \end{aligned}$$

$$\begin{aligned} TC_{II} &= \hat{p}_i \cdot 0 + \hat{q}_i \cdot [E(N_S) \cdot (C_A^q + C_A^u) \\ &\quad + (E(x_i) - 1 - E(N_S)) \cdot 0 + N_A^c \cdot C_c] \\ &= \hat{q}_i \cdot [E(N_S) \cdot (C_A^q + C_A^u) + N_A^c \cdot C_c] \\ &= \hat{q}_i \cdot [E(N_S) \cdot (C_A^q + C_A^u + C_c)] \quad (18) \end{aligned}$$

$$\begin{aligned} TC_{IP} &= \hat{p}_i \cdot N_A \cdot C_A^u + \hat{q}_i \cdot [E(N_S) \cdot 0 \\ &\quad + (E(x_i) - 1 - E(N_S)) \cdot 0 + N_A \cdot C_A^u] \\ &= \hat{p}_i \cdot N_A \cdot C_A^u + \hat{q}_i \cdot N_A \cdot C_A^u \\ &= N_A \cdot C_A^u \quad (19) \end{aligned}$$

5. The Algorithm

This section introduces the algorithm used in SAFIAM. For each attribute or attribute set A_i, an identity provider should maintain a data entry (ID_{A_i}, NQ_{A_i}, NU_{A_i}, s_{A_i}, $\mathbf{SS}_{A_i}$, C_{A_i}, t_{A_i}), where ID_{A_i} represents the identity of A_i, s_{A_i} represents the current strategy adopted to deal with A_i; NQ_{A_i} represents the number of queries about A_i; NU_{A_i} represents the number of update requests about A_i; $\mathbf{SS}_{A_i}$ represents the set of services that must be notified when A_i is updated; C_{A_i} represents content of the attribute, and t_{A_i} is the timestamp of the attribute.

According to Fig. 6, upon receiving requests to update an attribute with identity ID and contents C'_{A_i} to replace C_{A_i}, an identity provider first locates the attribute A_i

```
Receiving ("Update", $ID$, $C'_{A_i}$)
begin
    Find out ($ID_{A_i}$, $NQ_{A_i}$, $NU_{A_i}$, $s_{A_i}$, $\mathbf{SS}_{A_i}$, $C_{A_i}$, $t_{A_i}$)
            where $ID_{A_i}$=$ID$;
    $NU_{A_i}$++;
    If ($NU_{A_i} + NQ_{A_i}) > T$
        $\hat{p} = \frac{NU_{A_i}}{NU_{A_i}+NQ_{A_i}}$;
        Find the best strategy $s$ based on $\hat{p}$;
        If $s \neq s_{A_i}$
            $s_{A_i} = s$;
        Endif
        $NU_{A_i} = 0, NQ_{A_i} = 0$;
    Endif
    If $s_{A_i}$ = SV
        Update $t_{A_i}$;
        For each $ss_j$ in $\mathbf{SS}_{A_i}$
            Send ("Update", $ID_{A_i}$, $C'_{A_i}$, $s_{A_i}$) to $ss_j$;
            Remove $ss_j$ from $\mathbf{SS}_{A_i}$;
        Endfor
    Else if $s_{A_i}$ = II
        For each $ss_j$ in $\mathbf{SS}_{A_i}$
            Send ("Obsolete", $ID_{A_i}$, $s_{A_i}$) to $ss_j$;
            Remove $ss_j$ from $\mathbf{SS}_{A_i}$;
        Endfor
    Else If $s_{A_i}$ = IP
        Update $t_{A_i}$;
        For each $ss_j$ in $\mathbf{SS}_{A_i}$
            Send ("Update", $ID_{A_i}$, $C'_{A_i}$, $s_{A_i}$) to $ss_j$;
        Endfor
    Endif
    $C_{A_i} = C'_{A_i}$;
end
```

Fig. 6 Pseudocode for an identity provider to handle update requests.

[†]According to Eq. (13), $p(x_i = 1) = \hat{p}_i \cdot \hat{q}_i^0 = \hat{p}_i$. Additionally, because $\hat{p}_i + \hat{q}_i = 1$ by definition, $p(x_i > 1) = 1 - p(x_i = 1) = 1 - \hat{p}_i = \hat{q}_i$.

[††]As $E(x_i)$ requests include $E(x_i) - 1$ query requests and one update request, the $E(x_i) - 1$ query requests are spread around the $E(N_S)$ services. Therefore, the first query request regarding the attribute to a service is in the UR situation. The remainder $E(x_i) - 1 - E(N_S)$ requests are in the RR situation.

[†††]In Eq. 18, this study assumes that $N_A^c = E(N_S)$ because each service that receives service requests about A in a specific period needs to obtain the latest copies of A when $x_i > 1$.

```
Receiving ("Query", $ID$, $t$, $NQ$)
begin
    Find out ($ID_{A_i}$, $NQ_{A_i}$, $NU_{A_i}$, $s_{A_i}$, $\mathbf{SS}_{A_i}$, $C_{A_i}$, $t_{A_i}$)
            where $ID_{A_i}$=$ID$;
    $NQ_{A_i}$+=$NQ$;
    If ($NU_{A_i}$+ $NQ_{A_i}$) > $T$
        $\hat{p} = \frac{NU_{A_i}}{NU_{A_i}+NQ_{A_i}}$;
        Find the best strategy $s$ based on $\hat{p}$;
        If $s \neq s_{A_i}$
            $s_{A_i} = s$;
        Endif
        $NU_{A_i} = 0$, $NQ_{A_i} = 0$;
    Endif
    If $s_{A_i}$ = SV
        If $t < t_{A_i}$
            Respond ("Success", $C_{A_i}$, $t_{A_i}$, $s_{A_i}$) to the request;
        Else
            Respond ("Not Modified", null, null, $s_{A_i}$) to the
                request;
        Endif
    Else if $s_{A_i}$ = II or $s_{A_i}$ = IP
        Add the requester to $\mathbf{SS}_{A_i}$;
        Respond ("Success", $C_{A_i}$, $t_{A_i}$, $s_{A_i}$) to the request;
    Endif
end
```

Fig. 7 Pseudocode for an identity provider to handle query requests.

with identity *ID* and the associated data entry. The identity provider increases NU_{A_i} to reflect the number of update requests received. Therefore, the identity provider can use NQ_{A_i} and NU_{A_i} to estimate the probability $\hat{p}$ that the next request involving A_i is an update request. Additionally, this study re-estimates $\hat{p}$ every T requests. Advanced estimating schemes should be addressed in future work. The identity provider calculates the bandwidth requirement of various strategies based on Eqs. (17)–(19) and selects the best one as the new strategy. If selecting the SV strategy, the identity provider updates the timestamp of A_i. Moreover, the identity provider notifies services that owns copies of A_i if the identity provider switches to the SV strategy from II or IP strategies; If choosing the II strategy, the identity provider notifies services that owns copies of A_i that the contents of A_i are obsolete; the identity provider sends messages to services that owns copies of A_i to update the contents of the copies if IP strategy is adopted. Finally, the identity provider updates the contents of A_i. Notably, the identity provider is requested to piggyback its new strategy information in messages sent to services to reduce communication costs.

Figure 7 shows the pseudocode for identity providers to handle query requests from service providers. Because services may use their local copies of attributes if the services use II or IP strategies, services are requested to count the number of queries not sent to related identity providers as well as to send the number to identity providers along with their query requests to identity providers. After receiving a query request from a service, an identity provider identifies a matched data entry (ID_{A_i}, NQ_{A_i}, NU_{A_i}, s_{A_i}, $\mathbf{SS}_{A_i}$, C_{A_i}, t_{A_i}). The identity provider updates NQ_{A_i} based on the number of queries that have not been counted *NQ* in the request.

```
Receiving ("Query", $ID$)
begin
  Find out ($ID_{A_i}$, $NQ_{A_i}$, $s_{A_i}$, $C_{A_i}$, $t_{A_i}$) where $ID_{A_i}$=$ID$;
  If the data entry cannot not be found
    Send ("Query", $ID$, -1, 1) to the associated identity provider;
    Receive response (status, $C$, $t$, $ns$) from the identity provider;
    Create ($ID$, 0, $ns$, $C$, $t$);
    Send $C$ back to the requester;
  Else
    $NQ_{A_i}$++;
    If $s_{A_i}$ = SV
      Send ("Query", $ID$, $t_{A_i}$, $NQ_{A_i}$) to the associated identity
        provider;
      Receive response (status, $C$, $t$, $ns$) from the identity
        provider;
      If status = "Success"
        $C_{A_i} = C$; $t_{A_i} = t$; $s_{A_i} = ns$;
      Endif
      $NQ_{A_i}$= 0;
    Else If $s_{A_i}$ = II
      If $t_{A_i}$ = -1 or $NQ_{A_i} > T$
        Send ("Query", $ID$, $t_{A_i}$, $NQ_{A_i}$) to the associated
          identity provider;
        Receive response (status, $C$, $t$, $ns$) from the identity
          provider;
        $s_{A_i} = ns$; $C_{A_i} = C$; $t_{A_i} = t$; $NQ_{A_i}$= 0;
      Endif
    Else
      If $NQ_{A_i} > T$
        Send ("Query", $ID$, $t_{A_i}$, $NQ_{A_i}$) to the associated
          identity provider;
        Receive response (status, $C$, $t$, $ns$) from the identity
          provider;
        $C_{A_i} = C$; $t_{A_i} = t$; $s_{A_i} = ns$;
      Endif
    Endif
    Send $C_{A_i}$ back to the requester;
  Endif
end
```

Fig. 8 Pseudocode for a service to handle query requests triggered by users.

As for processes involving selection of the strategy as mentioned in the previous paragraph, the identity provider forecasts the costs of using various strategies and then selects the best one. If adopting the SV strategy, the identity provider checks the timestamp of the attribute in the service to determine whether the attribute is updated. If the attribute is updated, the identity provider transfers the latest contents of the attribute to the service. Otherwise, the identity provider notifies the service that the attribute has not been modified. If adopting either II or IP strategy, the identity provider adds the service to $\mathbf{SS}_{A_i}$ for future notification and then returns contents of the attribute to the service. Regardless of the strategy adopted, the identity provider piggybacks its strategy in response to the service. Therefore, the service can determine when to switch to another strategy.

Figure 8–10 provides major pseudocode of services. First, services may need users' attributes to satisfy their requests. In this case, services trigger query requests about attributes of the users. With respect to an attribute A_i, a service maintains a data entry (ID_{A_i}, NQ_{A_i}, s_{A_i}, C_{A_i}, t_{A_i}), where

```
Receiving ("Obsolete", $ID$, $s$)
begin
    If $s$=IP
        Remove data entry where $ID_{A_i} = ID$;
    Else
        Find out ($ID_{A_i}, NQ_{A_i}, s_{A_i}, C_{A_i}, t_{A_i}$) where $ID_{A_i} = ID$;
        $s_{A_i} = s; t_{A_i} = -1$;
    Endif
end
```

Fig. 9 Pseudocode for a service to handle obsolete requests.

```
Receiving ("Update", $ID$, $C$, $t$, $s$)
begin
    Find out ($ID_{A_i}, NQ_{A_i}, s_{A_i}, C_{A_i}, t_{A_i}$) where $ID_{A_i} = ID$;
    $C_{A_i} = C; s_{A_i} = s; t_{A_i} = -1$;
end
```

Fig. 10 Pseudocode for a service to handle update requests.

ID_{A_i}, s_{A_i}, C_{A_i}, and t_{A_i} have the same meaning as ID_{A_i}, s_{A_i}, C_{A_i}, and t_{A_i} in the data entry concerning attribute A_i that is maintained by the identity provider; NQ_{A_i} is the number of query requests about A_i that have not been counted by the associated identity provider. When a service wishes to obtain content of an attribute A_i, the service first find out the data entry about A_i based on the identity of A_i in its local storage. If the service cannot find out the data entry, the service sends a query request to associated identity provider to obtain content and timestamp of the attribute and the strategy to deal with the attribute. The service then creates a data entry of the attribute and store the entry in local storage for future usage. Otherwise, the service first increases NQ_{A_i} to reflect the query. If s_{A_i} is equal to SV, the service adopts the SV strategy to deal with the request. The service sends ID_{A_i}, t_{A_i}, and NQ_{A_i} to the associated identity provider. The identity provider checks whether the attribute is up-to-date. If the attribute is not up-to-date, the service receives content and timestamp of the attribute from the identity provider. After updating the data entry, the service reset NQ_{A_i} to 0. On the other hand, as the service uses II strategy to deal with the data, the service checks whether C_{A_i} is obsoleted based on the value of t_{A_i}. If C_{A_i} is obsoleted, the service sends a query request to the identity provider to obtain content and timestamp of the attribute. The service then deals with the attribute similar to the procedures in SV strategy. Finally, the service simply sends content of the attribute back to the requester if the service adopts IP strategy to deal with the attribute. Notably, to enable the identity provider to estimate $\hat{p}$ more accurately, the service can be enforced to send NQ_{A_i} to the identity provider when NQ_{A_i} is bigger than a threshold T if the service adopts II or IP strategies.

When an identity provider uses the II strategy to process an attribute, the identity provider sends a request to related services in order to make the attribute obsolete. According to Fig. 9, when a service receives an obsolete request ("Obsolete", ID, s) from an identity provider, the service checks the new strategy information s piggybacked in the request. If the new strategy is IP, the service removes the data entry whose identity is ID to enforce the service in order to establish a replication relationship with the identity provider when the service needs the attribute the next time. Otherwise, regardless of whether s is equal to SV or II, the service can simply set the timestamp of the attribute to -1 and switch the strategy to s to handle the attribute.

As mentioned earlier, when an identity provider uses IP strategy to handle an attribute, the identity provider sends update notifications to services that have copies of the attribute. According to Fig. 10, a notification includes an identity of an attribute, content and timestamp of the attribute, as well as the strategy to deal with the attribute. Consequently, a service can use the information to update information of the its local data entry with respect to the attribute.

6. Simulation Experiments

This work demonstrates the effectiveness of SAFIAM by implementing simulation programs with Java and exploiting the Mersenne Twister random number generator provided in uncommons-maths library to generate a random number for simulations. Moreover, this study performs the simulation experiments on a x86 personal computer with Intel i5-760 2.8 GHz CPU and 4 GB memory. First, the simulation experiment determines whether SAFIAM is better than simply using a strategy from beginning to end. This work implements identity providers and services with Java Servlet technologies and deploys an identity provider and a service in different contexts of an Apache Tomcat 6.0.32 application server. Moreover,the OAuth 1.0 protocol [9] is used for services to send query requests to the identity provider. In OAuth 1.0, identity providers ask users whether users allow services to obtain the user data. For users allowing the services to access their data, identity providers offer services keys to access the data. The services can thus use the keys and other credentials to obtain the data. In this case, the communication cost of transferring a query request to a specified attribute set A (C^q_A) is around 450 bytes. The communication cost of passing a message to update a set of attributes A or a message to inform services values of attributes in A (C^u_A) is around 200 bytes (including identity of the attributes, timestamp, new strategy, and other HTTP header information) plus size of the attributes. Communication cost of transferring an updated acknowledgement message, requesting an identity providers to transfer updated attributes back, or transferring a message to inform services that data have not been updated (C_c) is around 150 bytes (including identity of the attributes and other HTTP header information). This work also implements agent programs to emulate users. The agent programs send query requests to a specified service and send update requests to a specified identity provider.

The experiment is performed in three rounds. During each round, different sizes of data are used, i.e. from 32 bytes to 512 bytes. For a strategy, various agent programs are run concurrently to access different data items indepen-

Table 3 Average communication cost (bytes) per request by strategies and size of data.

	32 bytes	**128 bytes**	**512 bytes**
SV	727.0	837.8	1293.8
II	592.3	707.9	1163.1
IP	636.6	787.0	1362.5
SAFIAM-SV	558.4	681.2	1153.9
SAFIAM-II	552.9	674.1	1147.9
SAFIAM-IP	554.5	678.5	1158.8

Table 4 Average CPU load and response time per request by strategy and size of data.

Size of Data (bytes)	**Strategy**	**CPU Load(%)**	**Response Time (μsec)**
32	**SV**	37.13(0.72)	0.84($1.1\cdot10^{-4}$)
	II	41.98(1.56)	0.76($4.5\cdot10^{-5}$)
	IP	40.02(2.41)	0.82($3.5\cdot10^{-4}$)
	SAFIAM-SV	41.06(1.46)	0.71($3.3\cdot10^{-5}$)
	SAFIAM-II	41.11(1.41)	0.71($1.6\cdot10^{-5}$)
	SAFIAM-IP	42.92(1.95)	0.72($4.2\cdot10^{-5}$)
128	**SV**	36.62(1.17)	0.85($2.4\cdot10^{-4}$)
	II	42.41(3.07)	0.77($3.0\cdot10^{-5}$)
	IP	39.51(1.98)	0.82($1.7\cdot10^{-4}$)
	SAFIAM-SV	40.52(1.73)	0.72($1.3\cdot10^{-4}$)
	SAFIAM-II	41.86(0.65)	0.71($1.9\cdot10^{-5}$)
	SAFIAM-IP	40.82(2.19)	0.72($5.9\cdot10^{-5}$)
512	**SV**	39.26(1.13)	0.86($5.1\cdot10^{-5}$)
	II	40.71(2.94)	0.78($4.4\cdot10^{-5}$)
	IP	39.35(0.59)	0.84($1.9\cdot10^{-4}$)
	SAFIAM-SV	40.96(2.12)	0.73($6.8\cdot10^{-5}$)
	SAFIAM-II	40.73(1.78)	0.72($5.5\cdot10^{-5}$)
	SAFIAM-IP	40.37(2.29)	0.73($3.5\cdot10^{-5}$)

dently. Each agent program has its own update ratio (from 0.1 to 0.9), with all of them sending 20000 requests independently. The identity provider re-estimates $\hat{p}$ of a data every 100 requests regarding the data. Table 3 summarizes the experimental results. Because the SAFIAM scheme may initiate from different strategies, SAFIAM-XX is used here, demonstrating that the initial strategy is XX (SV, II, or IP). According to this table, the average communication costs of SAFIAM are the lowest in each case.

This study also considers the influence of the SAFIAM scheme on CPU loads and latency of requests. It implements a C#.NET program to measure CPU load with PerformanceCounter objects every second during the above experiments with various sizes of data (from 32 bytes to 512 bytes). Additionally, the agent programs mentioned above logs the time when the programs send requests and receive responses. An experiment is carried out ten times for each strategy and size of data. Moreover, mean and variance of the average CPU load and response time per request are calculated for each scenario. As presented in Table 4, the means and associated variances (in parentheses behind means) are given in each data cell. Although the experiments performed using the SAFIAM scheme have a higher average CPU load than associated with schemes in which only one strategy is used, the SAFIAM scheme does not put a heavy burden on the processors. Furthermore,

Table 5 Average CPU load by strategy, size of data, and number of requests to re-estimate $\hat{p}$ of a datum.

Number of Requests to Re-estimate $\hat{p}$		**100**	**500**
Size of Data(bytes)	**Strategy**	**CPU Load(%)**	**CPU Load(%)**
32	**SAFIAM-SV**	41.40(1.19)	40.72(1.50)
	SAFIAM-II	41.41(1.29)	40.96(1.33)
	SAFIAM-IP	41.45(3.13)	40.20(2.02)
128	**SAFIAM-SV**	41.22(0.66)	40.76(1.48)
	SAFIAM-II	41.70(1.12)	41.00(1.39)
	SAFIAM-IP	41.09(3.78)	40.13(2.62)
512	**SAFIAM-SV**	41.38(1.71)	40.81(0.70)
	SAFIAM-II	41.73(0.63)	41.06(1.05)
	SAFIAM-IP	41.77(4.88)	41.39(1.28)

since the SAFIAM scheme re-estimates the $\hat{p}$ of a datum every n^{th} request, where n is a specified number, as illustrated in Table 5, the experiments in which more requests are made to re-estimate $\hat{p}$ are associated with lower average CPU load than those in which fewer requests are made to re-estimate $\hat{p}$. However, increasing the number of requests to re-estimate $\hat{p}$ reduces the adaptability of the SAFIAM scheme because identity providers then need more time to become aware of changes to the access patterns. The impact of the number of requests for re-estimating is left for future work. Additionally, since this study adopts piggyback techniques to transfer control messages, the proposed SAFIAM scheme does not increase the response time beyond that of schemes in which only one strategy is used.

7. Conclusions and Future Work

Strong consistency is considered vital to preventing unauthorized access in FIAM systems. However, current FIAM systems normally do not stress efficiency to achieve strong consistency. To address this issue, this work presents a novel self-adaptive framework to achieve a strong consistency in federated identity and access management systems (SAFIAM). SAFIAM enables systems and identity providers to adapt themselves automatically and select the most efficient means of ensuring data integrity collaboratively based on access probabilities with respect to users' attributes. Identity providers or service providers do not need to know users' access probabilities with respect to attributes in advance, so SAFIAM is well suited for a situation in which new services, which have different requirements with respect to users' attributes, emerge daily. Therefore, the proposed self-adaptive framework significantly contributes to efforts to streamline the use of FIAM systems.

Despite its contributions, this work has certain limitations that point the way towards future research. Other than integrating SAFIAM into the current FIAM standards, future research should address other relevant issues. First, the current FIAM calculates the communication costs in different strategies based on the model in which all attributes are updated independently. However, various attributes are updated simultaneously. Therefore, the correlations about update requests among different attributes can be take into consideration. Under this circumstance, future studies should consider a more complex model.

Second, user attributes may have various consistency requirements. If a service obtains some permanently valid user attributes, such as birthday, from an identity provider, then the service does not have to maintain consistency about the data with the identity provider. Therefore, the total communication cost is reduced. Future work could set specifications for meta data on user attributes. Accordingly, identity providers may provide meta data about user attributes to help service providers determine how to process those attributes.

Third, various FIAM applications are available, including E-DRM and grid or cloud computing. Future work should address issues involving the implementation of SAFIAM in various applications. Further experiments or simulations can also validate the feasibility of SAFIAM in related applications.

Finally, future work should elucidate further the role of privacy in FIAM. To address this issue, identity providers may provide users with temporary assertions to access services. Intuitively, the services may not need to keep the contents of the assertions in a local repository because the assertions become useless in a relatively short time. Therefore, future work should incorporate the role of pseudonymity and the anonymity function in the proposed framework.

Acknowledgments

This work was supported in part by the National Science Council of Taiwan (R.O.C.) under grants NSC 96-2221-E-011-058 and NSC 97-2221-E-011-064. Ted Knoy is appreciated for his editorial assistance.

References

[1] J. Goldman, Facebook Cookbook: Building Applications to Grow Your Facebook Empire, O'Reilly, 2008.

[2] R. Bhatti, E. Bertino, and A. Ghafoor, "An integrated approach to federated identity and privilege management in open systems," Commun. ACM, vol.50, no.2, pp.81–87, 2007.

[3] E. Maler and D. Reed, "The venn of identity: options and issues in federated identity management," IEEE Security and Privacy, vol.6, no.2, pp.16–23, 2008.

[4] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet x.509 public key infrastructure certificate and certificate revocation list (CRL) profile," RFC 3280, April 2002.

[5] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - ocsp," RFC 2560, June 1999.

[6] B. Shafiq, E. Bertino, and A. Ghafoor, "Access control management in a distributed environment supporting dynamic collaboration," DIM '05: Proc. 2005 Workshop on Digital Identity Management, pp.104–112, New York, NY, USA, 2005.

[7] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen, and T. Scavo, "Security assertion markup language (SAML) v2.0 technical overview," OASIS Committee Draft 02, March 2008.

[8] L. Grewe, OpenSocial Network Programming, Wrox Press, 2009.

[9] E. Hammer-Lahav, "The oauth 1.0 protocol," RFC 5849, April 2010.

[10] C. Rigney, A. Rubens, W. Simpson, and S. Willens, "RFC 2138: remote authentication dial in user service (RADIUS)," April 1997.

[11] J. Kohl and C. Neuman, "RFC 1510: The kerberos network authentication service (v5)," Sept. 1993.

[12] J.D. Clercq, "Single sign-on architectures," Proc. International Conference onInfrastructure Security (InfraSec), Lect. Notes Comput. Sci., vol.2437, pp.40–58, Springer-Verlag, Berlin Germany, 2002.

[13] M.F. Grubb and R. Carter, "Single sign-on and the system administrator," Proc. 12th Conference on Systems Administration (LISA-98), pp.63–86, Berkeley, CA, Dec. 1998.

[14] A. Buecker, W. Filip, H. Hinton, H.P. Hippenstiel, M. Hollin, R. Neucom, S. Weeden, and J. Westman, Federated Identity Management and Web Services Security with IBM Tivoli Security Solutions, IBM, Oct. 2005. Retrived from http://www.redbooks.ibm.com/redbooks/pdfs/sg246394.pdf.

[15] J. Hodges, T. Wason, J. Kemp, and P. Thompson, "Liberty id-ff architecture overview," 2005. Draft of Liberty Alliance Project. Retrieved from http://www.projectliberty.org/specs/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf.

[16] M.R. Thompson, A. Essiari, and S. Mudumbai, "Certificate-based authorization policy in a pki environment," ACM Trans. Inf. Syst. Secur., vol.6, no.4, pp.566–588, 2003.

[17] S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, C. Kaler, H. Lockhart, H. Maruyama, A. Nadalin, N. Nagaratnam, A. Nash, H. Prafullchandra, and J. Shewchuk, "Web services federation language (ws-federation)," July 2003. Specification Version 1.0.

[18] OpenID Foundation, "Openid authentication 2.0," Dec. 2007. OpenID Specification.

[19] V. Bertocci, G. Serack, and C. Baker, Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities, Addison-Wesley, Dec. 2007.

[20] S. Carmody, "Shibboleth overview and requirement," Shibboleth Working Group Overview and Requirements Document, 2001. Shibboleth Working Group, Retrieved from http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-requirements-01.html.

[21] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey, "Identity federation and attribute-based authorization through the globus toolkit, shibboleth, gridshib, and myproxy," Proc. 5th Annual PKI R&D Workshop, 2006.

[22] J. Daiziel and E. Vullings, "Mams and middleware: the easily solved authentication, authorisation, identity, single-sign-on, federation, trust, security, digital rights and automated access policy cluster of problems," EDUCAUSE 2005, 2005.

[23] G. López, O. Cánovas, A.F. Gómez-Skarmeta, and J. Girao, "A swift take on identity management," Computer, vol.42, no.5, pp.58–65, 2009.

[24] F. Paci, R. Ferrini, A. Musci, K.S. Jr., and E. Bertino, "An interoperable approach to multifactor identity verification," Computer, vol.42, no.5, pp.50–57, 2009.

[25] W.A. Alrodhan and C.J. Mitchell, "A client-side cardspace-liberty integration architecture," IDtrust '08: Proc. 7th symposium on Identity and trust on the Internet, pp.1–7, New York, NY, USA, 2008.

[26] P. Madsen and H. Itoh, "Challenges to supporting federated assurance," Computer, vol.42, no.5, pp.42–49, 2009.

[27] F. Almenárez, P. Arias, A. Marín, and D. Díaz, "Towards dynamic trust establishment for identity federation," EATIS '09: Proc. 2009 Euro American Conference on Telematics and Information Systems, pp.1–4, New York, NY, USA, 2009.

[28] M.T. Goodrich, R. Tamassia, and D.D. Yao, "Notarized federated id management and authentication," J. Comput. Secur., vol.16, no.4, pp.399–418, 2008.

[29] D.W. Chadwick and G. Inman, "Attribute aggregation in federated identity management," Computer, vol.42, no.5, pp.33–40, 2009.

[30] M. Dabrowski and P. Pacyna, "Distributed identity discovery service for non-federated systems," MoMM '08: Proc. 6th International Conference on Advances in Mobile Computing and Multimedia, pp.409–413, New York, NY, USA, 2008.

[31] P. Rodriguez and S. Sibal, "SPREAD: scalable platform for reliable and efficient automated distribution," Proc. 9th International World Wide Web Conference on Computer Networks: The International Journal of Computer and Telecommunications Netowrking, pp.33–49, Amsterdam, The Netherlands, 2000.
[32] A.S. Tanenbaum and M. van Steen, Distributed Systems: Principles and Paradigms, Prentice-Hall, 2002.
[33] P. Triantafillou and C. Neilson, "Achieving strong consistency in a distributed file system," Softw. Eng., vol.23, no.1, pp.35–55, 1997.
[34] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," DIM '06: Proc. Second ACM Workshop on Digital Identity Management, pp.11–16, New York, NY, USA, 2006.
[35] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.

Shi-Cho Cha received his B.S. and Ph.D. in Information Management from the National Taiwan University in 1996 and 2003. He is currently an assistant professor at the Department of Information in the National Taiwan University of Science and Technology, where he has been a faculty member since 2006. He is a certified PMP, CISSP, and CISM. From 2003~2006, he was a manager at PricewaterhouseCoopers, Taiwan. His current research interests are in the area information security management, identity management, and RFID privacy.

Hsiang-Meng Chang received his M.S. degree in Information Management from the National Taiwan University of Science and Technology in 2009. His research interests include information security management, identity management, and secured programming.