DATA AND ACCESS MANAGEMENT USING ACCESS TOKENS FOR DELEGATING AUTHORITY TO PERSONS AND SOFTWARE

Hidehito Gomi
Yahoo! JAPAN Research, 9-7-1 Akasaka, Minato-ku, Tokyo 107-6211, Japan

Keywords: Delegation, Access management, Identity federation, Access token.

Abstract: Delegation of authority is an act whereby an entity delegates his or her rights to use personal information to another entity. It has most often been implemented in enterprise environments, but previous studies have focused little on the dynamic data and access management model or the design from a practical viewpoint. A data and access management model for the delegation of authority is proposed. In the proposed model, an access token that is an opaque string associated with authorized permission is issued and exchanged among users and entities across security domains. The framework enables fine-grained access control and permission assignment for delegated access by persons and software agents.

1 INTRODUCTION

With the proliferation of Web services on the Internet, there is an increasing need for providing personalized services for users. With such services, users are often required to share their personal information with other persons or software entities so that they can complete tasks collaboratively in the same way as in the real world. Because personal information is accessed by an entity that is not its owner, flexible access control is essential regardless of whether the entity is a person or a software agent. This falls within the scope of delegating authority to access personal information from person to person and/or to software agent from a technical perspective.

In general, users hope to provide only trusted entities with access to their personal information, and they only do so for a very specific reason. They are reluctant to delegate to unauthorized entities due to security and privacy concerns. Since trusted entities do not automatically have access privileges, it is necessary to flexibly and dynamically assign them. In addition, if we assume that both a person and a software agent can be users, there are a couple variations to the type of access that is delegated. One is a case in which a person delegates direct access of personal information to another person, and the other is a case in which a person delegates indirect access via a software agent. The challenge here is to develop a method of data and access management for delegating authority that works well with the above cases in a distributed environment.

Various models have been proposed to enable flexible and dynamic access management for delegation tasks. Unfortunately, many of these models are conventional in that they assume the delegation takes place in closed environments. Recently there has been some research on dynamism and flexibility when exchanging personal information and assigning permission, but they focus on either an abstract model or a specific protocol design and pay little attention to consolidating both aspects for practical requirements. Moreover, in terms of design viewpoint, they do not pay sufficient attention to supporting both person-to-person and person-to-software delegation.

This paper proposes a data and access management model and design framework for delegating authority between users in an environment that supports identity federation. The proposed system has participating entities exchange an access token that represents the authorization delegation information and is then used to control delegated access.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents scenarios that provided the motivation to attempt this work. Section 4 describes the overview of the proposed system and Section 5 describes the data and access management model. In Section 6, technical issues are discussed. Finally, Section 7 summarizes the conclusions and outlines future work.

Gomi H..
DATA AND ACCESS MANAGEMENT USING ACCESS TOKENS FOR DELEGATING AUTHORITY TO PERSONS AND SOFTWARE.
DOI: 10.5220/0003619504570463
In *Proceedings of the International Conference on Security and Cryptography* (MPEIS-2011), pages 457-463
ISBN: 978-989-8425-71-3

2 RELATED WORK

This section summarizes previous works related to delegation, access control models, and technical specifications.

Research on delegation and access control to enhance the role-based access control (RBAC) model (Sandhu et al., 1996) has frequently been reported (Barka and Sandhu, 2000; Zhang et al., 2003; Wainer and Kumar, 2005; Joshi and Bertino, 2006). When these works are combined with RBAC, they are effective for delegation, especially in a single domain or a closed environment such as an enterprise system, because roles generally have a hierarchical structure reflecting the users' position therein.

Another important line of contribution focused on dynamic and flexible delegation methods in distributed and multiple security domains (Li et al., 2003; Pham et al., 2008; Bussard et al., 2009; Chadwick et al.,2009). These works proposed access control models that support dynamic permission assignment at the point of delegation, which is relevant to the work proposed in this paper. However, this work differs in that it focuses on a practical method and system design for transferring information after delegation authorization beyond security domain boundaries.

Another line of research related to delegation is fine-grained access control models (Hasebe et al., 2010; Cohen et al., 2002). While that research focused on conceptual access control models, this work focuses more on key technologies for dynamic access control and permission assignment from the viewpoint of design and implementation. In the technical approach of SPKI (Ellison et al., 1999) and that proposed by Gomi et al. (Gomi et al., 2005), certificates of authorization rights are distributed and transferred from one person to another. They focused on the specification of the certificates whereas this work specifically introduces an access management model using access tokens.

Some emerging technical specifications provide schemes for exchanging identity information using a token made up of short lengths of string, which is relevant to the work proposed in this paper. SAML (OASIS) works by specifying a scheme for exchanging identity information using an *artifact*, which is a reference to an assertion containing the information. However, the artifacts are not specifically designed for delegation of authority and can only be used by trusted entities, not persons. OAuth (OAuth, 2009) provides a method for delegation from user to application by means of an *access token*. In contrast, this work focuses on a more generic access model and its design framework for user-to-user delegation of authority. In addition, SAML and OAuth do not provide any scheme for controlling token access.

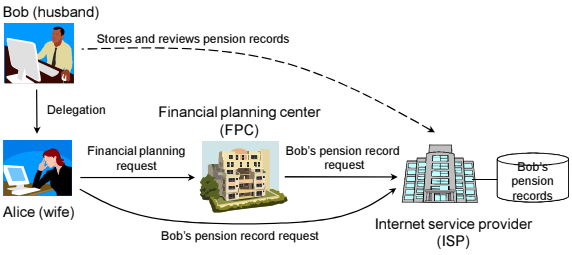


Figure 1: Motivating scenarios.

3 MOTIVATING SCENARIOS

This section describes two scenarios that illustrate the motivation behind this work (Fig. 1). Bob and Alice are a married couple who are working for different companies. They are interested in financial planning to reduce their income taxes, repay their loans, and prepare for their future. Bob has already stored his pension records on his account managed by an Internet service provider (ISP) and views the information privately. Alice would like to view his information by accessing the ISP so that she can consider their planning by herself from their family's perspective. In this case, the ISP needs to identify who Alice is and what she wants in order to ensure an appropriate access control for the request.

Alice is also considering signing up for a financial planning service publicized by a financial planning center (FPC) on the Internet. Such a service would require her to display her and Bob's current financial status to offer an ideal plan for their future lives. In this case, Bob's pension records at the ISP need to be provided to the FPC in a secure and privacy-preserving manner because the FPC can only provide Alice with the service on its site. If we assume that the FPC requests access to Bob's pension records managed by the ISP, the access request needs to be identified by the ISP in the same way as it identified Alice's original request.

In both scenarios, because Alice is the recipient of the services using Bob's pension records—in other words, one person is accessing and using another person's data—Alice's access needs to be authorized by Bob, even though they are married, before the ISP will accept her request for his data regardless of whether it is via the FPC or not. In order to do so, Alice needs to be provided with an appropriate privilege to access Bob's pension records and use the

information for the specific services.

4 SYSTEM OVERVIEW

This section describes the overview of the proposed system supporting delegation.

A **user** is a person who accesses restricted resources provided by other entities. A user has his or her trusted contacts' *contact addresses* (e.g., account identifiers or e-mail addresses) through which they can interact with each other. In this model, there are two types of users: **delegators** and **delegatees**. A delegator is a user who has privileges that can enable others to access his or her identity information. A delegatee is a user who is provided privileges by a delegator to access the delegator's identity information. A delegator and a delegatee have a prior trust relationship and share their contact addresses.

A **service provider (SP)** is an entity that provides access to restricted resources managed by its site to authorized users. An SP supports the delegation of resources containing personal information. This means that an SP may grant a delegatee's access to a delegator's resource on the basis of its security policies if an SP verifies the appropriateness of the delegated access request by the delegatee.

An **identity provider (IdP)** is an authentication and attribute authority that authenticates users by means of particular authentication methods and produces an *assertion* stipulating the completion of the authentication event or the correctness of personal attribute information. In this model, an IdP has an additional *delegation mediating (DM)* capability that supervises the delegation authorization of a delegator to a delegatee and conveys its information to a corresponding SP.

This work proposes introducing a **delegation access token (DAT)**, which is a randomized string representing a reference to a delegated permission to be transferred from a delegator to a delegatee. There are two phases in the overall procedure for completing delegation of authority. Each phase is briefly introduced from the viewpoint of data and access management, which is depicted in Figs. 2 and 3. The solid and dashed lines represent data flow and relations between entities or data, respectively.

Phase 1 (Delegation Authorization Mediation) is shown in Fig. 2.

1. The delegator sends a request to the delegation authorization mediation service (DAMS) at the IdP for mediating authorization for his or her delegation. The request specifies a delegatee, a set of personal attributes, and an entity at which a delegation operation is executed by the delegatee.
2. The IdP makes a decision on whether to grant or deny the request.
3. The IdP generates a DAT and assigns a permission for the delegatee or the SP.
4. The IdP sends the DAT to the delegator if the request is granted.
5. The delegator transfers the received DAT to the delegatee and informs the delegatee.

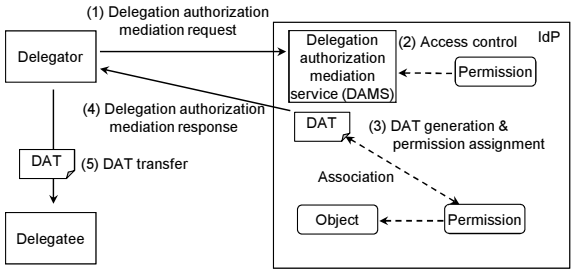


Figure 2: Delegation authorization mediation.

Phase 2 (Delegation Service Provisioning) is shown in Fig. 3.

1. A delegatee attempts to access an identity service (IdS) at an IdP (an SP) by presenting a DAT that he or she has received from the delegator.
2. The IdP (the SP) decides whether to grant or deny the delegated access request. When the SP receives the request (shown in Fig. 3(b)), the SP obtains an assertion of the delegator's personal information from the IdP in the following manner.

2-a. The SP sends an assertion request including the DAT to the assertion issuing service (AIS) at the IdP.

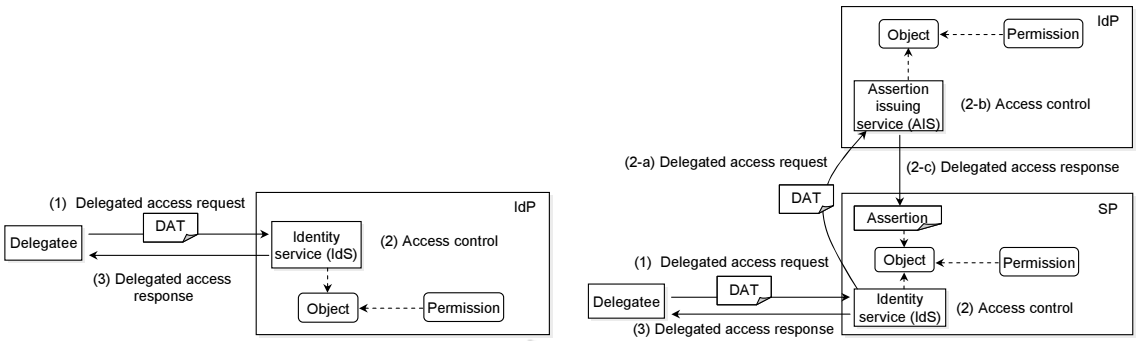
2-b. The IdP makes an access control decision on whether to grant or deny the SP's assertion request by verifying the received DAT.

2-c. The IdP returns the assertion if the SP's request is granted.

3. The IdP (the SP) provides the delegatee with the request delegation service if the above decision is confirmed.

5 DATA AND ACCESS MANAGEMENT MODEL

This section describes the access management model proposed in this work.



(a) Delegation service provisioning (at IdP). (b) Delegation service provisioning (at SP).

Figure 3: Delegation service provisioning.

Objects. An object is a passive entity that contains a set of a user's attribute information. This model denotes an object set O as $O = \{o | o \in A || o \in AS\}$, where A is a set of attributes consisting of pairs of attribute name and value $A = \{a | a = \{(n_1, v_1), \cdots, (n_i, v_i), \cdots\}\}$ if attribute name n_i and its value v_i for a specific user are given. AS is a set of assertions containing multiple attributes a in a format: $AS = \{as | as = \texttt{Assert}(a), a \in A\}$, where function $\texttt{Assert}(a)$ produces an assertion based on the information a of a specific user in a format common to its recipient. For example, Bob, who stores his pension records at an IdP, has the following attribute information: $o = \{(\texttt{BasicPensionNumber}, \texttt{13597})\}$.

Domains. A domain is the scope of entities (IdP or SP) in this model. The components and their relations in a domain are independent of those in other domains. If we let D denote a set of domains, $D = \{IdP, SP\}$. A domain that issues a DAT is called a *token issuer (TI)* and a domain whose service receives an access request with a DAT is called a *token consumer (TC)*.

Operations. An operation is a function that may be executed on an object. An operation opr is represented by $action@domain$, where $action$ is an action performed on an object and $domain$ is the TC at which the action on the object is performed. For example, operation $opr = read, write@SP \in OPR$, which is a set of operations.

Permissions. A permission is a description of the type of authorized operations that a user can perform on an object. $P \subseteq \{(opr, o) | opr \in OPR \wedge o \in O\}$.

Services. A service is an interface of interactions between users and the other system entities (IdP or SP) in a specified format. A service obtains a link to the object on which a user calls an operation when the request is granted. S has two types of services for an IdP and an SP: $S_{\text{IdP}} = \{DAMS, IdS\}$ and $S_{\text{SP}} = \{AIS, IdS\}$.

Constraints. A constraint (C) is a requirement that needs to be satisfied in order to grant an access request. There are separate constraints for DATs and permissions.

The access management model consists of DATs (Tokens for short, T), P, O, S, D, and C, which are shown in Fig. 4. Each component and its relations are

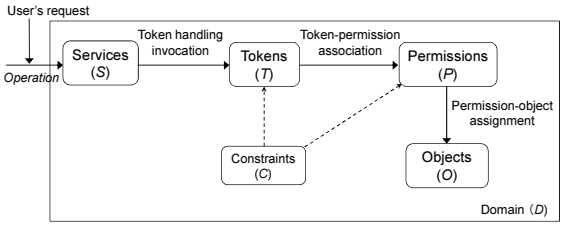


Figure 4: Data and access management model.

abstractly represented together for both phase 1 and 2. Namely, the user's request is a delegation authorization mediation request from a delegator or a delegated access request from a delegatee. The services are at the IdP or the SP in either phase.

5.1 Token-based Permission Assignment

The IdP manages an access control policy to allow a delegator to authorize his or her delegation request in a user-centric way. When an IdP receives a delegation authorization mediation request from a delegator, namely Domain $d = IdP$ and Service $s = DAMS$, and it is authorized, it assigns the appropriate permission

to grant access to the authorized object. This procedure consists of the DAT generation, the object and permission creation, and the DAT and permission association.

Delegation Access Token Generation. After an IdP grants a delegator's delegation authorization mediation request, the IdP generates a unique DAT t. A DAT encapsulates the information of an IdP which generates and issues the DAT. Given a DAT, an IdP and an SP can obtain its issuer information by a predefined method. A DAT also encapsulates its token type information, which is unique to its issuer's domain. The state transition from T to T' when generating a new DAT t is represented by $T' \leftarrow T \cup \{t\}$, where $t \notin T$. Let $t.issuer$ and $t.type$ respectively denote the TI's identifier and the type of DAT t that are available to the token consumer receiving the DAT. Constraints for DATs (*Token Constraints*) are specified by $t.type$ on the basis of how and where the delegator's personal information is to be used.

Object and Permission Creation. If an IdP receives a delegator's authorization mediation request specifying that he or she allows a delegatee to use a set of attribute information, the following object and permission is created:

- $O' \leftarrow O \cup \{o\}$, where $o \notin O$;
- $P' \leftarrow P \cup \{p\}$, if $p \notin P$; and
- $PO' \leftarrow PO \cup \{(p,o) | o \in O\}$.

Token and Permission Association. A DAT generated from the delegation authorization of a delegator and a permission created to perform the delegation are associated with each other to securely control the delegated access. We denote `permission` as a mapping function from T to P; $TP' \leftarrow TP \cup \{(t,p) | p \in P, p = \texttt{permission}(t)$, where $(t,p) \notin TP$. This mapping function will also be used when verifying a DAT attached to a delegated access request (see Sec. 5.2).

Permission Assignment Algorithm. The procedure for permission assignment is shown in Alg. 1, which integrates each operation described above.
First, an IdP (a TI), receives a delegation authorization mediation request and obtains the information of operation and attribute names contained in the request (steps 1–3). In steps 4–12, a set of attributes based on the attribute names received in the above request by retrieving a value of each attribute name for a given delegator u is obtained. After creating a permission for the requested operation and the attribute (step 13), the access decision for delegation authorization is made (steps 14–16), where the operator "$<$" indicates the dominance relation between two permissions. Here, $p' < p$ means that p has more authority than p'. Namely, if any permission's authorization is not included in p, the access request is denied. Next, *authzID* uniquely represents the identifier of the delegator's authorization in the IdP. Using this identifier as well as the issuer and type information, a DAT is generated and registered (steps 17–19). If the specified consumer is not the issuer domain, an assertion is created using attribute a and a new permission for propagating it to the consumer is created (steps 20–25). Finally, the target object is registered, the permission is also registered if its scope is not covered by any existing permission, and an association is made between the DAT and the permission (steps 26–30). If the request is granted, the DAT is returned (step 31).

5.2 Token-based Service Provisioning

When a service receives an access request with an attached DAT, the entity managing the service executes the procedure of access control in which it decides to grant or deny the request. In this model, the procedure consists of verifying the DAT, determining its corresponding permission and object, and enforcing its corresponding constraints.

Algorithm 1: Permission Assignment (at IdP).

Require: delegator u has been authenticated; *issuer*: this IdP;
1: $req \leftarrow$ `receiveDlgAuthZMedRequest()`
2: $opr \leftarrow$ `getOperation`(req)
3: $attrNames \leftarrow$ `getAttrNames`(req)
4: $a \leftarrow \emptyset$
5: **for all** i such that $n_i \in attrNames$ **do**
6: $v_i \leftarrow$ `getAttributeValue`(u, n_i)
7: **if** $v_i =$ **null**, **then**
8: **return false**
9: **else**
10: `add`$(a, (n_i, v_i))$
11: **end if**
12: **end for**
13: $p \leftarrow$ `createPermission`(opr, a)
14: **if** $\exists p' \in P; p' < p$, **then**
15: **return false**
16: **end if**
17: $authzID \leftarrow$ `generateAuthzID()`
18: $t \leftarrow$ `generateDAT`$(authzID, issuer, type)$
19: `add`(T, t)
20: **if** $opr.consumer = issuer$, **then**
21: $o \leftarrow a$
22: **else**
23: $o \leftarrow$ `Assert`(a)
24: $p \leftarrow$ `createPermission`$(provision@opr.consumer, o)$
25: **end if**
26: `add`(O, o)
27: **if** $\forall p' \in P; p' < p$, **then**
28: `add`(P, p)
29: **end if**
30: `associate`(t, p)
31: **return** t

Here, if the combination of a domain and a service is denoted by (d,s), $(d,s) = (IdP,IdS),(IdP,AIS)$, and (SP,IdS). This model deals with all three cases together. If the domain is either an IdP or SP, the TI is the IdP for all three cases, while for $(d,s) = (SP,IdS)$, the TC is the SP.

Token Verification. When an entity (an IdP or an SP) receives a delegated access request with an attached DAT, the entity determines whether or not it is correct. If a received DAT t does not have a predefined format or a fixed length, t is incorrect because it has been inappropriately forged or modified. Even if the above condition is true, if $t.type$ or $t.issuer$ is incorrect, t is incorrect. Otherwise, t is correct.

Permission and Object Discovery. After a DAT has been verified, the above entity retrieves and checks which entity issued it by referring to the information of $t.issuer$. If the TI is the entity itself, it attempts to determine the permission and the object corresponding to the DAT in its local domain by means of the mapping function, $\texttt{permission}(t)$. If the above function does not drive any existing permission, the entity determines that the access request is invalid because the DAT does not have an appropriate association with a permission. If the TI is not the entity, it attempts to retrieve an assertion by presenting it to the TI.

Constraint Enforcement. The proposed model supports dynamic and flexible permission assignment because a delegatee can access personal information as long as he or she presents a correct DAT. To prevent inappropriate access and strengthen the proposed token-based access control, a TI can specify constraints for permission (*permission constraints*) that place some restrictions and conditions on the permission of delegating authority.

Token-based Access Control Algorithm. The procedure of a TC controlling delegated access is shown in Alg. 2, which integrates the above operations.

In steps 1–3, the TC receives a delegate access request and then obtains its operation and DAT. In steps 4–6, the DAT is verified. In step 8, the permission associated with the DAT is obtained. If an expired permission is discovered, it is deleted from the permission set (steps 11–13). In steps 17–25, the TC sends a delegated access request recursively to the TI, retrieves an assertion, and assigns the permission. In steps 26–31, the procedure for constraint enforcement is performed. If any constraints associated with permission are not satisfied, the access request is denied. This procedure returns the authorized object if the request is granted. In this way, the scheme provides a delegation service that is consistent with the three types of interactions in Fig. 3.

6 DISCUSSION

This section discusses several security issues. The proposed model is considered to complement existing access control models such as RBAC. DATs and the roles used in the RBAC model are relevant in that these are intermediaries to permissions. Therefore, more investigation is needed to integrate these models. In this work, the focus was not on how a delegator distributes a DAT to his or her delegatee in a secure fashion. However, this is a fundamental issue relevant to how two entities share their secret with one another, and it remains an open issue.

Algorithm 2: Token-based Access Control.

Require: TC $thisEntity \in \{IdP,SP\}$; token type: $type$.
1: $req \leftarrow \texttt{receiveDelegatedAccessRequest}()$
2: $opr \leftarrow \texttt{getOperation}(req)$
3: $t \leftarrow \texttt{getDAT}(req)$
4: **if** $t =$ **null** or $\texttt{verified}(t) =$ **false**, **then**
5: **return null**
6: **end if**
7: **if** $t.issuer = thisEntity$, **then**
8: $p \leftarrow \texttt{permission}(t)$
9: **if** $p \notin P$, **then**
10: **return null**
11: **else if** $\texttt{expired}(p) =$ **true**, **then**
12: $\texttt{del}(P,p)$
13: **return null**
14: **else**
15: $o \leftarrow p.o$
16: **end if**
17: **else** $\{t.issuer \neq thisEntity\}$
18: $res \leftarrow \texttt{sendDelegatedAccessRequest}(t,t.issuer)$
19: $as \leftarrow \texttt{getAssertion}(res)$
20: $o \leftarrow \texttt{getObject}(as)$
21: $p \leftarrow \texttt{createPermission}(opr,o)$
22: **if** $\forall p' \in P; p' < p$, **then**
23: $\texttt{add}(P,p)$
24: **end if**
25: **end if**
26: $constraints \leftarrow \texttt{getConstraints}(p)$
27: **for all** c such that $c \in constraints$ **do**
28: **if** $\texttt{satisfied}(c) =$ **false**, **then**
29: **return null**
30: **end if**
31: **end for**
32: **return** o

7 CONCLUSIONS AND FUTURE WORK

The data and access management model proposed in this paper supports the delegation of authority from user to user and/or software agent using access tokens. Algorithms and a design for token-based permission assignment and access control enable dynamic delegation of authority with limited overhead cost for managing the association between an access token and its corresponding permission. Our future work will include further investigation into the integration of the proposed model and other relevant models.

REFERENCES

Sandhu, R., Coyne, E., Feinstein, H., Youman, C. (1996). Role-based access control models. IEEE Computer 38–47

Barka, E., Sandhu, R. (2000). Framework for role-based delegation models. In *Proceedings of the 16th Annual Computer Security Applications Conference*. 168–176

Zhang, L., Ahn, G. J., Chu, B. (2003). A rule-based framework for role-based delegation and revocation. ACM Transactions on Information and System Securiry, 404–441.

Wainer, J. and Kumar, A. (2005). A fine-grained, controllable, user-to-user delegation method in RBAC. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies (SACMAT'05)*. 59–66

Joshi, J. and Bertino, E. (2006). Fine-grained role-based delegation in presence of the hybrid role hierarchy. In *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT'06)*. 81–90

Li, N. and Grosof, B. and Feigenbaum, J. (2003) Delegation logic: a logic-based approach to distributed authorization. ACM Transactions on Information and System Securiry 128–171

Pham, Q. and Reid, J.and McCullagh, A. and Dawson, E. (2008) Commitment issues in delegation process. In: Proceedings of the 6th Australasian Information Security Conference (AISC'08). 27–38

Bussard, L. and Nano, A. and Pinsdorf, U. (2009) Delegation of access rights in multi-domain service compositions. Identity in the Information Society, 2, 137–154

Chadwick, D. W. and Otenko, S. and Nguyen, T. (2009) Adding support to XACML for multi-domain user to user dynamic delegation of authority. International Journal of Information Security, 8, 137–152

Hasebe, K. and Mabuchi, M. and Matsushita, A. (2010) Capability-based delegation model in RBAC. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT'10)* 109–118

Cohen, E. and Thomas, R. and Winsborough, W. and Shands, D. (2002) Models for coalition-based access control (CBAC). In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT'02)*. 97–106

Ellison, C. and Frantz, B. and Lampson, B. and Rivest, R. and Thomas, B. and Ylonen, T. (1999) SPKI certificate theory RFC 2693.

Gomi, H., Hatakeyama, M., Hosono, S., Fujita, S. (2005) A delegation framework for federated identity management. In *Proceedings of the ACM Workshop on Digital Identity Management (DIM'05)*. 94–103

OASIS (2005) Assertions and protocol for the OASIS security assertion markup language (SAML) v2.0 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

OAuth Core Workgroup: OAuth core 1.0 revision A http://oauth.net/.