Capability-based Access Control Delegation Model on the Federated IoT Network

Anggorojati, Bayu; Mahalle, Parikshit N.; Prasad, Neeli R.; Prasad, Ramjee

Published in:
2012 15th International Symposium on Wireless Personal Multimedia Communications (WPMC)

Publication date:
2012

Document Version
Early version, also known as pre-print

Capability-based Access Control Delegation Model on the Federated IoT Network

Bayu Anggorojati, Parikshit Narendra Mahalle, Neeli Rashmi Prasad, and Ramjee Prasad

Center for TeleInFrastruktur (CTIF)
Aalborg University, Denmark
Email: {ba,pnm,np,prasad}@es.aau.dk

***Abstract*—Flexibility is an important property for general access control system and especially in the Internet of Things (IoT), which can be achieved by access or authority delegation. Delegation mechanisms in access control that have been studied until now have been intended mainly for a system that has no resource constraint, such as a web-based system, which is not very suitable for a highly pervasive system such as IoT. To this end, this paper presents an access delegation method with security considerations based on Capability-based Context Aware Access Control (CCAAC) model intended for federated machine-to-machine communication or IoT networks. The main idea of our proposed model is that the access delegation is realized by means of a capability propagation mechanism, and incorporating the context information as well as secure capability propagation under federated IoT environments. By using the identity-based capability-based access control approach as well as contextual information and secure federated IoT, this proposed model provides scalability and flexibility as well as secure authority delegation for highly distributed system.**

***Index Terms*—capability-based access control, delegation, security, IoT**

I. Introduction

Security and privacy are the two key elements in providing trust and allowing successful operation of IoT. One way to enable security and privacy is to implement access control, which covers both authentication and authorization. On the other hand, IoT is characterized by highly dynamic nodes connectivity and network topologies due to the ever-changing nature of wireless channel, mobility, and factors such as limited power that might cause a node to die out. To this end, a dynamic and flexible design of access control system that is suitable for IoT is of the most importance.

First of all, in order to cope with the restriction in such a system like IoT, a lightweight access control model needs to be introduced. For this purpose, we introduce a secure Capability based Context Aware Access Control (CCAAC) model that is suitable for highly pervasive and ubiquitous system such as IoT. The main idea of the model lies in the principle of identity-based capability access control where capability becomes the central point on access control mechanism and an identifier that is used to increase the scalability and control the capability propagation [1]. Moreover, the model is combined with context awareness to accommodate dynamic access policy enforcement.

Secondly, in order to achieve dynamic and flexible access control system, various models of delegation of authority techniques in access control have been proposed. Some of those proposed methods addressed the delegation issue within federated identity management environment [2], [3], and also challenges in cross-domain delegation by specifically using capability- based access control [4], [5]. However, those delegation methods are designed mainly to serve web-based services which involve a large IT infrastructure. Those kinds of delegation models are not practically visible for an IoT system, which has a lot constraint in its resource, e.g. memory and power. Hence, we propose a delegation of authority method based on dynamic capability propagation suited for pervasive system such asIoT.

In the identity-based capability, the subject's identifier is included in the capability, which makes it able to authenticate itself upon its effort to gain access or authorization of a certain object. Moreover, it is important to mention that the delegation of authority by means of capability propagation is part of CCAAC overall design model. Therefore, delegation method in CCAAC is not an extension of any existing access control model, e.g. Role Based Access Control (RBAC), as presented in most of the previous works. The contributions of the paper include: providing a federated IoT model as a baseline for the entire proposed delegation model, and further defining the delegation model along with a protocol description and security considerations which incorporate identity-based capability and contextual information.

The rest of this paper is organized as follows. Related works in this area is presented in section II. A brief introduction and definition of the CCAAC model are presented in section IV. Section V explains the access delegation mechanism based on our proposed CCAAC model along with some security considerations. Finally, conclusion and future works are given in section VI.

II. Related Works

Research on delegation of authority using capability has been investigated in [4] and [5]. [4] addressed the issue of role and/or permission delegation based on a RBAC model in a cross-domain environment using capabilities. The central idea behind their proposed mechanisms was the mapping of capabilities into roles and permissions in each domain. [5]

extends [4] by adding the delegation of task to be performed in the model for workflow systems.

Some works focusing on dynamic and flexible delegation methods in distributed and multiple security domains have been reported in [2] and [3]. [2] focused on dynamic authorization delegation in federated environment between entities or machine-to-machine delegation. Furthermore, it investigated chain of delegation in multiple entities and how to provide secure delegation framework. [3], on the other hand, focused more on the user-to-user delegation and did not consider multiple entities delegation. Its main contribution was a delegation framework in federated environment using an access token, regardless of the access control model being used. Furthermore, [3] explains the mechanism of issuing a token, asserting it into an authorization document, and service provisioning based on the delegate token.

On the other side of the table, federated IoT networking has not been much discussed in the literature. However, [6] has defined a complete set of networking, management, and security framework for device-to-device or machine-to-machine communication in the context of a Personal Network (PN). Federated IoT environments based on the PN concept [6] will be further elaborated in the next section.

III. Federated-IoT

Identity "Federation" is a pretty popular term within the web security world and refers to management of a web user's identity across different security domains. The main reason of enabling federation in the web environment is that the work flow of the system often requires a user that is authenticated in one domain to be authenticated in other domains as well. However, a concrete definition of the Federated IoT which is conceptually different as compared to the web security world, needs to be determined before addressing the issue of authority delegation in such environment.

First of all, the identity in the web-based system refers to a person's identity while in IoT, identity refers to a device or "thing". Therefore, the interaction of identities in IoT is in the form of device-to-device communication. Some research direction on device-to-device federated network along with its security framework has been done in the context of PN [6]. Although PN [6] did not address IoT directly, its networking concept, especially the federated network, is very relevant to our purpose. Moreover, its security framework could provide a foundation for this work to further extend it to our access control model along with the model of authorization delegation.

An example of federated IoT network based on the concept of PN-Federation is depicted in Fig. 1. Three IoT network domains are considered, i.e. private user, retail shop, and goods producer network, and two IoT-Federated networks are considered. One IoT network domain consists of one or more IoT cluster and the inter-cluster communication can be done either through the internet infrastructure as shown in Fig. 1 or through a wireless ad-hoc connection. Device-to-device communication within a cluster, i.e. intra-cluster communication,

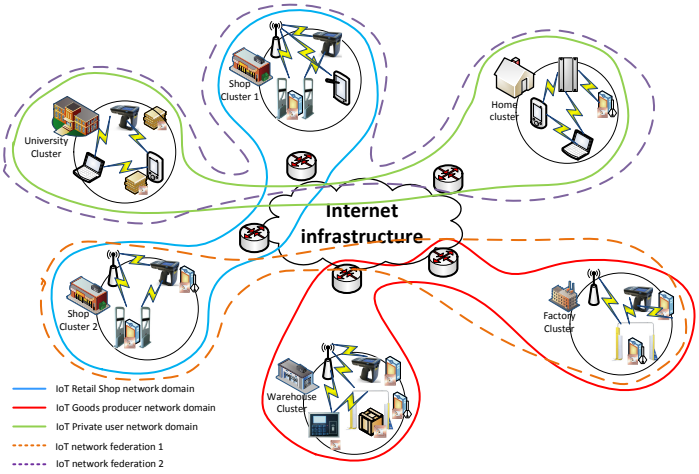

Fig. 1. An example of Federated IoT network with delegation scenario

can be carried out by using different wireless access technology, e.g. Radio Frequency Identification (RFID), IPv6 over Low-power Wireless Personal Area Network (6LoWPAN), ZigBee, bluetooth, wifi, etc.

IV. CCAAC Model

This section describes the basic concept as well as definitions of CCAAC model. This consists of a system architecture for supporting CCAAC, a proposed capability structure, and some important definitions in the CCAAC model.

A. System architecture for supporting CCAAC

The system architecture for supporting the CCAAC model is depicted in Fig. 2. This system architecture is adapted from Context Aware Security Manager (CASM), which is part of PN's security framework introduced in [6]. It is important to note that the PN has been referred in this work due to its advance networking concept for device-to-device communication that opens a path as one candidate of network implementation in IoT. Correspondingly, the security framework brought up within the PN would be a good starting point in designing a security framework in IoT, considering their similarities in characteristics and requirements.

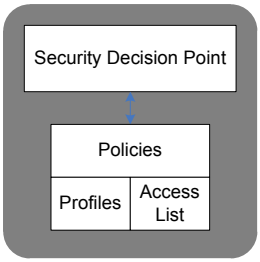

Fig. 2. System architecture for supporting CCAAC

Policies in Fig. 2 serves as Policies Repository that consists of a collection of various policies for accessing available resources or objects. *Profiles* serves as Profile Repository, which essentially consists of subject as well as object profiles. Both of these components will be referred to as Policies

Repository and Profiles Repository later on in Section V. Finally, *Access List* plays an important role in supporting the capability-based authority delegation of access control by controlling the capability propagation as well as revocation through maintenance of a propagation tree as proposed in [1].

B. Proposed capability structure

Our proposed capability structure is an extension to the Identity-based capability model [1], e.g. Identity based Capability (ICAP), which extends the classical capability structure [7]. In order to support the context awareness in ICAP, an additional field called *Contexts* ($\mathcal{C}$), which contains context information related to the capability, is added in the ${}_{ext}CAP$ for subject i (S_i). By including this field, the external capability structure in CCAAC is defined as:

$$ {}_{ext}CAP_i = \{\mathcal{O}, \mathcal{AR}, \mathcal{C}, Rnd_i\} \tag{1} $$

where

$$ Rnd_i = f(S_i, \mathcal{O}, \mathcal{AR}, Rnd_0) \tag{2} $$

$$ Rnd_0 = f(\mathcal{O}, \mathcal{AR}) \tag{3} $$

- S_i: Representing identifier of $Subject\ i$ that requests an access.
- $\mathcal{O}$: Name of object or resource to be accessed.
- $\mathcal{AR}$: Type of access right, e.g. read, write, execute.
- $\mathcal{C}$: Context information.
- Rnd: Random number generated from a one-way hash function to prevent forgery.

The internal capability (${}_{in}CAP$) that creates a pair with the ${}_{ext}CAP$ which is stored in the object itself or an entity that has higher "authority" over the object (e.g. in hierarchical type of network). It is defined as follows:

$$ {}_{in}CAP = \{\mathcal{O}, Rnd_0\} \tag{4} $$

where Rnd_0 is defined exactly as in Equation 3.

C. Basic definitions

The important definitions used in CCAAC are presented first by assuming a PN to be the targeted platform as the important components of a Virtual Identity (VID) (the profile and the context). However, VID can be defined in different ways depending on the target platform.

1) Definition 1 (VID): Conceptually speaking, an entity, e.g. subject or object, may have more than one identity, namely one main identity and numbers as other alias identities. Each identity that is associated with an entity is referred as VID, thus an entity may have multiple VIDs. On the other end, a VID can be linked to a set of policies where the same policy can apply to different VIDs. Therefore, the relationship between VIDs and disclosure policies is a many-to-many, which can be implemented using a pointer or hash map. Additionally, a VID consists of an identifier and is attached with a particular context as well as profile information of the corresponding entity and it can be assumed that the profile information can be pre-defined as a set of default profiles or customized to a specific VID. In any case, the profile is assumed to have an one-to-one relationship with the VID and, for the context, to have a many-to-many relationship with the VID. A more detailed explanation of context will be presented in the definition of contexts later on in this subsection.

Based on these relationships and assumptions, the VID is defined as follows:

$$ VID \in \{ID, \mathcal{P}, \mathcal{C}, Policies\} \tag{5} $$

The $Profile\ \mathcal{P}$ in VID may consist of objects' attributes and personal information. $\mathcal{C}$ refers to $Contexts$ which can be a security context, such as trust level or authentication level, as well as other contexts, such as time and location. The definition and detail information of $Contexts\ \mathcal{C}$ will be given later in this section. The ID is a unique identifier that can be acquired through cryptographic operations, while the $Policies$ is a set of $Policy$ which will be explained in more details in the next sub-section.

2) Definition 2 (Policy): As explained earlier, the policy is essentially associated with certain VID(s) that describes VID(s) preferences upon allowing other entities to access them. Please note that the entity or subject requesting access is described by its profile, e.g. subject's attributes, in the policy. A policy in the proposed CCAAC model holds an important role in access control decision as well as any process involving capability creation and delegation. It can simply be defined as a set of rules with parameters related to the user as seen in the following notation:

$$ Policy \in \{\mathcal{P}, \mathcal{C}, \mathcal{AR}\} \tag{6} $$

It is important to note that since the $Policy$ is linked with a VID, which is then linked with an object, the notation of the object or resource is not included in the $Policy$. On another note, unlike the definition of VID, the $\mathcal{P}$ and $\mathcal{C}$ that are included in the $Policy$'s rule are related to the subject who are trying to gain access to the object.

3) Definition 3 (Contexts): The *Contexts* $\mathcal{C}$, that is used to define the VID and $Policy$ is basically a set of contexts ($\mathcal{C}_{Set}$) with different types ($\mathcal{C}_{Type}$). The type of context can be a concrete property such as time or location, but also security-related context such as authentication and trust level. In order to apply the context in the access control decision, each of the context types has to be evaluated with a certain constraint ($\mathcal{C}_{Const}$).

The overall context definition in CCAAC can be expressed with the following notation:

$$ \mathcal{C}_{Type} \in \{authLevel, trustLevel, time, location, \cdots\} \tag{7} $$

$$ \mathcal{C}_{Set} = \{\mathcal{C}_{Type(1)}, \mathcal{C}_{Type(2)}, \cdots, \mathcal{C}_{Type(n)}\} \tag{8} $$

$$ \mathcal{C}_{Const} := \langle \mathcal{C}_{Type} \rangle \langle OP \rangle \langle VALUE \rangle \tag{9} $$

where OP is a logical operator, i.e. $OP \in \{>, \geq, <, \leq, =, \neq\}$, and $VALUE$ is a specific value of $\mathcal{C}_{Type}$. Finally, we define $\mathcal{C}$ as a set of context constraint $\mathcal{C}_{Const}$ as follows:

$$\mathcal{C} = \{\mathcal{C}_{Const(1)}, \mathcal{C}_{Const(2)}, \cdots, \mathcal{C}_{Const(n)}\} \quad (10)$$

4) Other definitions: : Other definitions that are used in the formal specification of CCAAC are as follow:

$$\mathcal{P} = \{Profile_1, Profile_2, \cdots, Profile_n\} \quad (11)$$

$$Policies = \{Policy_1, Policy_2, \cdots, Policy_n\} \quad (12)$$

$$\mathcal{AR} \in \{Read, Write, NULL\} \quad (13)$$

AR can either be $\{Read\}$, $\{Write\}$, $\{Read, Write\}$, or $\{NULL\}$. If $AR = \{NULL\}$, the permission to access a particular object is not allowed.

V. Delegation model

Based on short description of the IoT-Federated network in Section III and PN concept in [6], a high level delegation model in Federated-IoT environment will be presented in this section. This is followed by a delegation mechanism using capability propagation based on the CCAAC model.

A. High level delegation model

To support a federation network in IoT, an entity called IoT Federation Manager (IoT-FM) is introduced. IoT-FM is responsible for managing the participation of an IoT network domain in a federation by having some corresponding rules and policies. In our proposed authority delegation model, an IoT-FM has an additional functionality that is to authorize the delegation request from a *delegator* and grant it to the *delegatee*. *Delegator* is an entity that delegates some or all of its authority to another entity, while *delegatee* is an entity that receives an authority delegation from the *delegator*. Fig. 3 shows our proposed high level delegation model in federated-IoT.

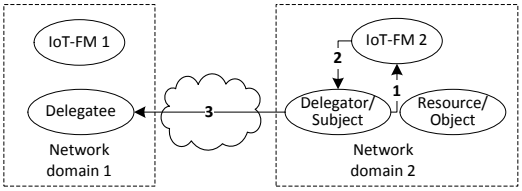

Fig. 3. High level delegation model on Federated-IoT

Please note that in CCAAC notation, we defined terms such as *Subject* and *Object*. Essentially, any *Subject* can be either *Delegator* or *Delegatee*. However, *Delegator* will be referred to as *Subject* ($\mathbf{S}$) for the sake of the protocol explanation in the rest of this paper. In another note, the resource to be accessed by *Subject* is referred as *Object* ($\mathcal{O}$) as defined in CCAAC.

Furthermore, Fig. 3 also shows a high level delegation from $\mathbf{S}$ to the *delegatee* ($\mathbf{D}$). In this case we assume that trust relationship has been established between two network domains when the federated network is created [6] through some mutual authentication mechanism between two domains. Therefore, $\mathbf{S}$ would send a delegation request signed with a shared secret key between $\mathbf{S}$ and IoT-FM 2 upon requesting its authority delegation towards $\mathbf{D}$ (step 1 in Fig. 3). Upon receiving the delegation request from $\mathbf{S}$, IoT-FM 2 would verify the signature with its pair key and then evaluate the delegation request based on some available policies which will be further explained in the next subsection. In case of a positive delegation request evaluation result, a delegation request response in a form of external capability ($_{ext}CAP_D$) with $\mathbf{D}$'s identity would be sent to $\mathbf{S}$, otherwise an error message would be sent instead (step 2 in Fig. 3). Finally, the $\mathbf{S}$ would send the $_{ext}CAP_D$ encrypted with a public key that is known by $\mathbf{D}$ as a result of trust relationship when federated network between two domains was established (step 3 in Fig. 3).

B. Delegation mechanism based on CCAAC

As described in previous subsection that IoT-FM would evaluate the delegation request upon receiving it and would further decide whether to grant the authority delegation to $\mathbf{D}$ or not. The whole process of the delegation mechanism along with the delegation request evaluation in IoT-FM is depicted in Fig. 4.

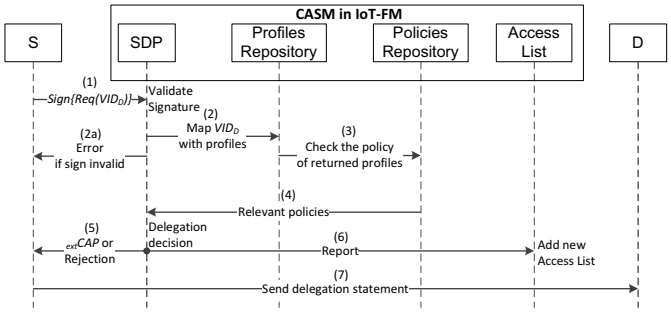

Fig. 4. Capability propagation protocol for authority delegation in our proposed access control

It is important to mention that Fig. 4 is the micro-level view of Fig. 3 where $\mathbf{S}$ and IoT-FM belong to Network Domain 2, and $\mathbf{D}$ belongs to Network Domain 1. Detailed explanations in the delegation mechanism depicted in Fig. 4 is presented as follows:

1) Sending authority delegation request: Authority delegation request is being sent by $\mathbf{S}$ to the Security Decision Point (SDP) within IoT-FM. The request message is signed with a shared key between $\mathbf{S}$ and IoT-FM so that IoT-FM is able to make sure the message is indeed sent by $\mathbf{S}$ and the integrity is maintained. Please note that the type of shared-key, i.e. either symmetric or asymmetric, and the specific encryption algorithm being used are not within the scope of this work.

2) Mapping the VID_D to Profile: The SDP checks the message's signature. If the signature is valid, the SDP then asks VID-Profiles mapping box to map the profiles of $\mathbf{D}$ given VID_D. It will return the profile of $\mathbf{D}$.

3) Check the relevant policies: The returned Profiles $\mathcal{P}$, together with the $\mathcal{C}$ and VID_O, are then sent to the Policies

Repository, to check the disclosure policies of the corresponding Object (based on its VID).

4) Return the relevant policies: The Policies Repository gets all the relevant policies from the given $\mathcal{P}$ and $\mathcal{C}$ of the object or resource of interest represented by its VID_O, and then gives them to the SDP.

5) Delegation decision: The SDP combines the received policies with a policy-combining algorithm and comes up with a decision whether to approve the authority delegation by creating a new capability (CAP) for $\mathbf{D}$ or not. In the case of a positive decision, the SDP creates a delegation statement in the form of ${}_{ext}CAP$ for the $\mathbf{D}$ and then sends it to $\mathbf{S}$. More specifically, the only difference between the newly created ${}_{ext}CAP$ and the one that is owned by the $\mathbf{S}$, lies in the $\mathbf{D}$'s identifier $\mathcal{D}$ within the Rnd_i component, that is used instead of the $\mathcal{S}$. In the case of a negative decision, a rejection message will be sent to $\mathbf{S}$ instead.

6) Update propagation tree: In parallel to sending the delegation decision, the SDP sends a report regarding the CAP creation of an object for Subject i, S_i, to the Access Control Servers (ACS) which will be followed by the creation of a new propagation tree.

7) Sending authority delegation statement: Finally, $\mathbf{S}$ sends the authority delegation statement in the form of ${}_{ext}CAP$ particularly for $\mathbf{D}$. Moreover, in order to maintain the confidentiality and integrity of the ${}_{ext}CAP$, it can be signed with a shared secret key between $\mathbf{S}$ and $\mathbf{D}$, based on an assumption that both domains have established a trust relationship by authenticating each other through a certain key pair.

Steps 2 trough 5 of the delegation mechanism as described in the above explanation can be expressed in the pseudo-code as presented in Algorithm 1.

Algorithm 1 Capability delegation decision

procedure DELEGATECAP(VID_D, VID_O)
  $\mathcal{P} \leftarrow getProfiles(VID_D)$
  $\mathcal{C} \leftarrow getContexts(VID_D)$
  $Policies \leftarrow checkPolicies(\mathcal{P}, \mathcal{C}, VID_O)$
  $decision \leftarrow combinePolicies(Policies)$
  if $decision \neq NULL$ **then**
    if $IntCAP = 0$ **then**
      ${}_{in}\mathcal{CAP} \leftarrow createIntCAP(VID_O)$
    end if
    ${}_{ext}\mathcal{CAP} \leftarrow createExtCAP(VID_D, \mathcal{AR}, VID_O)$
  end if
end procedure

First of all, it is assumed that $\mathbf{S}$ as the *delegator* knows the identity of the $\mathbf{O}$ and $\mathbf{D}$. This is possible when the *delegator* $\mathbf{S}$, subscribes to a service, in which a device in the service provider's domain needs to be given an authority delegation, i.e. as *delegatee* $\mathbf{D}$, in order to access a device or resource within the *delegator*'s network domain, i.e. $\mathbf{O}$. With this assumption, $\mathbf{S}$ needs to submit a delegation request by stating the identities of $\mathbf{D}$ and $\mathbf{O}$ in the form of VID_D and VID_O, respectively. Once submitted, *delegatee*'s Profile ($\mathcal{P}$) and Context ($\mathcal{C}$) can be obtained from VID_D as they are attached to it (see Equation 5). Afterwards, all relevant policies related to VID_O that contain *delegatee*'s $\mathcal{P}$ and $\mathcal{C}$ are gathered from the Policies Repository to be further evaluated by a certain Policies Combining Algorithm to obtain a delegation decision.

VI. Conclusion

Authority delegation is an important mechanism to support dynamic and flexible access control. It is mainly challenging to design such a delegation in access control for Federated IoT due to its dynamic and distributed nature. In this paper, we presented our definition of the Federated IoT, which thus far has not been particularly discussed in the literature. Furthermore, the definition of Federated IoT is used as a baseline in designing our proposed access control model along with authority delegation mechanism that incorporates identity-based capability and dynamic context information. The protocol description and security consideration involving the usage of cryptographic keys are further presented in the paper to give some guidelines in the practical implementation.

A possible future work is to incorporate a secure authority delegation method based on CCAAC of the proposed model and then evaluate its security effectiveness. Furthermore, other directions of this work also involves the extension of an authority delegation design along with verification and implementation, considering that no prior knowledge of the trust relationship between two network domains in Federated IoT. Unlike the approach used in this paper, an additional entity that is trusted by both domains, for instance Identity Provider (IdP), needs to be involved in the design. Another interesting direction would be incorporating our proposed access control as well as delegation model with an auto-delegation mechanism as presented in [8].

References

[1] L. Gong, "A secure identity-based capability system," in *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, may 1989, pp. 56 –63.

[2] H. Gomi, M. Hatakeyama, S. Hosono, and S. Fujita, "A delegation framework for federated identity management," in *Proceedings of the 2005 workshop on Digital identity management*, ser. DIM '05. ACM, 2005, pp. 94–103.

[3] H. Gomi, "Dynamic identity delegation using access tokens in federated environments," in *Web Services (ICWS), 2011 IEEE International Conference on*, july 2011, pp. 612 –619.

[4] K. Hasebe, M. Mabuchi, and A. Matsushita, "Capability-based delegation model in rbac," in *Proceeding of the 15th ACM symposium on Access control models and technologies*, ser. SACMAT '10. New York, NY, USA: ACM, 2010, pp. 109–118.

[5] K. Hasebe and M. Mabuchi, "Capability-role-based delegation in workflow systems," in *Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on*, dec. 2010, pp. 711 –717.

[6] R. Prasad, *My personal Adaptive Global NET (MAGNET)*, ser. Signals and Communication Technology Book. Springer Netherlands, 2010.

[7] H. M. Levy, *Capability-Based Computer Systems*. Butterworth-Heinemann, 1984.

[8] J. Crampton and C. Morisset, "An auto-delegation mechanism for access control systems," in *Proceedings of the 6th international conference on Security and trust management*, ser. STM'10. Berlin, Heidelberg: Springer-Verlag, 2011.