Study on Access Permission Control for the Web of Things

Se Won OH*/**, Hyeon Soo KIM*

*Department of Computer Science & Engineering, CNU (Chungnam National University), Yuseong, Daejeon, 305-764, Korea

**ETRI (Electronics and Telecommunications Research Institute), Yuseong, Daejeon, 305-700, Korea

sewonoh@etri.re.kr, hskim401@cnu.ac.kr

***Abstract*—The Web of Things (WoT) research is exploring ways on the interoperation among the smart things, since the Web has proven its potentials as open communication environment for accommodating a variety of Web resources. The Web technologies has enabled the Web-enabled devices to publish and exchange their resource information over the Web, whereas the Web-enabled devices should cope with the security threat regarding the information exposures over the Web, particularly, access permissions for the resources about the things. Thus, in this paper we analyse access permission control mechanism considering both the WoT characteristics and the REST-compliant resource-oriented Web architecture. In contrast to existing access control logics, the proposed mechanism utilizes not only the requester information such as the typical identity and the internet addresses, but also the context of the thing itself. Based on this mechanism, we present web-resource structure for access permission control, and describe an exemplary procedure in detail. This research contributes to the flexible and decentralized access permission control for WoT.**

***Keyword*—Access Control, REST, Security, Web of Things, Web Resource**

I. INTRODUCTION

THE number of smart things connected to the Internet even exceeds the population of human beings nowadays. As more smart things are capable of data communicating over the Internet, the concepts of the "*Internet of Things* (IoT)" or "*Machine-to-machine* (M2M)" are having been realized in many fields, as in [1]–[8], including smart home, smart meters, remote healthcare, and logistics process automation. Different sources predict that by 2020, as in [9] and [10], the number of connected things would be more than 25 billion, even up to 50 billion with an optimistic view, and the IoT market size will grow up to $7 trillion according to [11] and [12].

Whereas many IoT researches mainly have focused on how to establish connectivity among the networked devices, the *Web of Things* (WoT) or the Web-based IoT have been extensively studied for integrating these smart things with the well-known Web technologies, as in [13]–[15]. The Web-enabled things can interoperate freely over the Web utilizing the open Web standards, since the World Wide Web (WWW, Web) can provide flexible, scalable communication channels for any web clients to share web resources. Also, the scope of the traditional web services can be broadened into the physical-world, not only cyber-world. Moreover, the Web-enabled things can reuse and adopt the proven Web mechanisms such as discovery, searching, browsing, linking, and caching, as in [16]. Further, REST (Representational state transfer) architectural style with URIs (Uniform Resource Identifier), HTTP, and standardized media types, is very promising to make these things to share their data and resources over the Web.

A pressing open problem in this WoT environment is how to allow smart things to grant clients access to their own resources, since the Web-based open environment often leaves information vulnerable to disclosure resulting in security threats such as:

--Malicious clients and unwanted data sharing
--Attacks in any time and from anywhere
--Unpredictable work load and availability risk

Unfortunately, there are few studies on these concerns and several system prototypes just have shown inadequate functionality for security. In particular, from a security point, the existing access control mechanisms for the web resources, as in [17]–[18], are used to require either rigid access policy enforcement or dependency on the pre-configured access management procedures like a central authentication service (CAS) system. And these ways have limitations for dealing with the dynamic interactions and the scalability in the WoT.

Thus we analyzed the access permission control for the resources of web-enabled things, as in our previous research [19]. And a decentralized access permission control mechanism for Web resources for WoT is proposed. The proposed mechanism adopts REST-style resource-oriented architecture for things, in order to enable a thing itself (or its owner) to manage access permissions to its own information resources by means of simple CRUD (Create, Read, Update, and Delete) actions. To explain the mechanism in detail, we described the prototype module of resource access control as

Manuscript received November 28, 2014. This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korea government [13ZC1100 (2013~2015) – Unit Research Project 3, Development of USN/WoT Convergence Platform for Internet of Reality Service Provision, 15ZC1300].

Se Won OH is with the department of Computer Science & Engineering, Chungnam National University, as well as ETRI, Daejeon, 305-700, Korea (e-mail: sewonoh@etri.re.kr).

Hyeon Soo KIM is with the department of Computer Science & Engineering, Chungnam National University, Daejeon, 305-764, Korea (corresponding author; phone: +82-42-821-6657; fax: +82-42-822-9959; e-mail: hskim401@cnu.ac.kr).

well as resource structure XML.

The remainder of this paper is organized as follows: Section 2 presents the related researches and the basic concept of access permission management. Section 3 proposes access permission control mechanism considering both the WoT characteristics and the resource-oriented Web architecture with the exemplary cases in detail. Then, concluding remarks are provided in Section 4.

II. Related Study

This section introduces the basic model for providing access control and discusses the existing researches on access permission management, with describing research challenges coming from the WoT characteristics.

A. Access Control Model

The rudimentary form of access control model in a computer security system suggests that an active subject requests a specific access operation to a passive object, then a reference monitor between them decides whether to grant the access request or not, as shown in Fig. 1.

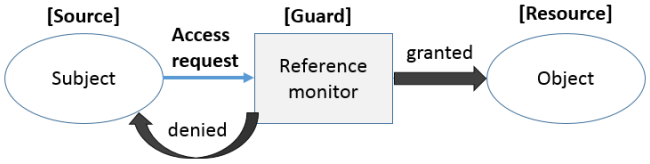


Fig. 1. Basic access control model including subject, object, and reference monitor

In order to protect the information system against malicious actions which could damage data and resources, it is essential to control access requests from subjects [17]. So, the reference monitor as a guard should perform two types of tasks; 1) to identify the subject who made the request, and 2) to decide who is allowed to do what to the object.

Traditional access control systems have usually relied on a local database which can maintain either user passwords, group membership, and/or access control matrices. Then, several modifications have been adapted for the Web security environment, including 1) SSL usage for securing the user channels, 2) checking host information for authentication, and 3) centrally managed database for a domain access control. However, the centralized Web security architecture often imposes limitations like a single point of failure as well as a dependency on the dedicated security configuration.

To assure access control in a distributed system environment considering an ad-hoc security domain for the thing-to-thing communication, as in [20]–[21], it is necessary to handle all the access requests even without asking to the central security manager. Likewise, the number and/or diversity of subjects/objects in the web of things make the access control issues more challenging [13].

B. Existing Access Control Logics in Computer Network

With the advancement of information security, in order to facilitate the management of access permission, many security models for computer network systems have been coming up as the followings, to name just a few:

--Subject/object access control matrix [22],

--Multilevel security using information flow [23],

--Role-base access control (RBAC) [24]–[25], and

--Attribute-based access control (ABAC) [26]–[27].

Here, RBAC represents the concept of *role*, in order to logically relate users and permissions. That is, the RBAC model assigns permissions to roles, and roles to the users. However, this model mainly focuses a kind of 'static' authorization assignment a priori, whereas many practical information systems require more flexible ways to update the subject's permissions. In the meantime ABAC can support changing needs much easier, since only subjects with valid set of attributes are permitted to access the data and this approach do not rely on the pre-defined roles for the specific sets of attributes. Also ABAC supports the notion of distributed access control enforcement, called policy enforcement point.

In addition, in consideration of environmental and context information, a concept of context-aware authorization model has been suggested as in [28]–[29]. And it is recognized that these semantic information can help to specify access control measures and to provide machine-interpretable description of security requirements [30], as the WoT environment enables things to communicate over the Web with disparate parties and users.

III. Access Permission Control for WoT

A. WoT Access Control Requirements

In the WoT environment a thing is expected to have a dynamic connection with other things over the web. That is, each thing may be either subject or object interchangeably in order to perform its own task. Further, according to the REST-style web architecture, each thing may represent itself as web resources which can be identified with unique URIs. According to the aforementioned properties, the access permission control for WoT should meet the following requirements:

1) Each thing may publish its information as one or more web resource(s) over the web.
2) Resources for a thing can be accessed by the basic HTTP/REST request from a subject (e.g. a HTTP client), as outlined in Fig. 2. Since each thing may communicate with other things in the WoT environment, Thing A can request access for the specific resources about the hosting entity, Thing B. Then, Thing B needs to assure the access permission about the requested resources against the Thing A's access request.
3) Permission assignment can be described as a web resource representation, then the decision to grant access for a given request to the specific resource should be made with referring to this type of resource.

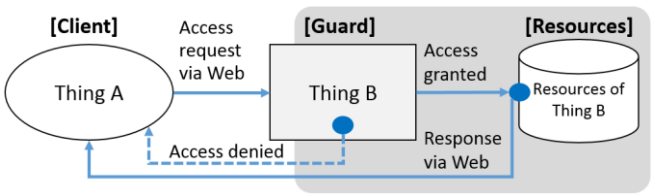


Fig. 2. Control flow about access request and response between two web-enabled things in WoT

B. Resource-oriented Architecture for WoT

Since the WoT makes use of the current resource-oriented Web architecture, a web-enabled thing can exchange a specific resource with the HTTP/REST methods about the CRUD (Create, Read, Update, and Delete) operations. Also, the given resource and its attribute(s) information, if any, can be identified with unique URI, of which situation is described in Fig. 3. The resource representation contains three resource objects (‘a’, ‘b’, and ‘c’) and the relative contents can be accessed with the usual HTTP/ REST CRUD methods. For example, in order to retrieve the content information of object ‘a’, which is hosted at 192.168.0.1, a HTTP request for GET operation should specify the URI of the given object as following: ‘http://192.168.0.1:8080/things/a/contents’. Like any web page or image file which might be linked using URI, WoT resource information could be accessible via Web.

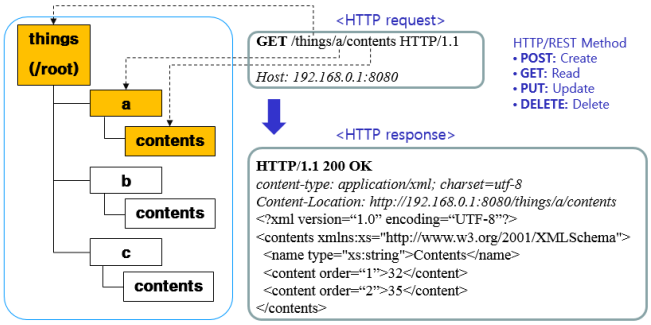


Fig. 3. Exemplary resource representations about Web-enabled things and the associated HTTP/REST request and response

In this situation, we considered that the hosting entity might accommodate a number of resource objects, even resources from the other thing (e.g. /things/b/contents and /things/c/contents) which do not have a HTTP server capability. It is because many smart devices practically provide constrained features as well as limited powers, so these constrained things may exchange its resource information through the support of much powerful entity, called WoT Gateway or WoT Broker [16]. Thus, in this case, the way to make each resource configured with different access permission should be provided.

C. Proposed Mechanism for WoT Access Control

Fig. 4 outlined the detailed steps (i.e., from the Step 1 through the Step 6) to process a request for accessing resources in the WoT. Each steps aims to filter out any incomplete or ambiguous access request, as summarized in TABLE I. Even though it seems the advanced secure channels as well as cryptographic algorithms improve the entire security level for the HTTP protocol communication, we presumed the participant things in the WoT at least supports the standard HTTP protocol basically. For example, if the request could not be interpretable in Step 2 due to any malformed syntax, then the original requestor should receive HTTP response with a ‘Bad Request (400)’ status code according to the HTTP protocol [31]. Likewise, if the requested resource is not found at the hosting entity, ‘Not Found (404)’ HTTP response should be generated.

We recognized that the matter of granting an access request to the specific WoT resource is very closely associated with the Step 5. Here, as shown in Fig 5, we considered that there might be several resource objects (Object 1 to Object N) and each object could be dealt with different access permissions. Particularly the request handling function is required to check the suitability for access permission to the corresponding resource objects in the Step 5, just after the existence of the requested resource has been checked from the Step 4. Then, access operation to the requested resource should be executed in the Step 6, and the relevant HTTP/REST response would be sent back to the requester.

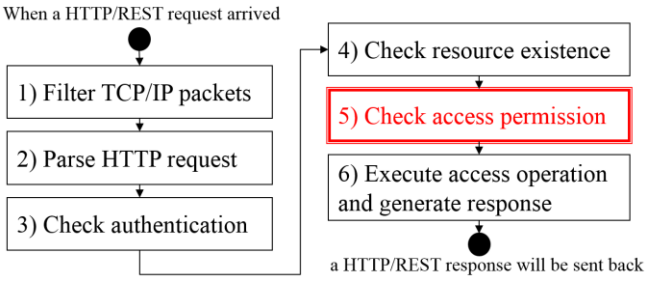


Fig. 4. Process flow from the arrival of a HTTP/REST request to sending the relevant HTTP/REST response back

TABLE I
STEPS TO ASSURE WOT ACCESS CONTROL

Step	Purpose	To filter out
1	To filter TCP/IP packets	Unallowable transaction
2	To parse HTTP/REST request	Invalid request, usually with abnormal parameters
3	To check HTTP header for basic authentication	Unverified client
4	To check whether the requested resource exists or not	Requests for the expired/outdated or irrelevant resource
5	To check the assigned access permission for the request operation	Unassigned access permission for the requested operation

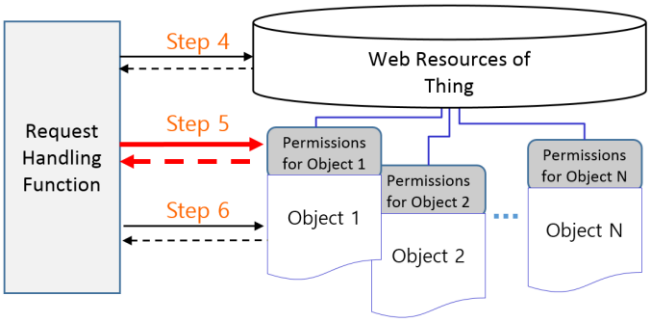


Fig. 5. WoT request handling function handling secure access control using the steps described in Fig 4 and TABLE I

Fig. 6 describes an exemplary case for assessing a HTTP/REST requests using the typical access control matrix, where a thing may have N resources (i.e. Object 1 to Object N) which can be requested from the other M things (i.e. Subject 1 to Subject M). The access permissions about Object 1 should be simply assigned for all the subjects, but this model do not provide scalability due to the overhead for maintaining the M by N matrix. That is, if a web-enabled thing has 10 resources and other 10 things are expected as subjects, the 10 by 10 matrix should be maintained and searched through. So the access permission about a resource object ‘Object 1’ can be specified for every subjects (Subject 1 to Subject M). Then, the hosting entity checks the matrix if the subject of the access

request has the enough right to access the given resource object. For example, Subject 1 have all the C/R/U/D access rights to Object 1, whereas Subject 2 has only R access rights. Here, C means Create, and likewise R for Retrieve or Read, U for Update, and D for Delete. Though additional classification of subjects into several associated groups based on Role-based Access Control (i.e. the role of 'SuperUser' may be assigned to several subjects like {Subject 1, Subject 2, …}) might alleviate this burden, but each thing still needs to maintain the lists of roles and compare the assigned access permissions against each HTTP request about CRUD operation to the specific resource object.

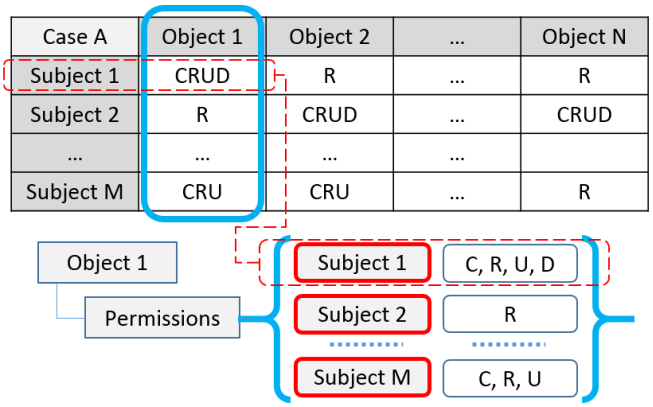


Fig. 6. Naive form of access control matrix specifying the Subject-Object relationship

As all the access operation can be categorized into the four types among C, R, U, and D, then we concentrated on the each CRUD operation with a modified access control matrix as shown in Fig. 7. When compared to the naïve form of access control matrix, now the 4 by N columns should be searched through. That is, the access permission for each specific CRUD operation can be assigned to the given subject(s) in the form of 'permission flag'. The permission flag value 1 of the specific operation means the corresponding subject is allowed to access the given resource object. For example, the Read operation to Object 1 is allowed to all the subjects (so, the permission flag R includes Subject 1 to Subject M), whereas only Subject 1 is assigned to have rights to the Update operation. We also noticed that a specific permission setting can be applied to more than one REST requests, for instance, the two operations, Create and Update, for the resource 'Object 1' in Fig. 7 can be specified identically. Likewise, more than one resource object, Object 1 and Object 2, may have the identical access permissions for Read operation.

Moreover, regarding the third WoT access control requirements, the set of permission flags can be represented as a web resource, a specific access permission rule could be applied to the multiple resources synchronously, as far as there exists a reliable web connection. Fig. 8 conceptualizes that resource objects of Thing A are referencing the same set of permission flags (i.e., 'Local Permissions A1' and 'Local Permissions A2'). And Object 1 and Object 2 have configured with Access Right A1. Since web resource can link other web resource readily using URI. Meantime, Object N of Thing A is referencing the external access permission rule, 'Global Permissions B1' which is managed by other host. Here, we assumed all the Retrieve access operation to this permission rule resource should be allowed for interoperability.

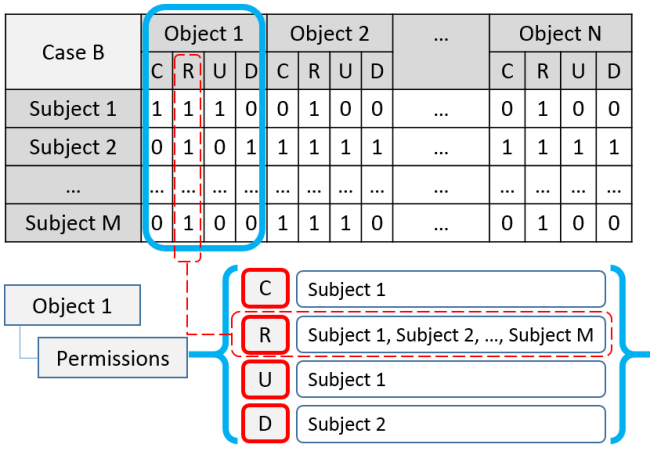


Fig. 7. Modified access matrix focusing on the each CRUD operation instead of the original subject-object relationship shown in Fig. 6

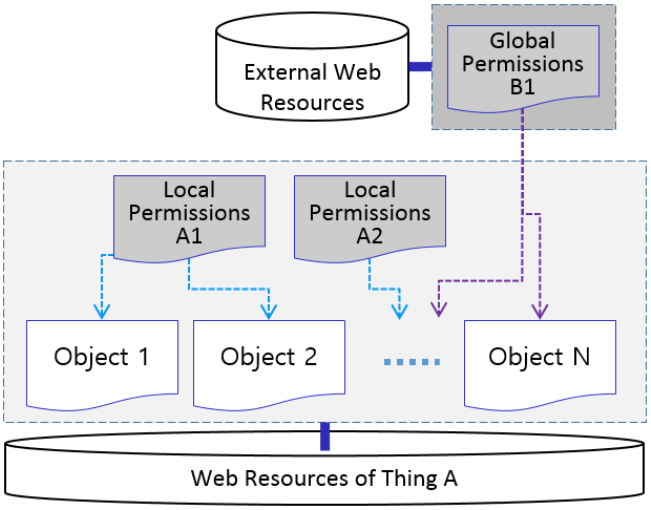


Fig. 8. Linking the set of permission flags as a type of web resource object

D. Access Permission Abstraction for WoT

Now we generalized the access permission by replacing the subjects with the generic conditions, which aggregate not only the subject information like typical identifiers for the requester (i.e. IP address, hostname, and domain information), but also context and environmental information related with the web-enabled thing itself (i.e. point of time, time schedule, location, device hardware state, and configuration limit value). Therefore, we can express a permission flag for each CRUD access operation as the following:

Permission Flag C [/R/U/D] = {Condition about subjects} + {Condition about the context of the requested resource object}

For example, as presented in Fig. 9, a web-enabled thing as the hosting entity for the resource objects may allow the 'Create' operation by judging from the combinative results throughout the several conditions (e.g. C1 to C3) as followings:

C1) if the requester is from the specific domain like '*.cnu.ac.kr',

C2) if the current time is between 9 AM and 3 PM, and

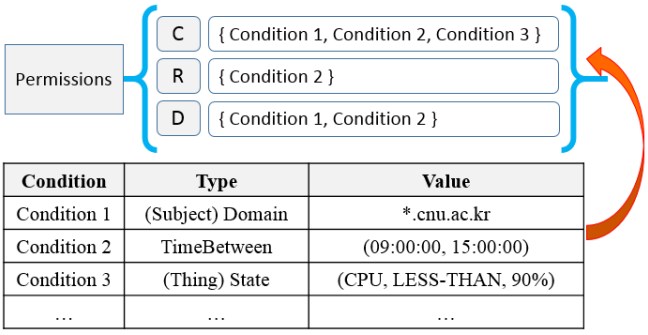
C3) if the CPU usage of the hosting entity is less than 90%.

In addition, an internal policy should be settled in order to deal with more than one condition. For instance, a policy have to be determined considering the following two situations:

--Situation 1: Permission should be granted only when all the conditions meet.

--Situation 2: Permission should be granted if any of the conditions meets.

Here, we considered the case of Situation 1 and all the conditions have the equal level of priority or weight for simplicity. So, in Fig. 9, only if the specified conditions are satisfied, access permission for each CRUD operation can be passed. For example, the permission flag C of the given resource object includes three conditions C1 to C3, whereas the flag R includes only C2.



Condition	Type	Value
Condition 1	(Subject) Domain	*.cnu.ac.kr
Condition 2	TimeBetween	(09:00:00, 15:00:00)
Condition 3	(Thing) State	(CPU, LESS-THAN, 90%)
…	…	…

Fig. 9. Exemplary representation of permission flags specifying various combination of conditions

E. Proof of Concept for WoT Access Control

We implemented a request handling function as a proof of concept with the example of Fig. 9, with two Java framework libraries, Jersey RESTful Services Framework (Available: https://jersey.java.net/) and Grizzly Web Framework (Available: https://grizzly.java.net/). The implementation accepts the HTTP request for accessing the example resource, '/root/object/', and checks the set of permissions related with the given resource, as shown in Fig. 10.

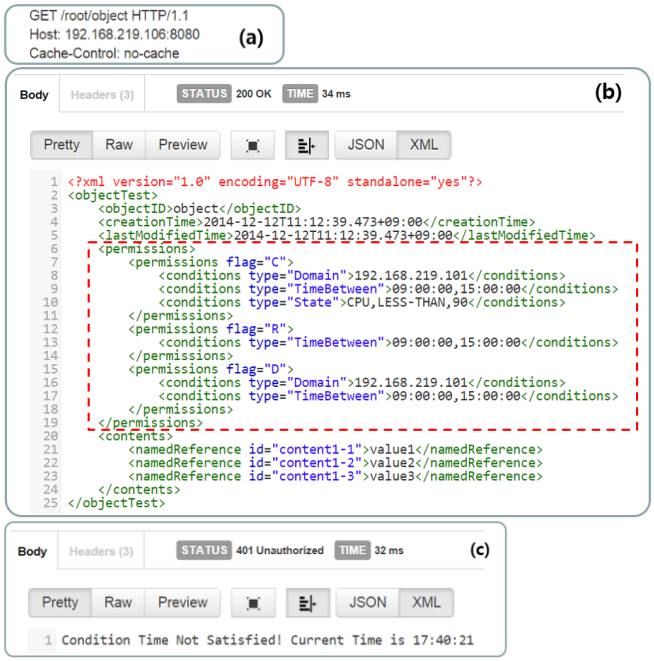


Fig. 10. **(a)** HTTP request to retrieve a web resource located at '192.168.219.106:8080/root/object/'; **(b)** HTTP response with successful retrieval of the requested resource; **(c)** HTTP response with a status code 401 ('Unauthorized')

We used Postman REST client utility to send HTTP request in order to retrieve a web resource located at '192.168.219.106:8080/root/object/', as shown in Fig. 10 (a), and receive the responses, as shown in Fig. 10 (b) or (c) accordingly. Since the retrieve access operation from (a) do not violate the condition about 'TimeBetween', the requested resource was successfully retrieved. The response payload presents the set of permission flags described in the example of Fig. 9. (c) If the request do not satisfy all the relevant conditions regarding the given access operation (in this case, the request time was beyond the time range), the requestor might get the HTTP response with a status code 401 ('Unauthorized'). The implementation showed that the web-enable thing as the hosting entity for the requested resource can control access permission as far as the relevant conditions are being checked well, apart from centralized authorization procedure.

F. Elaboration on Specifying Access Permission

For the more elaborated access control, we noted that a binary true-function can be adopted for processing the multiple conditions, then each condition can be classified into two types, i.e., inclusive and exclusive. For example, we set a rule about 'access permission would be granted only if any of inclusive conditions meets and if any of exclusive condition does not meet at the same time'. Then, the regarding permission flag might be expressed as the following:

Permission Flag C [/R/U/D] = {inclusive-type conditions}
- {exclusive-type conditions}

Fig. 11 outlined an access permission rule (<permissions>), in XML document format, including the permission flag about 'Retrieve' operation (<permission type="R">) which consists of 3 inclusive and 3 exclusive conditions. That is, the resource object referencing this permission rule can be accessed, only if any of 3 inclusive conditions (i.e. conditions about Subject id, Subject IP address, and Subject domain) can be satisfied, and at the same time if any of 3 exclusive conditions (i.e. conditions about Thing operation duration, Thing CPU usage, and Subject domain) is not applicable at all. Likewise, other three operations (i.e. C/U/D) can be listed as the other elements of this access permission rule.

```
<permissions>
  <permission type="R"> <!– // permission flag R –>
    <inclusiveConditions>                    <!– // inclusive conditions –>
      <condition type="id">Subject_1</condition>
      <condition type="ip" >168.188.100.*</condition>
      <condition type="domain" >*.cnu.ac.kr</condition>
    </inclusiveConditions>
    <exclusiveConditions>                    <!– // exclusive conditions –>
      <condition type="timeBetween" >23:55:00, 06:00:00</condition>
      <condition type="state" sensing="CPU" op="MORE-THAN" >80%</condition>
      <condition type="domain">seal.cnu.ac.kr</domain>
    <exclusiveConditions>
  </permission>
  … <!– // permission flag C, U, and/or D –>
</permissions>
```

Fig. 11. Exemplary structure for expressing the access permission

Fig. 12 describes the resource structure of a web-enabled thing using the web-based access permission mechanism, illustrated in Fig. 8. In order to configure resource objects with the specific access permissions, each resource object

within the web-enabled thing needs just a reference link to the appropriate permission rule resource, for example, '/root/permission_A1' or '/root/permission_A2'. Here, these permission rules include one or more conditions following the way specified with the example of Fig. 11. Thus, the web-enabled thing can authorize any access operation to resource object (i.e. '/root/object_1') by referencing a permission rule resource (i.e. '/root/permission_A1') and checking the access conditions.

Also, the permission rule resource '/root/permission_A1' might be referred by two resource objects, '/root/object_1' and '/root/object_2', so these objects could be accessible by checking the identical sets of access conditions. The object '/root/object_N' refers to the external permission rule '/permissions_B1' maintained by other hosting entity. Eventually, through the inter-linking among the web resources, a web-enabled thing manages access permission without any external authorization server's support. Besides, as far as the external web connection keeps alive, a number of things can be managed under the identical access permission policy when configured to adopt a global access permission rule resource.

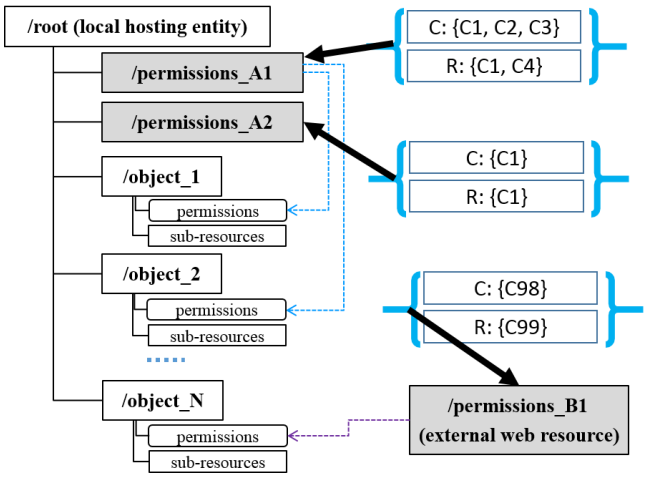


Fig. 12. Exemplary resource structure maintained by a web-enabled thing according to the case of Fig. 8.

IV. Conclusions

In this paper, we described the requirements for the access permission control to the resources of web-enabled devices in the web of things environment. Then, we presented an access permission control mechanism considering for the WoT characteristics. The proposed mechanism adopts resource-oriented architecture for Web-enabled things, utilizes the access conditions accommodating both the requester information and the context of the thing itself.

The generalized access permissions can be represented also as a web resource, easily reachable via Web architecture. This mechanism enables each Web-enabled thing to authorize access requests locally, even without dependency on a specific security domain. Moreover, it helps to apply an identical permission rule to the multiple resources just by referencing same access permission web resource. Thus, the proposed mechanism contributes to the flexible and decentralized access permission control for the Web of Things.

References

[1] J. Höller, V. Tsiatsis, C. Mulligan, S. Karnouskos, S. Avesand, and D. Boyle, *From Machine-To-Machine to the Internet of Things*. Elsevier, pp. 233–235, 2014.

[2] Vermesan, O. & Friess, P., *Internet of Things: From Research and Innovation to Market Deployment.* River, pp. 7–69, 2014.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer System*, vol. 29, no. 7, pp. 1645–1660, Feb. 2013.

[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.

[5] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[6] E. Lee, H. Lee, K. Lee, and J. Park, "Automating Configuration System and Protocol for Next-Generation Home Appliances," *ETRI Journal*, vol. 35, no. 6, pp. 1094-1104, Dec. 2013.

[7] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, "M2M: From mobile to embedded internet," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 36–43, 2011.

[8] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 1–11, 2014.

[9] Gartner. Press Release. [Online]. Available: http:// http://www.gartner.com/newsroom/id/2905717

[10] CISCO. The Internet of Things. [Online]. Available: http:// http://share.cisco.com/internet-of-things.html

[11] IDC. Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand. [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=248451

[12] Postscapes. IoT Market Analysis. [Online]. Available: http://postscapes.com/internet-of-things-market-size

[13] D. Zeng, S. Guo, and Z. Cheng, "The Web of Things: A Survey," *Journal of Communications*, vol. 6, no. 6, pp. 424-438, Sept. 2011.

[14] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the Web of Things," in *Internet of Things (IOT), 2010*, 2010, pp. 1–8.

[15] S. Duquennoy, G. Grimaud, and J.-J. Vandewalle, "The Web of Things: Interconnecting Devices with High Usability and Performance," in *International Conference on Embedded Software and Systems (ICESS)*, 2009, pp. 323–330.

[16] *Framework of the web of things*, ITU-T Y.2063, 2012.

[17] B.W. Lampson, "Computer security in the real world," *Computer*, vol. 37, no. 6, pp.37-46, June 2004.

[18] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices," in *Architecting the Internet of Things*, pp. 97– 129, Springer, 2011.

[19] S. W. Oh and H. S. Kim, "Decentralized access permission control using resource-oriented architecture for the Web of Things," in *16th International Conference on Advanced Communication Technology*, 2014, pp. 749–753.

[20] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security Challenges in the IP-based Internet of Things," *Wirel. Pers. Commun.*, vol. 61, no. 3, pp. 527–542, Sep. 2011.

[21] L. Seitz, G. Selander, and C. Gehrmann, "Authorization framework for the Internet-of-Things," in *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2013)*, 2013.

[22] B.W. Lampson, "Protection," *ACM SIGOPS Operating Systems Review*, vol. 8, no. 1, pp. 18-24, Jan. 1974.

[23] A.C. Myers and B. Liskov, "A decentralized model for information flow control," *ACM SIGOPS Operating Systems Review*, vol. 31, no. 5, pp. 129-142, Dec. 1997.

[24] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp.38-47, Feb. 1996.

[25] M. V. Tripunitara and B. Carbunar, "Efficient access enforcement in distributed role-based access control (RBAC) deployments," in *Proceedings of the 14th ACM symposium on Access control models and technologies*, 2009, pp. 155–164.

[26] M.A. Al-Kahtani and R.S. Sandhu, "A Model for Attribute-Based User-Role Assignment," in *Proceedings of the 18th Computer Security Applications Conference*, 2002, pp. 353-362.

[27] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for web services," in *Proceedings - 2005 IEEE International Conference on Web Services, ICWS 2005*, 2005, vol. 2005, pp. 561–569.
[28] M. J. Covington, P. Fogla, and M. Ahamad, "A context-aware security architecture for emerging applications," in *Proceedings 18th Computer Security Applications Conference*, 2002, pp. 249–258, 2002.
[29] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas, "Cerberus: a context-aware security scheme for smart spaces," in *Proceedings of the First IEEE International Conference on Pervasive Comput. Commun. 2003*, pp. 489–496, 2003.
[30] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: Early Progress and Back to the Future," *International Journal on Semantic Web and Information Systems*, vol. 8, no. 1, pp. 1–21, 2012.
[31] R. Fielding , J. Gettys , J. Mogul , H. Frystyk , L. Masinter , P. Leach , T. Berners-Lee, *Hypertext Transfer Protocol -- HTTP/1.1 (RFC 2616)*, RFC Editor, 1999.

Se Won OH is a senior member of engineering and research staff working for ETRI, and also under a PhD course at Chungnam National University (CNU), Korea. He received He received the BS (1999) and the MS degree (2001) from Pohang University of Science and Technology (POSTECH), Korea. Since 2001, he has been involved in several large research projects on software platform (such as RFID Event Management System, USN Middleware Platform, Software System Infrastructure) which integrates legacy applications with various data resources including RFID tags and Sensor Node. His recent research interests are Internet of Things (IoT) and Web of Things (WoT), particularly about interfacing Web-enabled devices. He has also participated standardization activities on automatic identification and data capture(AIDC) techniques as a member of JTC 1/SC 31, and as a secretary for JTC 1/SC 31/WG 6 (Mobile Item Identification and Management, 2008-2014).

Hyeon Soo Kim is a professor at Chungnam National University (CNU), Korea. He works for Department of Computer Science and Engineering at CNU. His current research areas include Software Engineering (Software Testing, Software Architecture, Software Maintenance, and Software Reengineering) and Applications (Nuclear Engineering) as well as Distributed Computing (J2EE/EJB, .NET, Web service, SOA, Internet of Things, and Large Scale Data Processing). He received the BS degree (1988) from Seoul National University (SNU), Korea, and the MS (1991) and the PhD degree (1995) from Korea Advanced Institute of Science and Technology (KAIST), Korea. He is a member of KIISE and KIPS.