Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2018) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 4th International Workshop on the Future of the Internet of Things (FIT 2018)

Fine-grained Access Control Framework for Igor, a Unified Access Solution to The Internet of Things

Pauline Sia Wen Shieng[ab], Jack Jansen[b], Steven Pemberton[b*]

[a] *University of Amsterdam, Science Park 904, 1098XH Amsterdam, The Netherlands*
[b] *Centrum Wiskunde & Informatica, Science Park 123, 1098XG Amsterdam, The Netherlands*

Abstract

With the growing popularity of the Internet of Things (IoT), devices in households and offices are becoming information sharing "smart" devices controlled via network connections. The growth of collection, handling and distribution of data generated by IoT devices presents ethical and privacy issues. Users have no control over what information is kept or revealed, the interpretation of data collected, data ownership and who can access specific information generated by their IoT devices. This paper describes an approach to data ethical/privacy issues related to IoT using a fine-grained access-control framework on Igor, a centralized home and office automation solution. We designed a capability-based access control framework on top of Igor that allows agents, either human or machine, to access and change only the data to which they are authorised. The applicability of this to the European General Data Protection Regulation (GDPR) should be obvious. The implementation, expert evaluation and performance measurement results demonstrate that this is a promising solution for securing access to data generated by IoT devices.

Keywords: Internet of Things; ethical and privacy issues; IoT; authorization; access control; framework

1. Introduction

The term *Internet of Things* (IoT) [1] was first used by Kevin Ashton in 1999. Since then, IoT has become mainstream. In 2013, Ashton insisted that IoT is here *now*; not the future but the *present* [1]. Cisco IBSG predicts there will be 50 billion devices connected to the Internet by 2020 [2]. IoT devices' data collection, handling and distribution are also growing significantly. With that, users are facing ethical and privacy issues, raising higher concerns as corporations start to monetize the data. Mason [3] categorized these issues as PAPA: a) ***Privacy***: *What information about one's self or associations must a person reveal to others, under what conditions and with what safeguards? What can people keep to themselves and not be forced to reveal to others?* b) ***Accuracy***: *Who is responsible for the authenticity, fidelity and accuracy of information?* c) ***Property***: *Who owns information? What*

* Corresponding author, steven.pemberton@cwi.nl, +31 624671668.

are the just and fair prices for its exchange? d) ***Accessibility***: *What information does a person or an organization have a right or a privilege to obtain, under what conditions and with what safeguards?*

Caron *et al.* [4] performed an analysis on how IoT impacts individual privacy and concluded that the IoT key themes related to the ethical issues highlighted by Mason [3] are: unauthorized surveillance, uncontrolled data generation and use, inadequate authentication, and information security risk. The problem statement this paper addresses is: "*The growth of collection, handling and distribution of data generated by IoT devices has presented ethical and privacy issues. Users have no control over what information to keep or reveal, interpretation of data collected, data ownership and who can access specific information generated by the IoT devices owned by them*."

Our aim is to solve data ethical/privacy issues related to IoT as defined by Mason's PAPA model. From the problem statement, four requirements of the access control mechanism are formulated:

- R1 - Access control can be delegated from the owner to others.
- R2 - Access control can be revoked by the owners.
- R3 - Access control of data should be fine-grained.
- R4 - Access control should not add heavy processing requirements, as IoT devices have low processing power.

Table 1 – Mapping of Mason's four ethical issues related to information generated by IoT with proposed solutions and project requirements

Mason's ethical issues (PAPA)	Solutions	Requirements
Privacy	Through proper access control mechanism, sensitive data is concealed / hidden from public	R1, R2, R3
Accuracy	Keeps the data only accessible to the owners, the data owner can verify on the accuracy of the information presented and is able to choose whether to share it with others or not	R1, R2, R3, R4
Property	Data is stored in an area controlled by the rightful owner, avoiding disputes on data ownership	R1, R2
Accessibility	The data owner decides who can access what data	R1, R2, R3, R4

Table 1 shows a summary of the mapping of Mason's four ethical issues related to information generated by IoT devices with our proposed solution through *Igor* and project requirements. R4 is a non-functional requirement requiring the solution to be able to be run on light weight IoT devices with acceptable performance.

1.1. This Paper's Contribution

We designed an access control framework on top of Igor, a unified access solution to the IoT [5] by focusing on authorization of different agents accessing, interacting, performing tasks and sharing information with each other. Authentication of agents, though important, was not in scope, to be addressed in future work. The solution covers lightweight IoT devices, in household and office environments. Industrial IoT which can continuously generate huge amount of data and requires high capacity storage space is out of scope. The following sections of this paper discuss: related earlier work on access control approaches for IoT, the interaction design, system design, implementation of our approach, evaluation of the design implementation, conclusion and future work.

2. Related Work

This paper extends the work done by Jansen & Pemberton [5] on Igor. Igor is a butler who knows and controls everything that goes on in the household and makes sure everything runs smoothly and perform tasks without passing judgments and maintaining complete discretion. Several Igors can work together in the environment.

Domoticz [6] is a light weight home automation system similar to Igor. However, for Domoticz, access is granted to users by devices and there is no fine-grained data access control. Ouaddah *et al.* [7] performed extensive qualitative analysis on the various security access models for the IoT devices. They highlight the strengths and weaknesses of each access control solution proposed by others that are most relevant, such as RBAC (Role-Based Access Control) by Sandhu, *et al.* [8], ABAC (Attribute-Based Access Control) by Yuan and Tong [9], UCON (Usage Control Model) by Zhang, *et al.* [10], CapBAC (Capability-Based Access Control Model) by Gong [11] and OrBAC (Organizational-Based Access Control Model) by Kalam, *et al.* [12]. Nevertheless, from their analysis, it is clear that CapBAC has more advantages than the other solutions as it is one of the oldest and is a proven access

control model [13-15]. Gong [11] introduced Identity-based CAPability protection system (ICAP) to improve capability propagation and revocation method of the traditional capability-based access control by incorporating subject identities. He used an exception list and capability propagation trees to enable full revocation of granted capabilities. With the growth of IoT, CapBAC has been adopted in many large-scale projects, for example, the European FP7 IoT@Work project [16]. Hernández-Ramos *et al.* [17] later built on the idea of PDP framework and introduced DCapBAC (Distributed Capability Based Access Control) by embedding authorization logic into IoT through Elliptic Curve Cryptography (ECC) optimization. Our work taps on the idea of using exception lists [11], capability tokens and PDP frameworks [16], modifying them to suit our needs.

The Igor software consists of a hierarchical data store (an XML repository), an XPath 1.0 [18] implementation and a server that allows REST-like [19] access to the database. Igor is primarily state-based, unlike ITTT (If This Then That) and many other IoT platforms which are primarily event-based. The advantage of being state-based is that it allows abstraction of information more easily. The advantages for Igor of using RESTful web services and XPath expressions are fast performance, better reliability and the ability to grow (more re-use of components) and perform updates without affecting the system as a whole. Thus, it is suitable for low resource IoT devices. On top of that, fine-grained access control can be achieved in a very light way. As Igor is based on XML, various access control policies for XML databases have been explored. Bertino *et al.* [20] introduced an access control model that supports positive and negative authorization as well as authorization propagation for hierarchical data stores and documents. Two kinds of privileges are supported: browsing privileges (read and navigate) and authoring privileges (append and write).

Seitz *et al.* [21] introduced an authorization framework for IoT. Their approach uses the access control standard called eXtensible Access Control Markup Language (XACML) [22]. The framework allows the definition of differentiated access rules for different requesting agents and access control at granularity of RESTful resources. Instead of using the full syntax of XACML, they proposed a subset to create a compact assertion format with only 10% of the size of the corresponding full XML assertion. The concept of propagation, the two types of privileges (browsing and authoring), and access controls defined at different granularity levels in the XML document have been adopted in our model.

3. Interaction & System Design

The objective of Igor is to provide security and offer data protection for users of IoT devices either at home or work. The target "agents" of Igor are physical users who are usually "owners" of the IoT devices, Igors (different Igors interact and share information with each other), IoT devices that are controlled by Igor as the "helper" of their owners and plug-ins (these plug-ins stand between the devices and Igor).

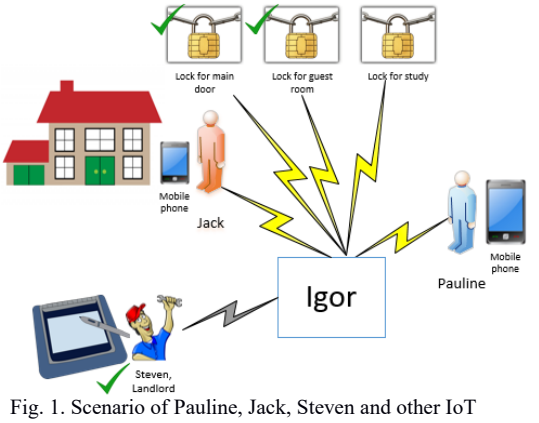

Fig. 1. Scenario of Pauline, Jack, Steven and other IoT devices accessing Igor

As illustrated in Figure 1, Pauline just bought a new smart phone. She would like to add this new mobile phone for Igor to know whether she is at home or not by detecting the Bluetooth unique ID address of the phone. As Pauline is the owner of Igor, she has administrator rights, allowing her to log in and trigger Igor to register the mobile phone's Universally Unique Identifier (UUID) address in her profile. Next, she will add the BLE (Bluetooth Low Energy) server, a little server that keeps track of which Bluetooth LE devices are in range. For each device it remembers when it was first seen, and when devices are no longer available, when last seen. The Bluetooth LE plug-in pulls data from the BLE server into its XML database and triggers Igor if the UUID address specified is in range. Once detected, Pauline can allow Igor to perform activities such as triggering the IoT smart lights (via the Lights Plug-in) of the house go on if it is dark.

The different agents accessing Igor have different access capabilities. Pauline as the owner of Igor has the full control of all the information in Igor, she is able to add new users, plug-ins and devices to Igor. Jack on the other hand is granted access only to the lock of the front entrance and the guest room as he is living in her house temporarily. The Bluetooth LE plug-in will need to be granted access to trigger Igor when Jack's mobile phone is in

range. Igor will then trigger the Lights Plug-in to switch on the lights of the house. The plug-ins only have access to the actions that are granted them. Steven, the landlord, needs to perform routine service on the heating system. As Steven is an external party, Pauline grants Steven access to the information of the number of occupants in the house, but not who in particular. Via his mobile phone, Steven can tell if there is someone at home, so that he can come over to service the heating system.

During Pauline's holiday, Jack's parents would like to visit Jack for 2 days. Pauline has to delegate her access rights to allow other visitors to enter her house to Jack, so that Jack can grant access to the front door to his parents. If necessary, Igor is able to trace back who delegated the access of those capabilities to Jack. After two weeks, Jack goes home. He no longer needs the access to Igor and the IoT devices in Pauline's house. Pauline then logs on to Igor and revokes all the accesses of the user Jack and his mobile phone.

3.1 Workflows

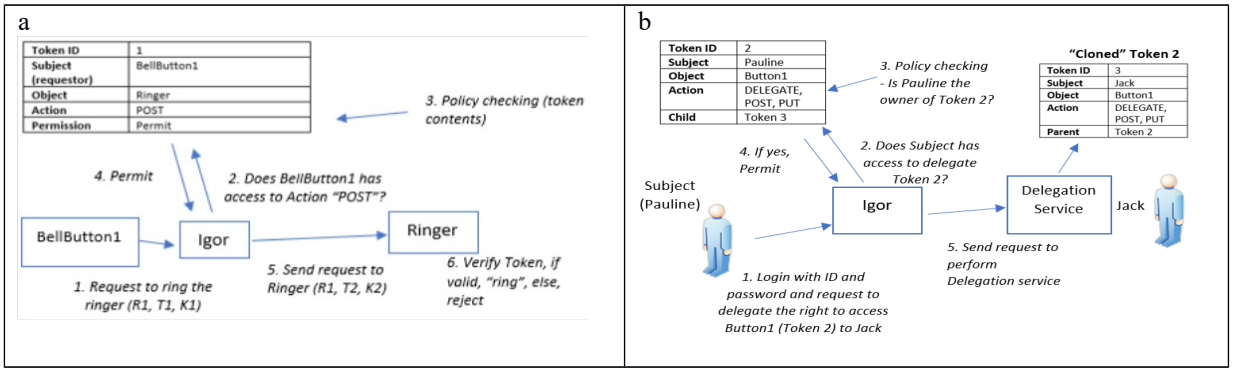

Fig. 2. (a) Igor's policy checking workflow (b) Delegation process for person

There are two types of requests going to Igor:

- *Requests from external devices* (for example, the bell button would like to trigger the house door bell to ring) – Figure 2a shows the policy checking workflow for external devices. Capabilities are represented in the form of tokens.

  1. The Subject starts the request by presenting R1 (a request), T1 (a token containing the access permissions required to perform the task) and K1 (the result of running a hash function with T1 and the shared symmetric secret key between Igor and the Subject). When Igor receives the request, Igor will run the same hash function with T1 and the shared secret key and check if the result is the same as K1. This is to ensure that the token presented has not been tampered with.
  2. If the request is valid, Igor checks if the requested Object is registered in Igor. If the Object is not available, the request will be denied.
  3. If the Object is found, the Policy Decision Point (PDP) in Igor then checks whether T1 contains the access rights required for the request. By default, access is always denied unless the token/policy assigned to the subject is found and access is mentioned. The token content design is covered in Section 4.
  4. The PDP reaches a decision (Permit / Deny) and returns it to the request service.
  5. If permitted, Igor then sends the access request to the Object by also presenting R1 (request), T2 (token containing the access permissions required to perform the task) and K2 (the result of running the hash function with T1 and the shared symmetric secret key between Igor and the Object).
  6. The Object repeats the same process in validating whether the request from Igor is valid.

- *Requests from people* (for example, Pauline would like to ring the bell by sending a request directly to Igor) – for existing users of Igor, their capabilities are contained inside their profiles. Therefore, users are only required to produce their login ID and passwords to access Igor and send their request without the need to present an external token and a generated key. Igor then checks if the user has the access rights (available

capabilities) to perform the request. If yes, Igor sends the access request to the Object (a similar process as mentioned above); if no, the request is rejected.

Capabilities that are owned by the Subject can be delegated either to a person (user) or another device. The steps for the delegation process as shown in Figure 2b:

1. Pauline logs into Igor and sends a request to delegate the right to access Button1 (Token 2) to Jack. It is assumed that Jack has a profile in Igor.
2. The PDP responds on the access request from Pauline (Subject).
3. The PDP checks whether Pauline is the owner of Token 2 and has the right to delegate the capability.
4. If so, Igor executes the Delegation Service; otherwise the request is rejected.
5. The Delegation Service "clones" Token 2 and renames it to Token 3. Token 3 id then stored in Jack's profile. Igor informs Jack of his new access (for instance by email). For traceability, Igor inserts Token 3's ID in Token 2 as a "child" and Token 2 is recorded as "parent" in Token 3.

It is assumed that the communication channel between Igor and the external device is secure. To remove the access (revocation), the same policy checking applies and Igor will send a request to the Revocation Service to remove the capability from the Subject's profile and add them into the Revocation list. There are two kinds of capabilities: internal capabilities (that are only stored in Igor's XML database) and external capabilities which Igor provided to external IoT devices. Only external capabilities will be recorded into the Revocation list upon revocation. To revoke internal capabilities, the requester must first have the access right to do so and if yes, the capabilities are immediately deleted.

3.2 Assertion (Token) Design

As mentioned before, the CapBAC model [17] and the authorization framework introduced by [21] are used as a basis for the design of the access control policy and authorization process for Igor. When an agent is granted access by Igor, the Token Generator encodes the authorization decision as a token (assertion). The contents of the tokens are used by Igor for the enforcement of the access control decisions. As suggested by [21], a subset of the full XACML and SAML [23] standard is adopted to simplify the processing on Igor. The format of the token in XML is as shown below:

```
<au:capability>
<comment>This allows all access to own data</comment>
<id>id-pauline-readself</id>
<ii>2018-02-15T10:02:52Z</ii>
<is>IGOR1</is>
<sk>BvDgLAXSHe...0RLhfwS1fue </sk>
<obj>/data/identities/pauline</obj>
<get>descendant-or-self</get>
<put>descendant</put>
<post>descendant</post>
<delete>descendant</delete>
<ob NB="2018-02-15 12:00:00" NA="2019-02-15 12:00:00"/>
</au:capability>
```

The "***comment"*** is just a free text description on what is the purpose of the token. The "***id"*** is the token identifier, "***ii"*** is the Issue Instant in UTC [24] format, "***is"*** is the identifier of the Assertion Issuer (only available in external tokens) and "***sk"*** (Subject Key) is the owner of the token, using a public key for confirmation; "***obj***" (object) is the target resource URI authorized by the assertion; "***get***", "***put***", "***post***", "***delete***" are the actions, with propagation type for each action; "***ob***" is an abbreviated XACML Obligation, a local condition that is verified on the device. In this case we have not-before (***NB***) and not-after (***NA***) times. For external token, an additional value is defined: the audience of the capability. For the prototype implementation, a token is used by the Token Generator in Igor to generate the outgoing "Authentication Bearer" header of the external access tokens.

The abstraction in a token is: "*Subject may do Action to Resource (object) under Conditions".* The other values (*id, ii, is*) are administrative, for traceability purposes. In the example, Pauline (subject) may read, update and delete her own profile (object) if the current date is between 2018-02-15 and 2019-02-15 (obligation).

3.3 Authorization Propagation

As Igor is based on an XML database, access control policies are enforced using authorization propagation for hierarchical data stores: a "top-down" strategy for granting authorization is applied. Table 2 shows the different types of authorization propagation used in Igor. As an advantage of the CapBAC model, the *Principle of Least Authority* (PoLA) (Least Privilege) is the default [16]. Therefore, no agents are able to access any of the data or

actions defined in Igor unless granted the permission to do so. There are four access control permission types POST (create), GET (Read), PUT (Update) and DELETE (Delete). Fine-grained access control can be achieved with the combination of the permission types defined with the propagation access control type and these are later mapped with the XPath of the XML database.

Table 2 – Types of Authorization Propagations

Propagation Type	Symbol	Description
Self	°	Access to the one specified level only
Child	+	Access to one level below
Descendants	*	Access to all levels below
Descendants or Self	^	Access to the specified level and all the levels below it
No Propagation	-	No access at any level

4. Implementation

The implementation part of the framework was carried out on a Raspberry Pi 3, a simple, small, yet sufficiently powerful device to perform the functionalities. Igor is programmed in Python 2. The secure https [27] protocol is used to create secure channels for Igor to connect to the Internet and other IoT devices. We opted for self-signed certificates as sufficient to meet our needs for this implementation. We chose not to perform encryption on internal tokens for better performance and because it is assumed that Igor can trust its own internal capabilities. For external tokens, there are extra fields implemented as compared to the internal tokens, such as the Issuer ID, audience of the capability, subject of the capability and the period of validity of the token. This is for better traceability and to enable security checks on those external tokens when presented back to Igor. Simple user interfaces are implemented to demonstrate the workings of the access control design. All code is open source on Github [25,26].

5. Evaluation

The goal was to test our prototype and obtain feedback from IoT access control experts familiar with the ethical and privacy issues. The second goal was to perform comparison of the response time of the system before and after the implementation of the proposed security framework. Evaluation forms were prepared. Separate evaluation sessions were conducted with five experts independently. The design and functionalities of the security framework of Igor explained and later tests were done on the prototype and feedback captured. This is a qualitative evaluation and is subjective to the experts' personal opinions. The Likert Scale was used (0 the lowest and 5 the highest) to measure whether the project meets the overall objectives in addressing ethical and privacy issues users are facing regarding information generated by IoT devices.

For measuring the response time of the system, the Apache Benchmarking (ab) [28] tool was used. Measurements were taken to compare the difference of response time before and after the implementation of the proposed security framework for loading the first landing page of Igor (with and without https), executing a read (GET) request function and executing an update (POST) request function.

5.1 Expert Evaluation Results

As the objective of the prototype was to demonstrate the authorization framework of the proposed solution, user-friendly interfaces and automation of some of the steps are still lacking. Designing friendly user interface is out of scope for this project. In the current design, revocation of capabilities can be done after they have been delegated. All of the evaluators agree that the approach of using a revocation list to revoke external capabilities is a good approach. It is suggested to have distributed revocation lists to lower the risk of depending on one centralized list. As a summary, all of the evaluators agree that the project requirements mentioned in Section 1 are met with the ranking of 4 or higher (Likert Scale). They feel that the proposed solution is very useful in solving the problem and use case presented in the beginning of the paper. All of them agree that the prototype presented is sufficient to demonstrate how the proposed solution works. More details on the evaluation results can be found in [29].

5.2 Performance Measurement Results

The performance measurement readings are based on the time taken for 10 concurrent sessions of 100 requests in milliseconds. There is minimal difference between the time of loading the first page of Igor and for executing

GET requests. However, a greater difference in response time is observed when POST requests are executed with access control checking, which is about 3.5 seconds (470% increase) as shown in Table 3. The readings of the system average load performance were also taken but they show no significant difference despite the longer response time when security checks are performed.

Table 3 – Comparison of system response time before and after implementation of the security framework for Igor

Type	Before (without security) in ms	After (with HTTPS and Capability) in ms	Difference (ms)	Ratio (%)
Front Page Landing	776	864	88	11%
GET Function	313	471	159	51%
POST Function	373	4200	3464	470%

In summary, performance degradation of the system after the implementation of the security framework for Igor in a lightweight device is acceptable. This fulfills the R4 requirement mentioned in Section 1.

6. Discussion

The current design uses an XML database with XPath expression and RESTful web services. The XPath expression has produced fast performance, high reliability and flexibility to grow for lightweight IoT devices. The proposed authorization propagation types are designed for an XML database. Nevertheless, the design is not tied to XML and should be generic. Similar implementation can also be achieved using another simple file format like JSON. Two very popular approaches to access Web services are: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). REST is chosen in our case as it is simpler to use, flexible and provides a lighter-weight alternative. Furthermore, REST is not limited to XML but it can also output to CSV, JSON and RSS. Therefore, for future expansion opportunity for IoT solution, REST is a better option.

Security checks and access granting should be seamless allowing users to continue their tasks on the IoT devices without interruption. Automation is also key to the success of the solution as users should not notice the verification and validations performed by Igor or on the IoT devices. The current implementation of https and shared symmetric key generations are off-the-shelf security solutions. Should security technology change with new and more secure authentication and encryption solutions, it should be easy to adopt them in future versions of Igor.

In the access control mechanism, Igor performs centralized checks on all access requests to all devices, services and actions. There are concerns that this is a single point of failure or this approach carries a high risk in the event the single Igor is attacked. However, the current design has capabilities assigned to every component in Igor that if one component is attacked, the others will still be protected. The proposal of having multiple Igors working together to control access to different IoT devices tries to get the best of both approaches - decentralized centralization.

7. Conclusion & Future Work

With the rapid growth of the data generated by IoT devices, users are facing ethical issues in how the data are handled. Mason's PAPA model [3] (privacy, accuracy, property and accessibility) presents the main ethical issues hitting the data management effort in the IoT industry. The main aim of this paper is solving data ethical/privacy issues related to IoT as defined by Mason's model. This paper presents a fine-grained access control solution framework on Igor, a centralized home and office automation solution. Data collected from IoT devices are stored in a centralized location controlled by the data owner and all accesses are controlled and granted through Igor. CapBAC has been identified as the access control model suitable to meet the objectives for this project. The implementation, expert evaluation results and performance measurements taken show this is a promising solution for securing access to data generated by IoT devices.

Based on the evaluations conducted, there is still room for improvement and future work. More work should be done to analyze the strengths and weaknesses of different authentication approaches for IoT devices. Adoption of the proposed access control mechanism will increase once user-friendly interfaces are built for novice users. For future implementations, each person or device should be given a set of default capabilities with pre-defined templates which can be assigned to different user groups. Access capabilities can be grouped by the owner's preference such as family group, external parties, cleaners, etc. External capabilities have a limited lifespan as defined upon creation. The renewal of these capabilities should happen autonomously. This approach keeps these external capabilities fresh

and in check. More work needs to be done to review all the different approaches for high volume capabilities creation or the possibility of sharing revocation lists when there are more than one Igor in the home or office.

8. Acknowledgements

Thanks to all evaluators participating in the expert evaluation, to all colleagues in CWI-DIS for sharing their expertise and support, and finally to Frank Nack from the Information Studies, University of Amsterdam, for valuable guidance in completing this paper.

9. References

[1] Kram T, *et al. Introduction to the Internet of Things.* In: Bassi A, *et al.* Eds. Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model, Springer, 2013: 1-10.
[2] D. Evans. The internet of things how the next evolution of the internet is changing everything. *Cisco Internet Business Solutions Group (IBSG)* 2011; 1: 1-11.
[3] Mason RO. Four ethical issues of the information age. *MIS Quarterly* 1986; 10: 5-12.
[4] Caron X, Bosua R, Maynard SB, and Ahmad A. The internet of things (IoT) and its impact on individual privacy: An australian perspective. *Computer Law & Security Review: The Int. Journal of Technology Law and Practice* 2016; 32: 4-15.
[5] J. Jansen, S. Pembenton. An architecture for unified access to the internet of things. *XML London 2017 – Conference Proceedings* 2017; 1: 38-42.
[6] Gizmocuz. Domoticz. 2018. Retrieved 10 February, 2018, from https://www.domoticz.com/wiki/Main_Page .
[7] Ouaddah, A., Mousannif, H., Elkalam, and Ouahman. Access control in the internet of things: Big challenges and new opportunities. *Computer Networks* 2017; 112: 237-262.
[8] C. E. Youman, R. S. Sandhu, H. L. Feinstein, and E. J. Coyne. Role-based access control models. *Computer* 1996; 29: 38.
[9] Yuan E, Tong J. Attributed based access control (ABAC) for web services. *Web Services, 2005.ICWS 2005.Proceedings.2005 IEEE International Conference on* 2005; 569.
[10] Zhang X, Park J, Parisi-Presicce F, and Sandhu R. A logical specification for usage control. *Proceedings of the ninth ACM symposium on access control models and technologies* Jun 2, 2004; 1-10.
[11] Gong L. A secure identity-based capability system. 1989; 56-63.
[12] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, and G. Trouessin. Organization based access control. *Proceedings POLICY 2003 IEEE 4th International Workshop on Policies for Distributed Systems and Networks* 2003; 120-131.
[13] Dennis J, Van Horn E. Programming semantics for multiprogrammed computations. *CACM* 1966; 9: 143-155.
[14] Henry M. Levy. *Capability-Based Computer Systems.* GB: Digital Press. 1984: .
[15] Tanenbaum AS, Mullender SJ, and Renesse v,R. Using sparse capabilities in a distributed operating system. 1986; .
[16] Gusmeroli S, Piccione S, and Rotondi D. A capability-based security approach to manage access control in the internet of things. *Math Comput Model* 2013; 58: 1189-1205.
[17] Hernández-Ramos JL, Jara AJ, Marín L, and Skarmeta Gómez AF. DCapBAC: Embedding authorization logic into smart things through ECC optimizations. *International Journal of Computer Mathematics* 2016; 93: 345-366.
[18] W3Schools. XPath tutorial. 1999. Retrieved 27 April, 2018, from https://www.w3schools.com/xml/xpath_intro.asp .
[19] Oracle. What are RESTful web services? 2013. Retrieved 27 April, 2018, from https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html .
[20] Bertino E, Castano S, Ferrari E, and Mesiti M. Specifying and enforcing access control policies for XML document sources. *World Wide Web* 2000; 3: 139-151.
[21] Seitz L, Selander G, and Gehrmann C. Authorization framework for the internet-of-things. *2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)* 2013; 1-6.
[22] OASIS. eXtensible access control markup language (XACML) version 3.0. 2013. Retrieved 2 January, 2018, from http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html .
[23] Oasis. Security assertion markup language (SAML) V2.0 technical overview. 2008. Retrieved 27 April, 2018, from http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html .
[24] W3C. Date and time formats. Retrieved 27 April, 2018, from https://www.w3.org/TR/NOTE-datetime .
[25] Igor, an extensible home automation server. 2018. Retrieved 10 Feb, 2018, from https://github.com/cwi-dis/igor .
[26] Jansen, J., Sia, P. Igor database. 2018. Retrieved 8 April, 2018, from https://github.com/cwi-dis/igorDatabase.pauline .
[27] Google. Secure your site with HTTPS. 2018. Retrieved 27 April, 2018, from https://support.google.com/webmasters/answer/6073543?hl=en .
[28] The Apache Software Foundation. Ab - apache HTTP server benchmarking tool. 2018. Retrieved 20 April, 2018, from https://httpd.apache.org/docs/2.4/programs/ab.html .
[29] Sia P. Fine-grained access control framework for igor, a unified access solution to the internet of things. 2018; 1-15.