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

SUBMITTED IN PARTIAL FULLFILLMENT FOR THE DEGREE OF MASTER OF SCIENCE

Pauline Wen Shieng Sia
11305169

MASTER INFORMATION STUDIES
HUMAN-CENTERED MULTIMEDIA

FACULTY OF SCIENCE
UNIVERSITY OF AMSTERDAM

24 May 2018

1st Supervisor
Dr. P.S. César Garcia
Distributed and Interactive Systems, CWI

2nd Supervisor
Dr Frank Nack
Faculty of Science, University of Amsterdam

ABSTRACT

With the growing popularity of "Internet of Things" (IoT), devices in our households and offices are becoming configurable to be information sharing "smart" devices and be 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 to keep or reveal, interpretation of data collected, data ownership and who can access specific information generated by the IoT devices owned by them. More and more IoT users are seeking ways to control access to data generated. The main aim of this paper is to describe how to solve data ethical/privacy issues related to IoT using a fine-grained access-control framework on Igor, a centralized home and office automation solution to solve the data sharing and access issues. Data collected from IoT devices are stored in a centralized location controlled by the data owner and all access is controlled and granted through Igor. A Capability-Based Access Control Model (CapBAC) solution has been adapted for this project. 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 expressed by Kevin Ashton in 1999. Since then, IoT has become more and more mainstream. In 2013, Ashton insisted on the realization that IoT is here *now*; it is not the future but the *present* [1]. Cisco IBSG predicts there will be 50 billion devices connected to the Internet by 2020 [2]. With the growing popularity of IoT, more and more of the devices in our households and offices are configurable to be information sharing "smart" devices, controlled via network connections.

IoT devices' data collection, handling and distribution are also growing drastically. With that, users in general are facing ethical and privacy issues. This raises higher concerns as corporations start to monetize the data collected. Mason [3] categorized these ethical issues as PAPA: a) ***Privacy***: *What information about one's self or one's associations must a person reveal to others, under what conditions and with what safeguards? What things 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 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 on 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. Privacy and security are also the main topics of concern during a public consultation involving general users, associations, academic groups, civil societies and industry players conducted by the European Commission in 2012 [5].

Besides privacy, there is also a need to protect data access at different levels of detail. Figure 1 shows an example where Pauline is going away on holiday for 2 weeks. Her 17-year-old nephew, Jack is coming over to stay at Pauline's house while she is away. Pauline would like to give Jack access to the main door of the house and the guest room, but not her study where she has confidential documents. The house is equipped with smart door locks, however, of different makes and models. Each lock uses a different remote control, and user console, and has a different data storage location. There is no central control where Pauline can specify access for Jack to only certain locks and data collected from these smarts locks to be stored in one location only accessible to Pauline.

While Pauline is away, she would like to know when Jack is at home without knowing the details of his activities or his exact location in the house (for Jack's privacy). Pauline also wants to know if there are more visitors in the house besides Jack. For Jack's safety, she wants to be triggered if there are unknown people in the house at night (between 23:00-7:00). Pauline would like access to all this information without sharing it with other parties like corporate companies.

Other external parties, for example, Steven, the landlord of the house where Pauline is the main tenant, only need to know if there is someone in the house, so that he can come over to service the heating system. He does not need to know who is at home because he does not need that information to complete his task. However, there is no solution currently to trigger Steven when the number of occupants is more than 0, between working hours 9:00 to 17:00. After 2 weeks, Jack returns to his parents. Pauline needs to remove all his accesses and information generated during his stay as they are not needed anymore, and to protect Jack's privacy.

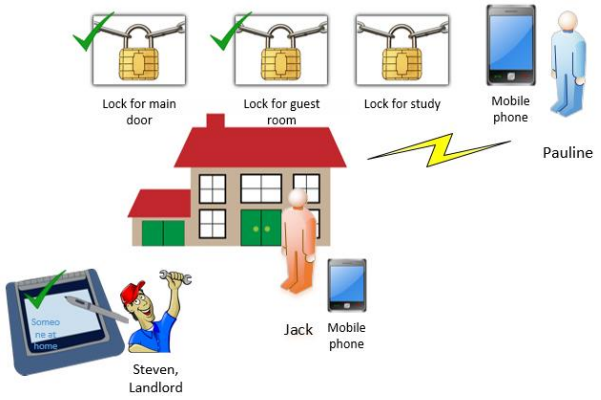


Figure 1 – Example of challenges in providing fine-grained access control

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 and as explained in the

example above. From the use case, six requirements of the access control mechanism are formulated:

- R1 – New users or devices can be easily added
- R2 – Access control can be delegated from the owner to others
- R3 – Access control can be revoked by the owners
- R4 - Access control of data should be fine-grained
- R5 – Access control should be easily managed and modified for/by normal users
- R6 - Access control solution should not add heavy processing requirement as IoT devices have low processing power

Mason's ethical issues (PAPA)	How to solve that?	Requirements
Privacy	Through proper access control mechanism, sensitive data is concealed / hidden from public	R1, R2, R3, R4
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	R2, R3, R4, R5
Property	Data is stored in a storage area controlled by the rightful owner. Thus, no dispute on data ownership	R1, R2, R3
Accessibility	The data owner decides who can access what data	R1, R2, R3, R4, R5

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

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. R6 is a non-functional requirement requiring the solution to be able to be run on light weight IoT devices with acceptable performance.

The solution is to keep data generated by IoT devices in a storage location controlled by the data owners, where they are empowered to decide who can see or access the data, even to a fine-grained level.

1.1 This Paper's Contribution

We designed an access control framework on top of Igor, a unified access solution to the IoT [6] 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 and should be addressed in future work. The solution covers lightweight IoT devices such as mobile phones, in household and office appliances. 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 work that has been carried out previously 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 [6] on Igor. Igor, is named after a Discworld [7] character, a butler who knows and controls everything that goes on in the household, and makes sure everything runs smoothly and perform tasks without passing judgements and maintaining complete discretion. There are also other Igors working together in the environment.

Domoticz [8] is a light weight home automation system similar to Igor. It allows users to monitor and configure IoT devices and notifications or alerts can be sent to any mobile device. Domoticz uses its own embedded web server written in C++. On top of that, users can write simple Blockly [9] or LUA [10] scripts right in the graphical user interface (GUI) provided. Domoticz uses an SQL Lite [11] database and access to the database is controlled at database level. For Domoticz, access is granted to users by device and there is no fine-grained data access control defined.

Fahrny and Park [12] designed authentication and binding multiple devices where the first device (of a higher security profile) may vouch for the authenticity of a second device with a lower security profile. This allows the second device to access the content from a content provider. The vouching process involves the overlaying of a digital signature of the first device on a registration request that has been signed and transmitted by the second device.

Ouaddah et. al. [13] 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. [14], ABAC (Attribute-Based Access Control) by Yuan and Tong [15], UCON (Usage Control Model) by Zhang, et. al. [16], CapBAC (Capability-Based Access Control Model) by Gong [17] and OrBAC (Organizational-Based Access Control Model) by Kalam, et. al. [18]. Ouaddah et. al. [13] found that quantitative evaluation on most of the proposed access control models from the available literature is not possible yet because they have not been developed in practice. 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 [19-21] .

In 2017, Hossain, Hasan and Skjellum [22] performed a meta study of the challenges, approaches and open problems with the security of IoT. They concluded that it is hard to find a solution that accommodates a heterogeneous mix of diverse IoT devices. They are positive about CapBAC access control mechanism, as it is suitable for both the human-to-things and things-to-things communications.

The concept of capability-based access control started with the paper presented by Dennis and Van Horn [19] in 1966 to manage computer hardware computation activities in multi-programmed computer systems. They introduced capability as token or key that defined permissions to processors to access computational objects. From then onwards, many more books and papers have been written about capability-based access control to further refine the concept and to address its weaknesses. In 1988, Levy [20] further explained the concept of capability-based computer systems and its benefits, which are: having high flexibility, a uniform mechanism for naming objects, a great

protection mechanism and normal users can add capabilities without having special privileges.

Gong [17] 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. This enforces better security policies during capability propagation as the subjects are now traceable. He used an exception list and capability propagation trees to enable full revocation of granted capabilities. It was proven by Gong [17] that ICAP requires less storage, incur lower costs and performs better than prior solutions.

With the growth of IoT, CapBAC has been adopted in many large-scale projects, for example, the European FP7 IoT@Work project [23]. Hernández-Ramos *et al.* [24] 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 or its standard protocols such as a modified Diffie-Hellman using ECC (ECCDH) [25] or Ellyptic Curve Digital Signature Algorithm (ECDSA) [26]. The ECC is specifically designed for constrained environments (light weight), can be used in a distributed approach, allows fine-grained and context aware authorization decisions [24]. Our work taps on the idea of using exception list [17], capability token and PDP framework [23] and modify them to suit our needs.

The software Igor is a hierarchical data store (an eXtensible Markup Language, XML repository), an XPath 1.0 [27] implementation and a server that allows REST-like [28] access to the database. There are different people, IoT devices, plug-ins and different Igors interacting with each other. In this paper, these different parties are called agents. Igor is primarily state-based, unlike ITTT (If This Then That) and many other IoT platforms which are primarily event-based. The advantage of state-based is that it allows abstraction of information more easily. The advantages for Igor to use RESTful web services and Xpath expression are fast performance, better reliability and the ability to grow (more re-using components) and performing 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.

In our project, Igor acts as the main "trusted" device that controls all the other connected IoT devices via various plug-ins. Once Igor has established the first authenticated connection with an IoT device, that connection is assumed to be trusted for all future access requests under specified conditions. All the other IoT devices or users who want access to that registered device would need to first request access via Igor. Igor offers a solution to accommodate the communication of heterogenous mix of diverse devices. Policy decision points (PDPs) are used to obtain authorization decisions. Whenever the subject tries to access an object or service, a capability token is presented to the access request. PDP decides whether to grant access or not based on the received capability and the internal rules defined.

As Igor is based on XML, various access control policies for XML databases have been explored. Bertino *et al.* [29] introduced access control policies that are enforced at DTD (Document Type Definitions) level as well as the specific XML document level. Their access control model supports positive and negative authorization as well as authorization propagation for hierarchical data store and documents. The access controls are defined at different granularity levels. Two kinds of privileges are supported: browsing privileges (read and navigate) and authoring privileges (append and write). The propagation policies are also defined by Bertino *et al.* [29]:

- Cascade: authorization to all direct and indirect levels
- First level – authorization to all the direct sub-levels
- None – no authorization propagation is performed

They also introduced different direction of propagation within the XML document or DTD, depending on different classification levels: homogenous, heterogeneous and mixed. Homogenous level would apply a "top-down" strategy for granting authorization, whereas the heterogenous level would apply a "bottom-up" strategy for granting authorization. The third option is to implement both "top-down" and "bottom-up" strategies (mix level). They mentioned that the authorization defined at lower levels always supersedes the higher level. This proposed model has been implemented on a system called Author-X [30,31]. Author-X provides an interface for system administrators to define access control to XML documents. It supports multi-granularity protection objects and positive/negative authorizations at document and DTD levels. Chebotko *et al.* [32] later enhanced this access control model by introducing graphs matching to analyze if an input query is fully acceptable, fully rejected or partially acceptable. They also included an index structure for XML element types to speed up the processing of access granting. Nevertheless, this solution was introduced for large DTDs and is very complex to implement.

Seitz, Selander and Gehrmaan [33] introduced an authorization framework for the IoT. The aim of the framework is to allow fine-grained and flexible access control for the connection of devices with limited processing power and memory. Their approach uses the access control standard called eXtensible Access Control Markup Language (XACML) [34]. The framework that allows definition of differentiated access rules for different requesting agents and access control at granularity of RESTful resources. Instead of using a full syntax of XACML, they proposed a subset of XACML to create a compact JSON-based [35] assertion standard. This compact assertion is only 10 percent of the size of the corresponding full XML assertion.

For the case of Igor, access control at the XML database level is sufficient. 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 proposed access control model. In our solution, the framework of Seitz, Selander and Gehrmaan [33] is adopted for the access control model proposed for Igor.

3. INTERACTION DESIGN

This section covers who are the target agents, and examples of scenarios applicable for these agents as well as the main authorization workflows defined for Igor.

3.1 Target Agents

The objective of Igor is to provide security and offer data protection for users of IoT devices either at home or office locations. The target "agents" of Igor are:

- People/Users – physical users who are usually "owners" of the IoT devices, as well as other users who interact or

would like to view information from the IoT devices like the house temperature readings, whether the lights are on, etc.

- Igor – there can be multiple Igor devices, each with different tasks to perform, like monitoring hardware performance in the server room, controlling who can access the office locations, controlling lighting based on the presence of inhabitants in the rooms, etc. Sometimes, these different Igors interact and share information with each other[1].
- Devices – The IoT devices that are controlled by Igor as the "helper" of their owners. Once connections have been established between these devices and Igor, the access to the data and information generated by these devices are stored securely in Igor and only authorized agents are allowed to view or modify them.
- Plug-ins – Igor uses different plug-ins to access heterogenous IoT devices. These plug-ins stand between the devices and Igor. Access has to be granted to these plug-ins to ensure that they can only perform functions that are allowed such as trigger the lights of the house to be on, alert the owners of the house if there are any intruders detected through the IoT sensors, etc.

3.2 Scenarios

This section describes the scenarios to illustrate what and how access rights are granted to different agents and the relevant process flow.

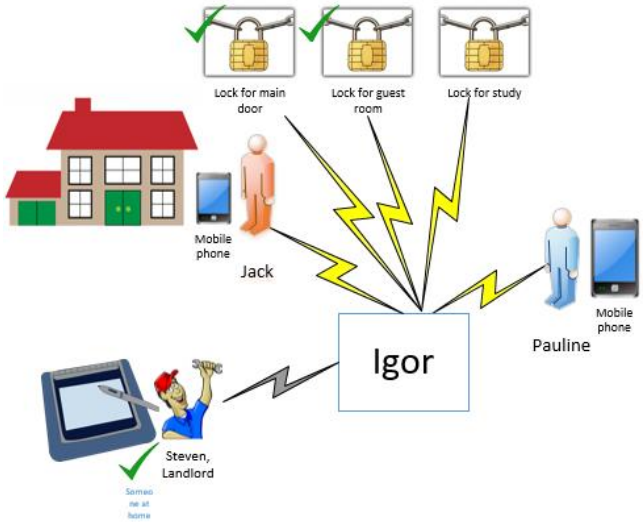


Figure 2 – Scenario of Pauline, Jack, Steven and other IoT devices accessing Igor

3.2.1 Adding new users, devices and plug-ins into the Igor system

As illustrated in Figure 2, 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 smart mobile phone. As Pauline is the owner of Igor, Pauline has administrator rights to Igor. This allows Pauline to log in to Igor and trigger Igor to register the mobile phone's Universally Unique Identifier (UUID) [36] address in her profile, so that Igor knows that the UUID address belongs to Pauline's mobile phone.

Next, she will need to add the BLE (Bluetooth Low Energy) server. This little server keeps track of which Bluetooth LE devices are in range. For each device it remembers when it was first seen (and for devices that are no longer available when it was last seen). Steps to setup BLE server is available at [37]. The Bluetooth LE plug-in is used by Igor to pull data from the BLE server into its XML database and triggers Igor if the UUID address specified for Pauline's mobile phone is in range. Once detected, Pauline can allow Igor to perform some activities like trigger the IoT smart lights (via the Lights Plug-in) of the house to be on if it is already dark.

From the scenario mentioned earlier, when Pauline's 17-year-old nephew, Jack comes over to stay for two weeks, she then registers Jack and his mobile phone to Igor so that Jack can access most of the IoT devices in the house, except the lock of Pauline's study. The IoT sensors in Pauline's house are also given the rights to know when Jack is at home from the UUID address of his mobile phone.

3.2.2 Specifying different access levels

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 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 Lights Plug-in to switch on the lights of the house. The plug-ins only have access to the actions that they are allowed to do.

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.

3.2.3 Editing (Update) the current settings of the access rights of the users, devices and plug-ins

If Jack decides to stay longer with Pauline, Pauline then logs in to Igor again to extend the capability of Jack to access all the IoT devices in the house except for the study. Besides switching on the light, Pauline would like Igor to show if there are any unknown visitors at home via sensors between 23:00 to 8:00 for Jack's safety. As Pauline is the administrator and the owner of the IoT devices, she is able to perform the task of changing the settings of Igor and other IoT devices. Jack is not able to perform that function as he is only a visitor to the house and he does not have the edit rights to the IoT devices settings.

3.2.4 Delegation

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

[1] Although interaction between different Igors is possible, it is not implemented in this project.

access to the front door to his parents. Pauline then logs on to Igor and since Jack already has a profile in Igor, she just selects the right to delegate access rights to open the main door and assign them to Jack. Igor checks if Pauline has the right to delegate the capabilities to Jack and if yes, replicate that capability to Jack. If necessary, Igor is able to trace back who delegated the access of those capabilities to Jack.

3.2.5 Revocation

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. Igor verifies if Pauline has the access rights to perform capability revocation and if yes, the accesses of Jack and his mobile phone are removed. Unfortunately, when Pauline was on holiday, she lost her mobile phone. Pauline logs on to Igor and revokes all accesses related to that mobile phone.

3.3 Workflows

This section presents the workflows of the proposed access control framework which is based on the CapBAC approach as discussed in Section 2.

3.3.1 Basic Authorization Model

The basic access control model used by *Igor* is shown in Figure 3.

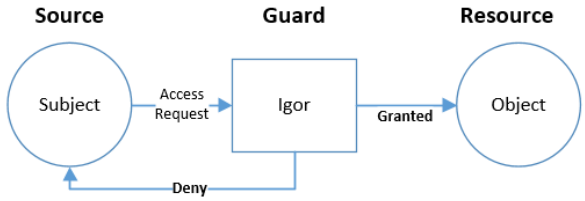


Figure 3 – Basic access control model

In a basic access control model, a subject (any agent) requests access to a resource (such as thermometer readings or to switch on the lights) to Igor. Igor then checks if the request is valid and if the Subject has the correct capabilities for the request. If yes, Igor will grant access or performs the tasks as requested by the Subject towards the Object, else, the request will be denied.

3.3.2 Performing Actions / Requests

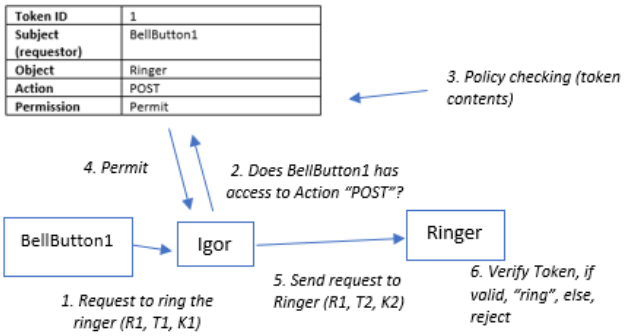


Figure 4 – Igor's policy checking workflow.

There are two types of requests going to Igor:

- *Request from external devices* (for example, the bell button would like to trigger the house door bell to ring) – Figure 4 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 (request), T1 (token containing the access permissions required to perform the task) and K1 (the result of running 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 checks 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 T2 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.

- *Request 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 (similar process as mentioned above), if no, the request is rejected.

3.3.3 Subject Updating Capabilities / Data in Igor

Figure 5 shows the process of updating or changing any capabilities or data in Igor. The Subject (user) first logins to Igor by specifying a login ID and password. Igor checks if the ID and password are valid. If yes, the user is successfully logged in, otherwise, the login request is rejected. The user then tries to edit the capabilities or data in Igor. The Policy Decision Point (PDP) in Igor then checks if the user has the right capabilities to carry out the action. If yes, the action is permitted otherwise, the action request is rejected.

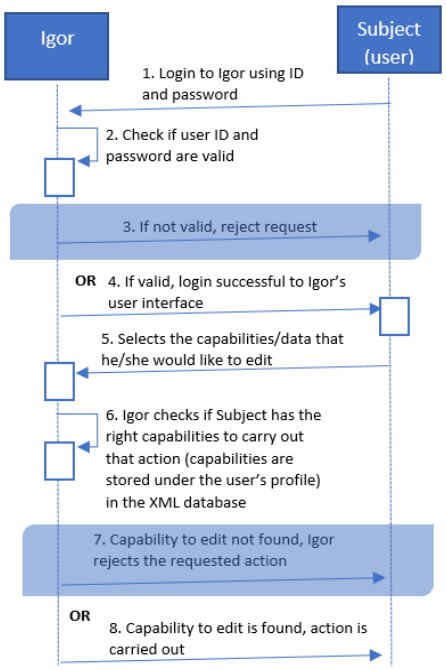


Figure 5 – Update Capabilities / Data in Igor

3.3.4 Delegation

Capabilities that are owned by the Subject can be delegated either to a person (user) or another device:

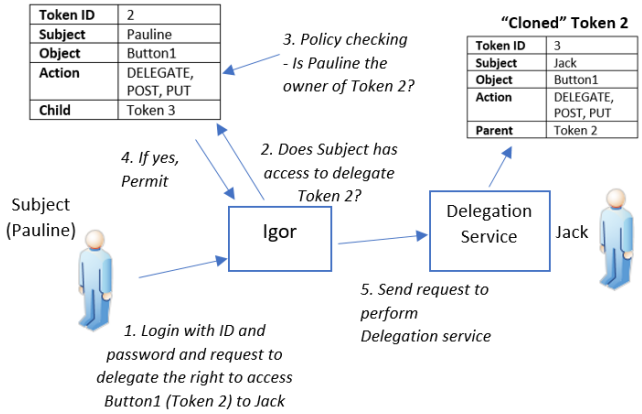


Figure 6 – Delegation process for person

Steps for the delegation process as shown in Figure 6:

1. Pauline logs into Igor (by keying in ID and password) 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 Policy Decision Point (PDP) in Igor responds on the access request from Pauline (Subject).
3. PDP checks the policy on whether Pauline is the owner of Token 2 and has the right to delegate the capability.
4. If yes, Igor executes the Delegation Service, else, the request is rejected.
5. The Delegation Service will first "clone" Token 2 and renamed it to Token 3. Token 3 will be stored in his profile. Igor informs Jack of his new access via 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. Pauline can decide if Jack is allowed to delegate further this capability. If yes, Jack is named as the new Subject of Token 3, else, Token 3 will not have a Subject named and Jack is not allowed to delegate the capability further.

The process to delegate access to a device is similar to the process described for delegation to another user except that Igor does not need to inform the device via email but the new token contents are transferred and stored on the external device. The Token ID is stored in the device's profile in Igor. It is assumed that the communication channel between Igor and the external device is secure.

3.3.5 Transfer Capability

There is also an option to transfer capabilities to another user from the original "owner". This function works just like delegation as discussed in Section 3.3.4 except that the first subject (Pauline) no longer has access to the device after the capability (Token 6) has been transferred to Jack, as shown in Figure 7.

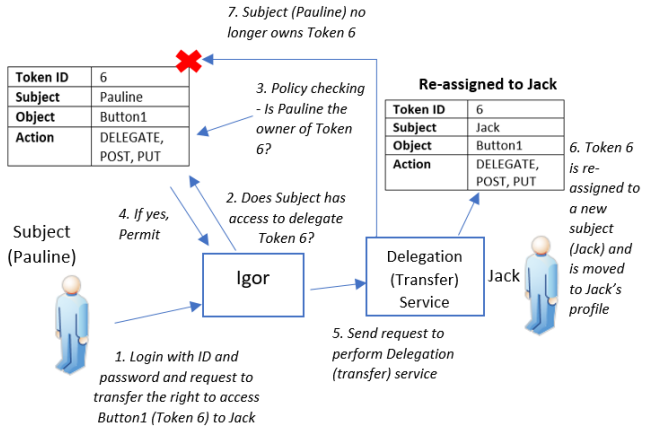


Figure 7 – Transfer capability to another user

3.3.6 Remove access (revocation)

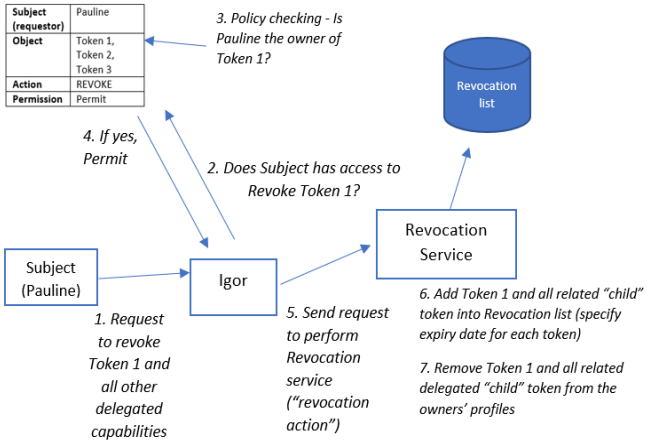


Figure 8 – Removing access (Revocation) process

As shown in Figure 8, subject (Pauline) requests Igor to revoke the access rights of an agent (Token 1). The Policy Decision Point (PDP) in Igor then checks if the subject has the

rights to revoke that access (revocation capability is granted to all the owners of devices when these devices are first registered with Igor). If no, the request is rejected, else, Igor will send request to the Revocation Service to remove Token 1 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. For external capabilities also, the "Refresh" approach is used, where each capability will have a limited lifespan (1 year or any other duration specified upon creation) and after that date, it will need to be refreshed (to check if it has been revoked) with Igor. There is no expiry date for internal capabilities. To revoke internal capabilities, the requestor must first have the access right to do so and if yes, the capabilities are immediately removed from Igor's XML database.

3.3.7 Adding agents and specifying access levels

All agents need to be registered on Igor before any access can be granted. Figure 9 shows the process of adding a new device to Igor.

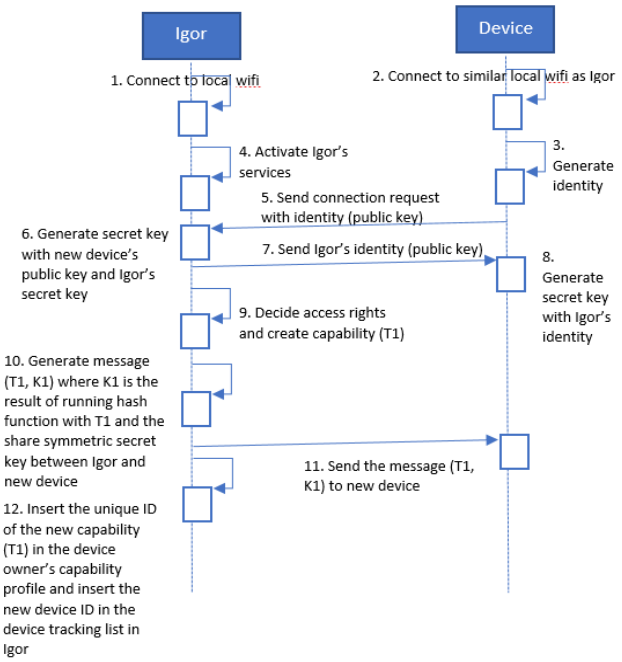


Figure 9 – Adding new device process

First, both Igor and the new device have to be on the same local Wi-Fi[2]. Then, the new device generates a new identity (ID) which also is its Public Key. An example function to generate unique ID can be found in [38]. It then sends a connection request to Igor together with its ID. As mentioned in Section 2, the ECCDH symmetric key agreement protocol can be used to allow both Igor and the new device, each having elliptic-curve public–private key pair to establish a shared secret key. It is assumed that the connection establishment channel is secure by using security protocol like Datagram Transport Layer Security (DTLS) [39].

The owner of the device specifies the access rights to be given to the new device. For example, for a new bell button, the device owner can grant the device the right to ring the house door bell. Igor then generates the message (T1, K1) where K1 = T1* (shared secret key). After that, Igor will send the message back to the new device and at the same time, insert the unique ID of the new capability (T1) in the device owner's capability profile for traceability. At the same time, the new device ID is also stored in the device tracking list in Igor.

For adding new users, Igor will follow the similar process as mentioned above but instead of passing the message (T1, K1), Igor will create a new profile and asks the new user to specify a new password and a unique logon ID for the new user through an interface.

4. SYSTEM DESIGN

This section covers the hierarchical data store, system architecture, token (assertion) design and the access control policy for Igor.

4.1 Hierarchical Data Store

As mentioned earlier, Igor is basically a hierarchical data store (an XML database). There are three basic operations on the database [40]:

- Plugin/ "Helpers" modules – enable the IoT devices to modify/update the database (for example, the BLE Plug-in will update the XML database with all the Bluetooth LE devices detected by the BLE Server)
- Rules – rules which trigger certain actions whenever changes in the XML database are detected (for example, if Pauline's mobile UUID is detected, then, Pauline is home)
- Action plugins – trigger the database changes and allow control over external IoT hardware and software (for example, if it is getting dark, the "Light On" action switches the status and triggers the Light plugin to switch on the lights)

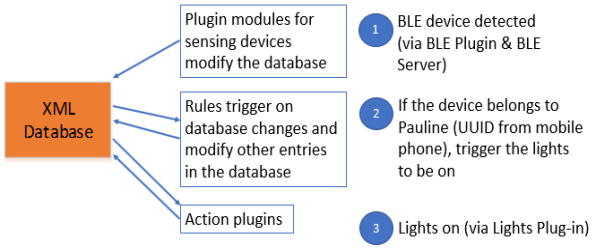


Figure 10 – Igor's BLE device detection

Figure 10 shows an example of an Igor function, the BLE (Bluetooth Low Energy) device detection. In this case, the BLE device, BLE server and Lights plug-in are agents. The BLE server detects and keeps track of which Bluetooth LE devices are in range. Both Igor server and BLE server can be setup on the same device.

4.2 System Architecture

Figure 11 shows the system architecture for Igor. Igor has very low hardware requirements and it can be setup on a

[2] https://en.wikipedia.org/wiki/Wi-Fi

Raspberry Pi3[3] or equivalent devices. As mentioned earlier, Igor functions as the "intermediary" between the IoT device owners and their heterogenous IoT devices. It is completely self-contained (not dependent on any cloud infrastructure). This means that the IoT devices and Igor can continue to function even if the connection to the Internet is down. Igor stores all IoT devices' generated data in its database and owners of the data can decide which information to share and with whom. However, Igor can work together with cloud based IoT servers and the owners can combine this information with other IoT device data. Igor is "state-based", which mean any change in state triggers other actions automatically.

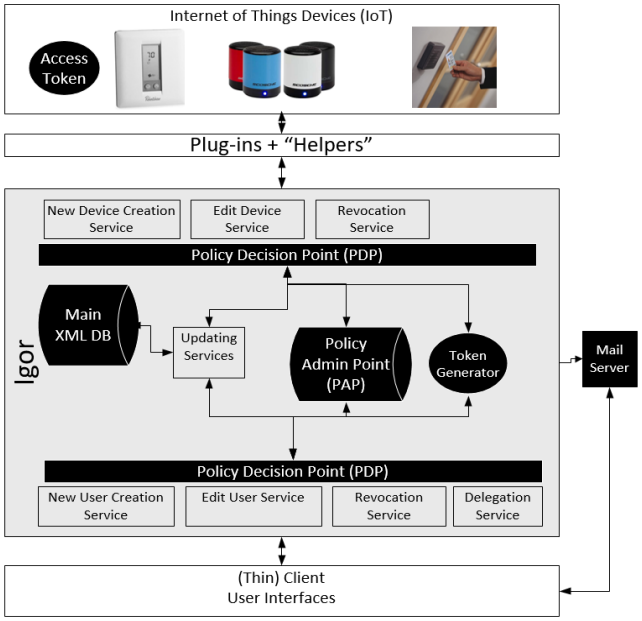


Figure 11 – Igor's System Architecture

This paper has introduced new security and access control modules which comprises five new services:

- *Creation Service* – add new agents into the Igor system
- *Edit Services* – change the current settings of the access rights of the agents. In any case, new tokens are generated to replace the old one.
- *Revocation Service* – revoke accesses of agents and the relevant delegated accesses
- *Policy Decision Point (PDP)* – The concept of PDP and PAP (Policy Administration Point) has been adapted from the XACML architecture [34]. The PDP decides whether access should be granted by first validating whether the token presented is valid or if the user/device is in the revocation list and by the defined policies in the PAP.
- *Delegation Service* – existing users can delegate all or partial of his/her access rights to another person (either permanent or temporarily)

Besides the new access management services, there are other modules/components required to complete the access authorization and XML database updating processes:

- Token Generator (Issuer) – as described in Section 3.3.7, whenever a new agent is added into Igor, tokens (capabilities) are generated to define the access rights the agent will have. The Token Generator service generates these tokens. There are two categories of tokens: internal or external. For a new person/user, Igor stores new tokens under his/her profile in the main XML database (internal) and if the newly added agent is an external IoT device, the Token Generator generates an external token (T1) and creates the message (T1, K1) where K1 = T1* (shared secret key)
- Updating Services – once the authorization processes move completed, the updating services updates the XML database with the requested actions
- (Thin) Client User Interface – users access the functionalities of Igor via a thin client user interface (UI). The design and development of a user-friendly UI is not in the scope of this project
- Mail server – Igor communicates with new or existing users via email by informing them on their new or updated ID and password. The mail server enables Igor to send emails to these users. The mail server is also not in scope for implementation for this project

4.3 Token (Assertion) Design

As mentioned before, the CapBAC model [24] and the authorization framework introduced by [33] are used as a guide 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.

For a token to be valid and usable, the token presented to Igor must contain:

- Which object/service does the decision apply to?
- Which actions (create, read, update and delete) do the decision apply to?
- Which subject (agent) does the token belong to?
- Which Token Generator (ID of Igor) has issued this token?
- Under what conditions are the tokens valid (e.g. expiry date, propagation direction, etc.)

As suggested by [33], a subset of the full XACML and SAML [41] 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"</ob>
</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 [42] format, "***is"*** is the identifier of the

[3] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Assertion Issuer (only available in external token) and "***sk"*** (Subject Key) is the owner of the token, using a public key for confirmation. "***obj***" (object) 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, JSON Web Token (JWT) [43] is used by Token Generator in Igor to generate the outgoing "Authentication Bearer" header of the external access tokens.

The abstraction is: "*Subject may do Action to Resource (object) under Conditions and the other values (id, ii, is) are administrative (for traceability purposes).*

For 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).

4.4 Access Control Policy

4.4.1 Propagation

As Igor is based on an XML database, access control policies are enforced as authorization propagation for hierarchical data stores. Homogenous level of only "top-down" strategy for granting authorization is applied here. Table 2 shows the different types of authorization propagation used in Igor.

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 for all levels

Table 2 – Types of Authorization propagations

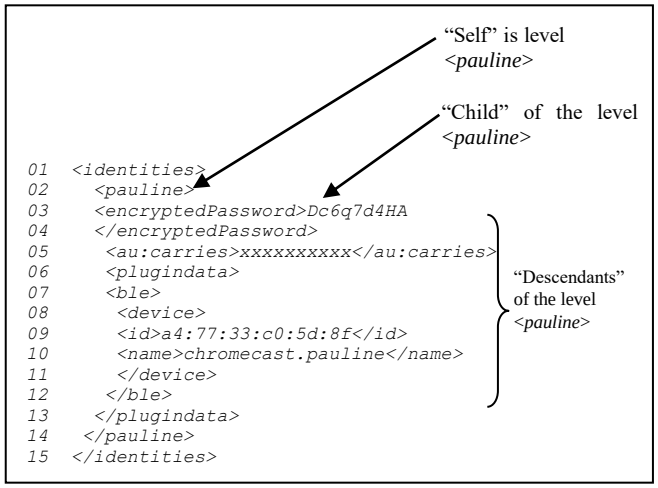


Figure 12 - Sample part of XML Database

There are five types of propagation defined for Igor. "Self" propagation means access to the one specified level only. In the example of XML database in Figure 12, access to the line 02 <*pauline*> means the access is defined only for that line. "Child" propagation for <p*auline*> means access is applied to the level below <*pauline*>, which is line *"<encryptedPassword>Dc6q7d4HA</encryptedPassword>"* only.

"Descendants" propagation means access applies for all levels below <*pauline*>. "Self or Descendants" means access granted for that defined level and all the levels below it. "No propagation" means the access for that level is not granted.

4.4.2 Access Control Permission Type

As an advantage of the CapBAC model, the *Principle of Least Authority* (PoLA) (Least Privilege) is the default [23]. 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 defined in Igor as shown in Table 3.

Permission Type	**Function**	**Description**
POST	Create	The capability to create new lines / command in the XML database
GET	Read	The capability to read the code/data in the XML database
PUT	Update	The capability to update code/data in the XML database
DELETE	Delete	The capability to delete code/data in the XML database

Table 3 – Access Control Permission Types

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. Figure 13 shows a sample list of access tokens defined in Igor's XML database. Each token is given a unique identity (ID). In this example, these tokens belong to Pauline (Subject) and the Xpath (object) is defined in each token, with the access right permissions. For token ID "1" for example, the Object defined is "/data/environment" and the permission "GET" assigned to it with the propagation "Descendants or Self". This means that Pauline has the permission to read only the data in "/data/environment" and the levels below it. For each user registered in Igor, he/she can create, read, update and delete data in his/her own profile (as shown as token ID "3" in Figure 13).

An access control policy table can be drawn for Igor based on the concept of mapping each Subject to the access rights permission and the propagation type for each Xpath (object) in the Igor's XML database mentioned in the earlier sections. An example of the access control policy for Igor can be found in Appendix A. Subject can be any agents (people, devices, plug-ins, etc.).

4.5 Scalability

The advantage of using a state-based XML database is that it provides flexibility to add multiple heterogenous IoT devices and actions can be defined upon changes in state. However, Igor might face inability to cope with overload of concurrent requests. This can be overcome by introducing external firewall functionality restricting the number of simultaneous requests.

It is also possible to have multiple Igors working together and thus, creating distributed centralized access control devices. With the current access control design, there is no limit in adding

more IoT devices and defining their actions in this distributed ecosystem of private Igors. However, the complexity in managing, updating capabilities and revocation lists increases with the additional number of Igors interacting with each other.

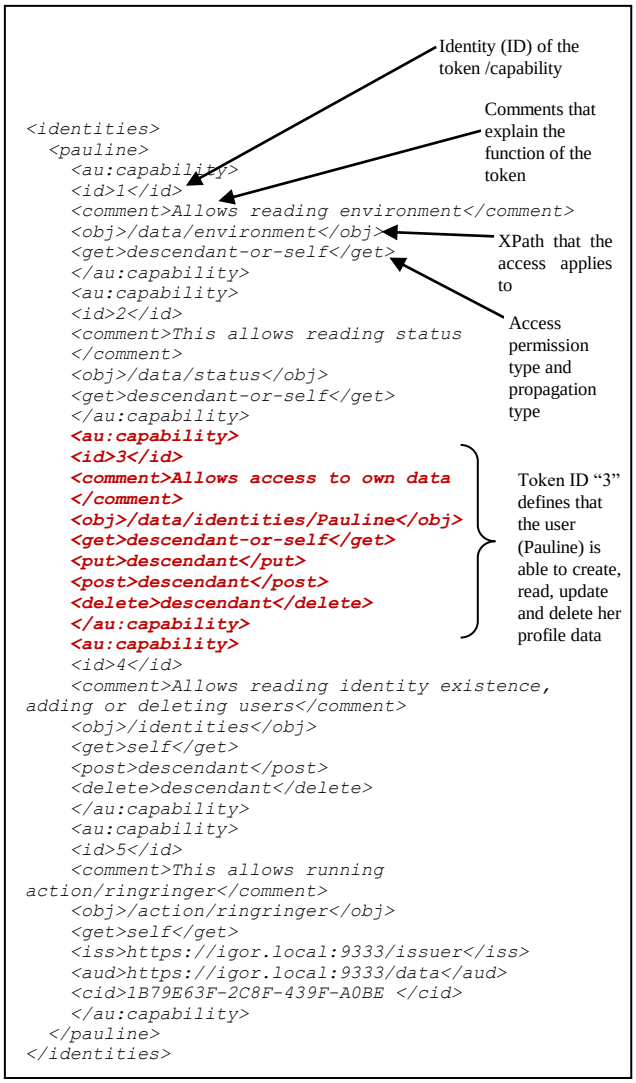


Figure 13 – Sample access token defined in XML Database

4.6 Technical Limitations

Igor is not built to manage multiple concurrent access to the XML database (no access locking mechanism). Some IoT devices are vendor locked and users are forced to use their application to access and store data in their cloud infrastructure. The lightweight database file is not built to store high data loads from connections to thousands of IoT devices. However, the threshold of the data capacity depends highly on the hardware Igor runs on. In the current design also, data sharing through group credentials has not been explored.

During the commissioning of IoT devices to Igor, there could be the possibility of an Eavesdropping [44] attack. This can be overcome by having secure connections (via HTTPS) and performing physical verification that indeed the correct devices are connected. Nevertheless, the authentication and encryptions methods used in this project are "off-the-shelf" market solutions and are suitable for IoT devices. It is assumed that these solutions have been extensively tested and proven to be secure. Their security weaknesses are not addressed in this project.

5. IMPLEMENTATION

The implementation part of the proposed framework was carried out by setting up Igor on a Raspberry Pi 3. The hardware specifications were:

- *Quad Core 1.2GHz Broadcom BCM2837 64bit CPU*
- *1GB RAM*
- *BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board*
- *16GB MicroSDHC memory card*
- *Micro USB power source up to 2.5A*

Raspberry Pi 3 was chosen because we wanted a simple, small and yet sufficiently powerful device to perform all the functionalities required for Igor as the unified access for the IoT.

As mentioned in Section 4.1, the core of Igor is a hierarchical data store (an XML database). Igor is programmed in Python 2, which is simple to code and lightweight, suitable for constrained devices. All the code is shared as open source in Github [45,46] . Igor and other IoT devices are connected through Wi-Fi or CoAP [47]. Figure 14 shows the Raspberry Pi 3, Iotsa [48] (bell ringer) and smart key finder [49] devices used to test the Igor access control framework. We have simulated adding new users (Pauline, Jack and Steven) and adding a new Iotsa device to Igor. The smart key finder is used to allow Igor to sense a BLE device which is tied to a user's identity.

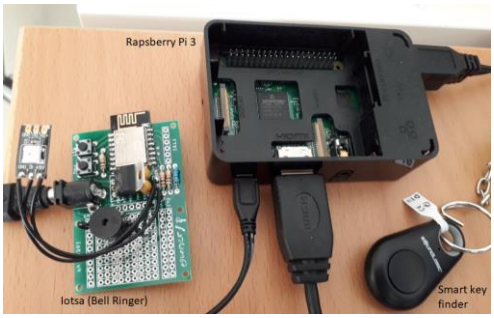


Figure 14 – Raspberry Pi 3, Iotsa (bell ringer) and smart key finder used to implement the Igor access control framework design

The secure https [50] protocol is used to create a secure channel for Igor to connect to the Internet and other IoT devices. One can either buy the SSL certificates [51] from SSL certificate vendors like Verisign [52] or to use self-signed certificates. We opted for self-signed certificate as it is 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. As mentioned before, JWT is used to generate the outgoing "Authentication Bearer" header of the external access tokens. It takes a couple of seconds to generate

these external tokens because of JWT, but as the number of external tokens is far less than the internal ones, performance is bearable.

Different sections of the XML database were added to implement the authorization framework [53] :

- Registered users profile (***/data/identities/user/au:capability)*** – stores all the capabilities this user carry when logged in
- Actions profile ***(/data/actions/action/au:capability)*** –stores the capabilities the respective "action" carry when executing.
- Plug-in data section (***/data/plugindata/pluginname/au:capability***) – stores the capabilities the respective plugins carry when executing.
- External devices section (***/data/externaldevices/au:capability***) – stores the capability IDs granted to the respective devices. This allows early validation whether the requested device is available in Igor
- Default capabilities ***(/data/au:access/au:defaultCapabilities***) – stores the capabilities that are used for any request that has no Authentication: Bearer header, or actions, users or plugins that do not have their capabilities specified.
- Profile of symmetric keys ***(/data/au:access/au:sharedKeys***) – stored the symmetric keys shared between Igor and a single external party. These keys are used to sign outgoing capabilities (and check incoming capabilities). Each key is stored in an au:sharedKey element with the following fields:
    1. iss Issuer.
    2. aud (optional) Audience.
    3. sub (optional) Subject.
    4. externalKey Symmteric key to use.
    5. Keys are looked up either by the combination of iss and aud (for outgoing keys) or iss and sub (for incoming keys).

Other design choices that we made during implementation are:

- Separation of policy administration and checking from the XML database. Separate Python modules are used (checker.py, capability.py and vars.py) [54] to function as the Policy Decision Point (PDP) and PAP (Policy Administration Point). This is to enable better code management and reduce errors when replicating capabilities and performing policy checks.
- There are several approaches to perform delegation. For example, the approach proposed by Gong [17] as discussed in Section 2. In our design, the capability owner can login to Igor using his/her credentials and delegate the capability through a user interface and the receiver of the capability is notified through email that he/she is granted access to certain objects. We have also given the choice to the token owner to specify if the newly delegated token can be further delegated by the new owner or not.
- All external tokens have limited lifespan (1 year or any other duration specified upon creation) and a renewal process is forced upon the token holder after the expiry date. This forces token renewal process for not more than 1 year, instead for an unlimited duration.
- Each agent (device, people, plug-ins, other Igors) have their own profiles or listing in the XML database where the assigned capabilities and other credentials (e.g. logon ID, passwords, secret keys, etc.) are stored for ownership tracking purpose.
- For this implementation, Igor and the external IoT devices use secret symmetric keys to establish trust and JWT is used to transfer generated token from the "token generator" in Igor to the external IoT devices.

Simple user interfaces are implemented to demonstrate the workings of the access control design. Some of the screens implemented for the prototype of Igor can be found in Appendix B.

6. EVALUATION

In this section, we discuss the evaluation conducted with IoT security and privacy issues experts, performance assessment of the system and summarise the results.

6.1 Goals

Our first goal was to test our prototype and obtain feedback on the design of fine-grained access control mechanism with experts in IoT access control, familiar with the ethical and privacy issues related to them. There are three parts of the evaluation:

- To test whether the solution covers all the six requirements discussed in the introduction of this paper.
- To assess if the design choices made are acceptable.
- To gather general feedback from experts on how they feel about the usefulness of the solution and discuss any improvement points.

An additional goal was to perform comparison of the response time of the system before and after the implementation of the proposed security framework.

6.2 Method

Evaluation forms were prepared as shown in Appendix C. 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 [55] 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, Apache Benchmarking (ab) [56] 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
- Executing an update (POST) request function

6.3 Results

6.3.1 Expert Evaluation Results

The detailed results of the evaluation can be found in Appendix D. The main results of the evaluation are discussed here:

a) Lack of friendly user interface for current prototype
As the objective of the prototype implementation is to demonstrate the authorization framework of the proposed solution, user-friendly interfaces and automation of some of the steps are still lacking (Appendix B). Designing friendly user interface is out of scope for this project. Although all of the experts interviewed can understand easily the proposed solution, almost all of the them feel that friendly user interfaces and automation would help normal users to understand better how the system would actually work in real life.

b) Capability transfer should be retractable
In the current design, revocation of capabilities can be done after they have been delegated. When capabilities are transferred to a new owner, the action is irreversible. It is suggested to cater for any mistake done in transferring and allow the administrator or special privileged user to reverse the action.

c) Risk of having centralized revocation list for external capabilities
All of the evaluators agree that the approach of using a revocation list to revoke external capabilities is a good approach. However, at least two of the evaluators raised the concern that there are complications if a revocation list is corrupted or unreachable because of malicious attacks, especially if the revocation list is centralized and many Igors are sharing one list. It is suggested to have distributed revocation lists to lower the risk of depending on one centralized list.

d) Separation of policy administration and checking from the XML database is a good design
Four of the evaluators feel that the separation of the main Igor database from the policy administration and checking, e.g. the Policy Decision Point and Policy Administration point, is a good design as it enables easier code maintenance and reduce errors when performing policy checks.

e) The capability delegation approach is easy to understand
All of the evaluators feel that the proposed capability delegation approach is easy and simple. However, the automation of this process can be improved for external capabilities as the current implementation still needs manual intervention from users to "copy and paste" the JWT generated the to external IoT devices.

f) External capabilities should have limited lifespan
All of the evaluators agree that the external capabilities should have limited lifespan. However, flexibility should be given to the capability owners to decide how long is the lifespan during the generation of new external capabilities. For future expansion, careful thought should be given on how to cater for generation of external capabilities by multiple Igors and multiple revocation lists.

g) Internal capabilities do not need to be encrypted
In the current implementation, there are two types of capabilities: internal and external. Internal capabilities are used to control access to data, actions and plugins within the Igor's database. For better performance, internal capabilities do not to use any encryptions. All the evaluators think this is an acceptable solution with the assumption that the access to the XML file is secure.

h) Performance is good and acceptable
All of the evaluators agree that the subjective performance of the system provided is good and acceptable. One of them even challenged us to use more basic hardware such as Arduino [57] instead of the Raspberry Pi.

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. With the feedback collected, we are able to redesign and improve further the proposed access control framework.

6.3.2 Performance Measurement Results

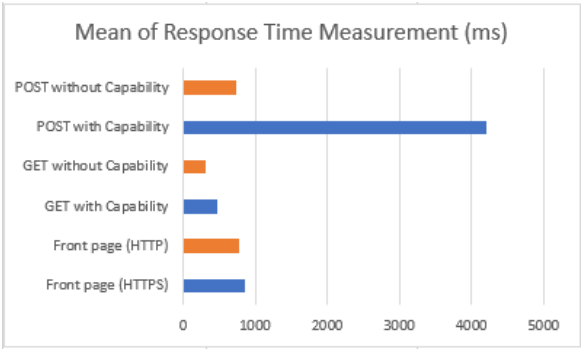


Figure 15 – Mean value of the response time for loading the front page of Igor, executing GET and POST requests before and after implementation of the security framework

Figure 15 shows the breakdown of the response time before and after the implementation of the security framework for Igor. The 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 4. The readings of the system average load performance are also taken but they show no significant difference despite the longer response time when security checks are performed.

	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	737	4200	3464	470%

Table 4 – Comparison of system response time before and after implementation of the security framework for Igor

In summary, performance degradation of the system after the implementation of the security framework for Igor in a lightweight device is acceptable. This fulfils the R6 requirement mentioned in Section 1.

7. DISCUSSION

During the development and evaluation phases of the access control framework a few points for discussion have been raised:

a) XML database design

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.

b) REST versus SOAP

Two very popular approaches to access Web services are: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) [58]. Both approaches have their advantages and disadvantages. REST is chosen in our case as it is simpler to use, flexible and provides a lighter weight alternative. The four different HTTP 1.1 verbs (GET, POST, PUT, and DELETE) provides a simple and light solution for Igor. Furthermore, REST is not limited to XML but it can also output to CSV [59], JSON and RSS [60]. Therefore, for future expansion opportunity for IoT solution, REST is a better option.

c) Higher security or better performance

The main challenge is to create an access control framework that is functional and applicable to solve the research problem in sharing information to certain people or devices at a fine-grained level without compromising on performance and requiring high hardware specifications. The question is how much security can you get while maintaining a performing system? All the evaluators feel that performance is acceptable even when the capabilities and policy checking are implemented on Igor. However, this is a subjective assessment. Security checks and access granting should be seamless allowing users to continue their tasks on the IoT devices without interruptions. 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.

d) Adopting available security solutions and authentication methods

The current implementation of https secure protocol and shared symmetric key generations are based on the off-the-shelf security solutions available in the market today. Shall security technology change with new and more secure authentication and encryption solutions, it should be easy to adopt those changes in the future versions of Igor.

e) Hardware performance baseline

As part of the feedback gathered from the evaluators, hardware specifications for IoT devices are improving rapidly with higher specifications being built, the sizes of the devices are shrinking with every new version at cheaper price. Therefore, the performance benchmark gathered with the current implementation on the Raspberry Pi 3 and Iotsa devices would quickly be outdated when new and better versions of IoT devices are released in the market. Nevertheless, the performance measurements gathered are important to prove that the proposed solution is lightweight and suitable for constrained IoT devices.

f) Centralized versus decentralized solution

In the proposed 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. In a way, allowing different capability owners to grant access to their own capabilities is a decentralized approach [61]. Having a centralized access control approach carries its own benefits as well [62]. The proposal of having multiple Igors working together to control accesses to different IoT devices is trying to get the best of both approaches - decentralized centralization.

8. 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 [3] PAPA model (privacy, accuracy, property and accessibility) presents the main ethical issues hitting the data management effort in the IoT industry. More and more IoT users are seeking ways to control accesses to data generated by the IoT devices they own, even to a detailed level (fine-grained). The main aim of this paper is to work on how to solve data ethical/privacy issues related to IoT as defined by Mason's PAPA model. This paper presents a fine-grained access control solution framework on Igor, a centralized home and office automation solution which is also called the unified access to the IoT to solve the data sharing and access issues. 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 access control model suitable to meet the objectives for this project. Among many access control approaches, the CapBAC solution is the most proven one and has been used since 1966 [19]. One of the main advantage is that it solves the "confused deputy problem" [63] where the system is tricked in believing the presenter has the right to access a request which it actually does not. 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 are still room for improvement in future work:

a) Authentication mechanism

The current implementation of the authentication mechanism covers checking mutual shared symmetric keys and the ID and password presented by the user who wants to login to Igor to view the data they are granted access to. More work should be done to

analyze the strengths and weaknesses of different authentication approaches for IoT devices and select the best solution for Igor. The current implementation uses a self-signed SSL certificate which is not trusted by Internet browsers. To solve this, the implementation can be done using a trusted SSL certificate generated by a trusted certificate authorities like Let's Encrypt [64] and Verisign.

b) User-friendly Interfaces

Adoption of the proposed access control mechanism will increase once user-friendly interfaces are built for novice users. Users should be able to select the list of devices managed by Igor and to view what actions are available for each of the devices. Once found, users can easily select which access to the data or actions they want to grant to their chosen users or devices. All creation, modification, delegation and transfer of capabilities should be seamless and easy to perform via a single user interface accessible by different mobile devices (e.g. mobile phones, tablets or laptops).

c) Capabilities templates and definition of user groups

For future implementation, 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. For example, for those who are categorized as "family", they can have access to all sensitive information related to the household like the power utilization trends, who is in the house at certain times, room temperature readings, etc. The group only needs to be defined once and all the other subsequent access management can be done automatically by selecting the defined groups. A similar concept can be applied when granting access to different IoT devices.

d) Autonomous external capabilities renewal and easy delegation to external parties

External capabilities have a limited lifespan as defined upon creation. The renewal of these capabilities should happen autonomously whenever the capabilities are presented by the subject to request access to certain objects, with the condition that these capabilities are not listed in the revocation list. This approach keeps these external capabilities fresh and in check. More work needs to be done to define how the holder of external capabilities can further delegate them to other parties. For example, a service manager for the IoT devices should be able to grant short-term access to these devices to his colleagues who need to support and perform maintenance on these devices. However, they cannot delegate further the granted capabilities which expire after being used once.

e) Interaction between multiple Igors and revocation lists

The current design allows different Igors to connect with each other as separate devices. However, this is not fully tested and implemented in this paper. 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 Igors in the household or office ecosystem. In the event one Igor is being hacked or attacked, for example, by a DDos attack [65] other Igors in the ecosystem should continue to function and the attack should only be contained to that one impacted Igor.

9. ACKNOWLEDGEMENTS

I would like to thank all evaluators participating in the expert evaluation on the proposed solution and the testing of the prototype: Jorrit Jorritsma, Jeroen Vlieg and Dr Hadi Jamali-Rad from Shell, Sander de Kievit from TNO and Dr Anthony Skjellum from SimCenter, University of Tennessee at Chattanooga (UTC). Also, I would like to thank Jack Jansen, Steven Pemberton and all other colleagues in CWI-DIS for sharing their expertise and support. Finally, to Frank Nack from the Information Studies, University of Amsterdam, for valuable guidance to complete this paper.

10. REFERENCES

[1] Kramp T, van Kranenburg R, and Lange S. *Introduction to the Internet of Things.* In: Bassi A, Bauer M, Fiedler M, et al eds. Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model. Berlin, Heidelberg: Springer Berlin Heidelberg, 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 International Journal of Technology Law and Practice* 2016; 32: 4-15.

[5] European Commission. Conclusions of the internet of things public consultation. 2013. Retrieved 15 March, 2018, from https://ec.europa.eu/digital-single-market/news/conclusions-internet-things-public-consultation .

[6] J. Jansen, S. Pembenton. An architecture for unified access to the internet of things. *XML London 2017 – Conference Proceedings* 2017; 1: 38-42.

[7] Pratchett T. Discworld. 1983. Retrieved 4 April, 2018, from https://en.wikipedia.org/wiki/Discworld .

[8] Gizmocuz. Domoticz. 2018. Retrieved 10 February, 2018, from https://www.domoticz.com/wiki/Main_Page .

[9] Google Developers. Blocky. 2018. Retrieved 27 April, 2018, from https://developers.google.com/blockly/ .

[10] Ierusalimschy R, Celes W, and Henrique de Figueiredo L. Lua (programming language). 1993. Retrieved 27 April, 2018, from https://www.lua.org/ .

[11] SQLite Consortium. SQL lite. 2018. Retrieved 27 April, 2018, from https://www.sqlite.org/index.html .

[12] Fahrny JW, Park K. Authentication and binding of multiple devices. 2013; .

[13] 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.

[14] C. E. Youman, R. S. Sandhu, H. L. Feinstein, and E. J. Coyne. Role-based access control models. *Computer* 1996; 29: 38.

[15] 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.

[16] 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.

[17] Gong L. A secure identity-based capability system. 1989; 56-63.

[18] 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.

[19] Dennis J, Van Horn E. Programming semantics for multiprogrammed computations. *Communications of the ACM* 1966; 9: 143-155.

[20] Henry M. Levy. *Capability-Based Computer Systems.* GB: Digital Press. 1984: .

[21] Tanenbaum AS, Mullender SJ, and Renesse v,R. Using sparse capabilities in a distributed operating system. 1986; .

[22] Hossain M, Hasan R, and Skjellum A. Securing the internet of things: A meta-study of challenges, approaches, and open problems. *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on* 2017; 220-225.

[23] 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.

[24] 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.

[25] Flylib.com. The ECCDH key exchange. 2018. Retrieved 27 April, 2018, from https://flylib.com/books/en/1.188.1.42/1/ .

[26] BitcoinWiki. Elliptic curve digital signature algorithm. 2017. Retrieved 27 April, 2018, from https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm .

[27] W3Schools. XPath tutorial. 1999. Retrieved 27 April, 2018, from https://www.w3schools.com/xml/xpath_intro.asp .

[28] Oracle. What are RESTful web services? 2013. Retrieved 27 April, 2018, from https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html .

[29] 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.

[30] Bertino E, Castano S, Ferrari E, and Mesiti M. Protection and administration of XML data sources. *Data & Knowledge Engineering* 2002; 43: 237-260.

[31] Bertino E, Castano S, and Ferrari E. Securing XML documents with author-X. *IEEE Internet Computing* 2001; 5: 21-31.

[32] Chebotko A, Chang S, Lu S, and Fotouhi F. Secure XML querying based on authorization graphs. *Inf Syst Front* 2012; 14: 617-632.

[33] 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.

[34] 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 .

[35] Introducing JSON. Retrieved 27 April, 2018, from https://www.json.org/ .

[36] Leach P, Mealling M, and Salz R. A universally unique IDentifier (UUID) URN namespace. 2005. Retrieved 27 April, 2018, from https://tools.ietf.org/html/rfc4122 .

[37] Jansen J. Setup BLE server. 2017. Retrieved 22 April, 2018, from https://github.com/cwi-dis/igor/tree/master/helpers/bleServer .

[38] XSLT generate-id() function. 1999. Retrieved 24 Jan, 2018, from https://www.w3schools.com/xml/func_generateid.asp .

[39] Rescorla E, Modadugu N. Datagram transport layer security version 1.2. 2012; .

[40] Jansen J. Igor, your personal IoT butler (readme). 2017. Retrieved 27 Jan, 2018, from https://github.com/cwi-dis/igor .

[41] 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 .

[42] W3C. Date and time formats. Retrieved 27 April, 2018, from https://www.w3.org/TR/NOTE-datetime .

[43] Auth0. Introduction to JSON web tokens. from https://jwt.io/introduction/ .

[44] Investopedia. Eavesdropping attack. 2018. Retrieved 27 April, 2018, from https://www.investopedia.com/terms/e/eavesdropping-attack.asp .

[45] Jansen J. Igor, an extensible home automation server. 2017. Retrieved 10 Feb, 2018, from https://github.com/cwi-dis/igor .

[46] Jansen, J., Sia, P. Igor database. 2018. Retrieved 8 April, 2018, from https://github.com/cwi-dis/igorDatabase.pauline .

[47] Bormann C. RFC 7252 constrained application protocol. 2014. from http://coap.technology/ .

[48] Jansen J. Iotsa - internet of things server architecture. 2016. from https://github.com/cwi-dis/iotsa .

[49] SoundLogic M. Wireless key finder. Retrieved 27 April, 2018, from http://wirelesskeyfinder.xyz/soundlogic_xt_track_find_wireless_bluetooth_key_valuable_finder_iphone_android.php .

[50] Google. Secure your site with HTTPS. 2018. Retrieved 27 April, 2018, from https://support.google.com/webmasters/answer/6073543?hl=en .

[51] Globalsign. What is an SSL certificate? 2018. Retrieved 27 April, 2018, from https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/ .

[52] Verisign. Verisign security services. Retrieved 27 April, 2018, from https://www.verisign.com .

[53] Jansen J. Access control schema. 2018. Retrieved 10 Feb, 2018, from https://github.com/cwi-dis/igor/blob/access-capabilities/doc/capabilities.md .

[54] Jansen J. Igor. 2018. Retrieved 27 April, 2018, from https://github.com/cwi-dis/igor/tree/access-capabilities/igor/access .

[55] Trochim W. Likert scaling. 2006. Retrieved 27 April, 2018, from https://socialresearchmethods.net/kb/scallik.php .

[56] 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 .

[57] Arduino. What is arduino? 2018. Retrieved 27 April, 2018, from https://www.arduino.cc/ .

[58] Mueller J. Understanding SOAP and REST basics and differences. 2013. Retrieved 3 April, 2018, from https://blog.smartbear.com/apis/understanding-soap-and-rest-basics/ .

[59] Python Software Foundation. CSV file reading and writing. Retrieved 27 April, 2018, from https://docs.python.org/3/library/csv.html .

[60] RSS America LLC. What is RSS? 2018. Retrieved 27 April, 2018, from https://www.rss.com/whatisrss .

[61] Sandhu RS, Samarati P. Access control: Principle and practice. *Communications Magazine, IEEE* 1994; 32: 40-48.

[62] Manufacturer benefits from centralized access control.(ACCESS CONTROL). *Security Distributing & Marketing* 2013; 43: 107.

[63] Wikipedia. Confused deputy problem. 2018. Retrieved 27 April, 2018, from https://en.wikipedia.org/wiki/Confused_deputy_problem .

[64] Linux Foundation. Lets encrypt. Retrieved 27 April, 2018, from https://letsencrypt.org/ .

[65] Cloudflare Inc. What is a DDoS attack? Retrieved 27 April, 2018, from https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ .

Appendix A
Example of Access Control Policy for Igor

"Object"	Description	Pauline (Admin with Full Access Rights)	Jack (Sub-Admin)	Steven (Normal User)	Frank (guest)	Button1 (Device)	Button2 (Device)
data/environment	The status of the environment (e.g. whether it is at night, in the morning, if there is a message, etc) Default capability	C^ R^ U^ D^	C^ R^ U^	R^	R^	-	-
data/status/	Default capability. This allows reading status	C^ R^ U^ D^	C^ R^ U^	R^	R^	-	-
data/status/igor/save		C^ R^ U^ D^	C^ R^ U^	R^	R^		
data/status/igor/web		C^ R^ U^ D^	C^ R^ U^	R^	R^		
data/sensors		C^ R^ U^ D^	C^ R^ U^			-	-
data/services/igor	Default capability. Different types of services (e.g. announcetime service, createtoken service) to show on the dashboard	C^ R^ U^ D^	C^ R^ U^	R^	R^	-	-
/static	Default capability. This allows reading static web pages	C^ R^ U^ D^	C^ R^ U^	R^	R^	-	-
/internal/accessControl	Default capability. This allows accessing capability information	C^ R^ U^ D^	C^ R^ U^	R^	R^	-	-
data/people	To detect if the person is present or not (Boolean value)	C^ R^ U^ D^	C^ R^ U^	R^	-	-	-
data/identities	Each person/device are provided a profile with information like token, MAC address, etc	R+ C+ D+	R+ C+	-	-	-	-
data/identities /pauline		C^ R^ U^ D^	R° C°	-	-	-	-

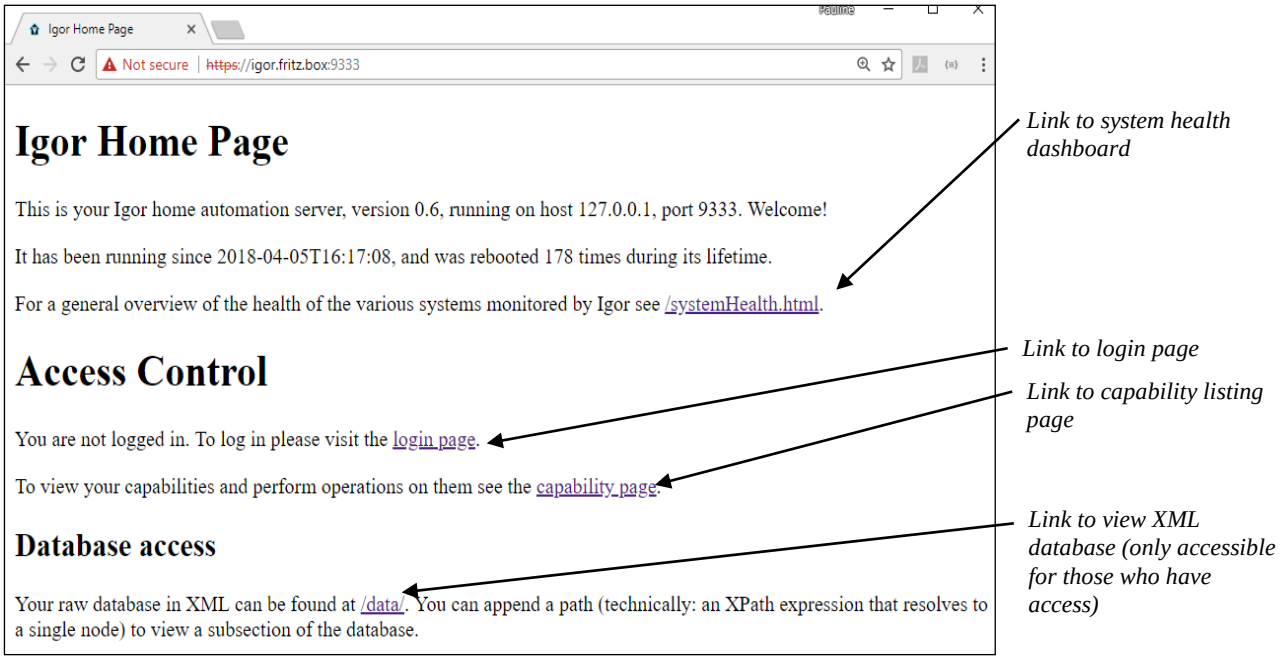
"Object"	Description	Pauline (Admin with Full Access Rights)	Jack (Sub-Admin)	Steven (Normal User)	Frank (guest)	Button1 (Device)	Button2 (Device)
data/identities /jack	Jack can access to his own data in his profile. Administrator cannot read the detail content in his profile but he/she can remove his identity	R° C° D°	R^ C* U* D*	-	-	-	-
data/identities /steven	Steven can access to his own data in his profile. Administrator cannot read the detail content in his profile but he/she can remove his identity	R° C° D°	R° C°	R^ C* U* D*	-	-	-
data/identities /frank	Frank can access to his own data in his profile. Administrator cannot read the detail content in his profile but he/she can remove his identity	R° C° D°	R° C°	-	R^ C* U* D*	-	
Data/actions	Actions to be performed within Igor	C^ R^ U^ D^	R^ C^ U^	R^	R^		
Data/actions/pressbutton1		C^ R^ U^ D^	R° C°	-	-	R^ U*	-
Data/actions/pressbutton2		C^ R^ U^ D^	R° C°	-	-	-	R^ U*

Legend

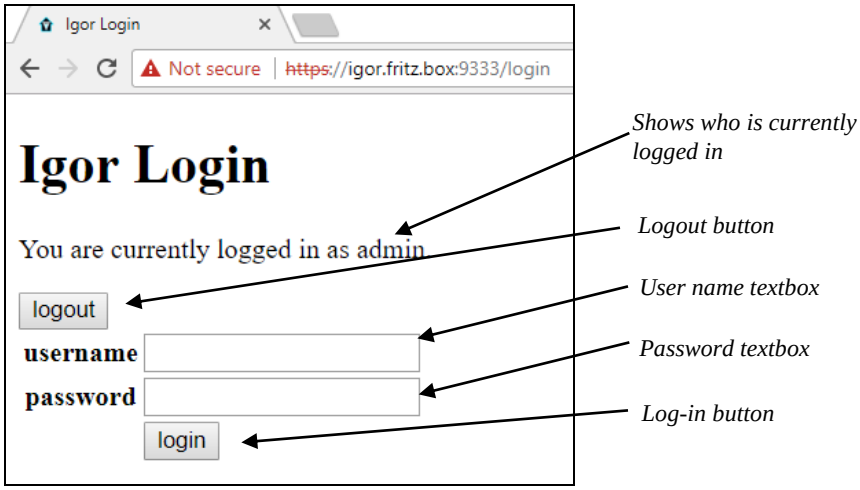
	Permission		Symbols	Propagation
C R U D	Create (POST) Read (GET) Update (PUT) Delete (DELETE)		° + * ^ -	Self Child Descendants Self or Descendants No propagation

Appendix B - User Interfaces for Igor

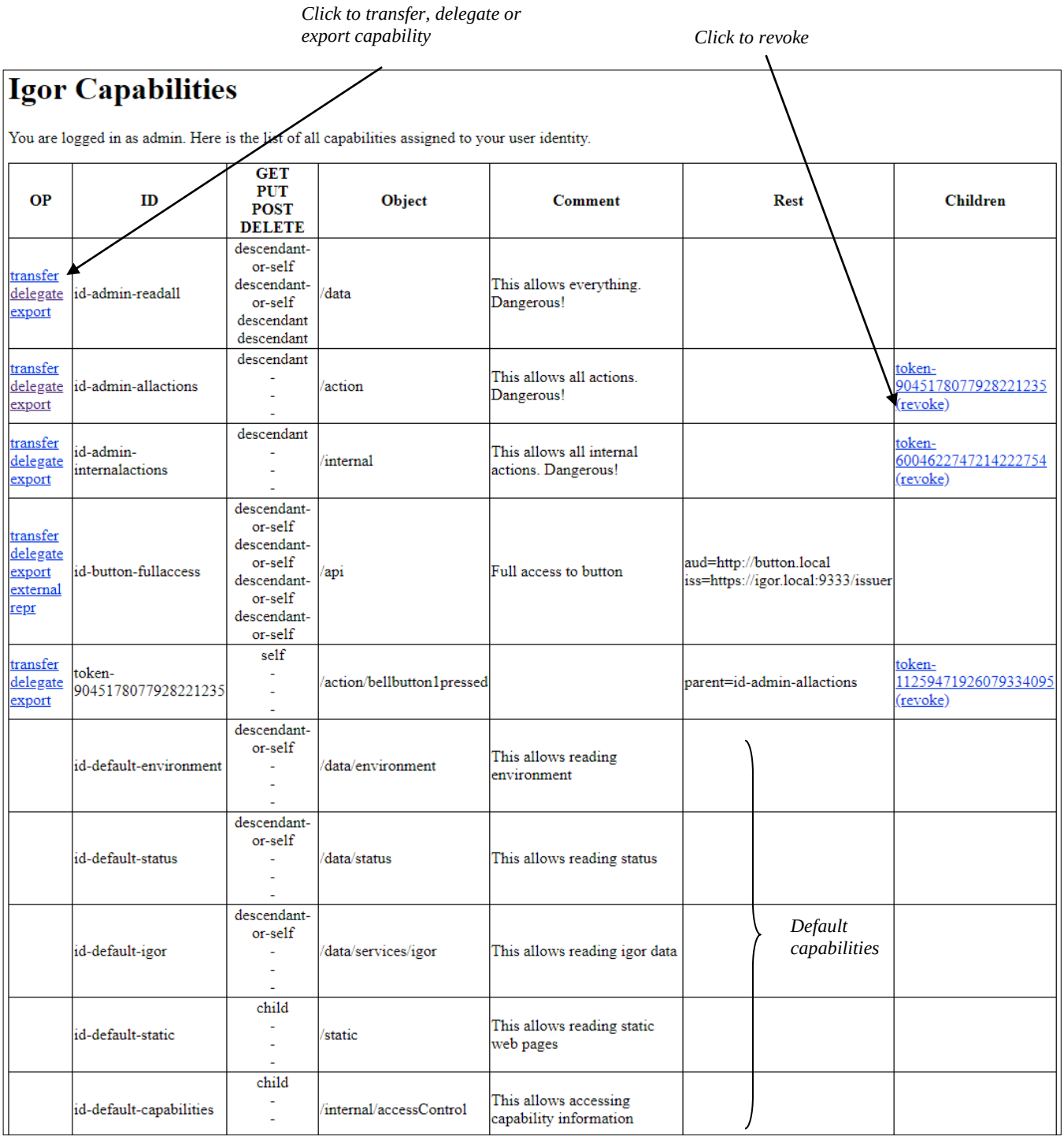
Main Page



Login Page



Capabilities Listing Page



Appendix C – Evaluation Guidelines

Guideline for the Interviewers

Below you will find all the questions. It's important to read the comment section to ask the questions in such a way that you don't influence the answer. Since it's qualitative research, don't be afraid to continuously throw the "Why?" question. It's advised that you **record** and **immediately write** down the answers.

Evaluation Questions

INTRODUCTION		
Get to know about person's expertise in IoT and Access Control Security models		
Intro	What is your education level? (BSc, MSc, PhD)	
Intro	How do you describe your knowledge level on IoT and Access Control security models?	
Adding New Device / User		
a) Requirement 1: New users or devices can be easily added to be IoT solution		
Scenario	Questions	Comments
Adding a new user pauline	• Is it easy to add a new user? • Is it easy to add capabilities to new user? • Any improvement points?	
Adding new devices: Button1 and Ringer	• Is it easy to add a new device? • Any improvement points?	
Delegation & Transfer b) Requirement 2 : Access control can be delegated from the owner to others		
Scenario	Questions	Comments
Login as admin. Go to the Capabilities list screen. Delegate the capability to view /data/profiles/Pauline to user Pauline Login as Pauline, check whether you have access to information in Pauline's profile. Transfer a capability to jack. Once transferred, the capability will no longer available to the first owner. Login as jack, check whether jack is able to access to the information that you just transferred. Note: Delegation/transfer can only be done for users/devices that are already registered with Igor	• Is it easy to delegate capabilities to new users / devices • Any improvement points?	

Revocation c) Requirement 3 : Access control can be revoked by the owners when not needed anymore		
Scenario	Questions	Comments
Login as admin. Revoke the capability that you just delegated to Pauline. Logout. Login as Pauline. Go to the capabilities screen. Check if the capability is now removed from your capability list. Note: There are 2 types of capabilities: internal and external capabilities. External capabilities are assigned to external IoT devices.	• Is it useful to revoke access? • Any improvement points?	
Fine-grained access control d) Requirement 4 : Access control of data should be to the detailed level (fine-grained)		
Scenario	Questions	Comments
Login as ID: admin Go to the Capabilities list screen. Grant access to Pauline the information in /data/people/jack/device Logout. Login as Pauline. Check if Pauline is able to see the information in /data/people/jack/device	• Is the access control model fine-grained enough? • Any improvement points?	
Easily Managed and Modified e) Requirement 5 : Access control should be easily managed and modified for normal users		
Scenario	Questions	Comments
Login as ID: pauline Go to the Capabilities list screen Try to modify access for the capabilities granted to Jack	• Is it useful to manage and modify access? • Is this function simple and easy? • Any improvement points?	
Lightweight – suitable for IoT devices f) Requirement 6 : Access control solution should be lightweight as IoT devices have low processing power (with acceptable performance)		
Scenario	Questions	Comments
The implementation part of the proposed framework is carried out by setting up Igor on a Raspberry Pi 3. The core of Igor are an XML database, a web server and Xpath 1.0.	Is the performance acceptable? Any improvement points?	
Other design choices (Optional)		
Area	Questions	Comments
Design choice 1 - Separation of policy administration and checking from the XML database. A separate Python module is used (access.py) to function as the Policy Decision Point (PDP) and PAP (Policy Administration Point). This is to enable better code management and reduce errors when replicating capabilities and performing policy checks	Do you think this design is useful? Why? Any improvement points?	

Design choice 2 - There are several approaches to perform delegation. Gong's approach is for the initiator to create a delegation message and pass it to the person who need the capability. He/she will then request access from the access server by presenting the message from the owner and the access server will then generate new capability and assign it to the requestor. We did not go with that approach as it presents too many steps and is cumbersome for the users. Instead, the capability owner can login to Igor using his/her credentials and delegate the capability through a user interface and the receiver of the capability will be notified through email that he/she has been granted access to certain objects. We have also given the choice to the token owner to specify if the newly delegated token can be further delegated by the new owner or not.	Do you think this design is useful? Why? Any improvement points?	
Design choice 3 - We have also made the choice to enforce that all external tokens have limited lifespan (1 year) and a renewal process is forced upon the token holder after the expiry date.	Do you think this design is useful? Why? Any improvement points?	
Design choice 4 - Each agent (device, people, plug-ins, other Igors) have their own profiles or listing in the XML database where the assigned capabilities and other credentials (e.g. logon ID, passwords, secret keys, etc.) are stored for ownership tracking purpose	Do you think this design is useful? Why? Any improvement points?	
Design choice 5 - We have chosen to have two types of capabilities: internal and external. Internal capabilities are used to control access to data, actions and plugins within the Igor's database. For better performance, internal capabilities do not to use any encryptions. External capabilities are used by Igor to control access for external IoT devices that are requesting services from Igor. For this implementation, Igor and the external IoT devices use secret symmetric keys to establish trust and JWT is used to transfer generated token from the "token generator" in Igor to the external IoT devices	Do you think this design is useful? Why? Any improvement points?	
Summary		
Questions	Comments	
What is your overall feedback about the <u>usefulness</u> of the design of the fine-grained access control for Igor?		
How would you rate whether the project meets the overall objectives in address ethical/privacy issues we are facing regarding information generated by IoT devices?	Rating from 1 – 5 (where 1 is the lowest and 5 the highest)	

Appendix D – Detail Results of the Expert Evaluation (Part 1: Expert 1-3)

		Expert 1	Expert 2	Expert 3
Profile				
What is your education level		MSc	MSc	PhD
How do you describe your knowledge level on IoT and Access Control security models		Expert	Expert	Moderate - Expert
Adding New Device / User				
a) Requirement 1: New users or devices can be easily added to be IoT solution				
Adding a new user pauline	• Is it easy to add a new user? • Is it easy to add capabilities to new user? • Any improvement points?	Yes, easy to add use/capabilities, however, there is no friendly UI	Yes	Yes
Adding new devices: Button1 and Ringer	• Is it easy to add a new device? • Any improvement points?	Yes, however, there is no friendly UI	Yes	Yes
Delegation & Transfer				
b) Requirement 2 : Access control can be delegated from the owner to others				
Login as admin. Go to the Capabilities list screen. Delegate the capability to view /data/profiles/Pauline to user Pauline Login as Pauline, check whether you have access to information in Pauline's profile. Transfer a capability to jack. Once transferred, the capability will no longer available to the first owner. Login as jack, check whether jack is able to access to the information that you just transferred. Note: Delegation/transfer can only be done for users/devices that are already registered with Igor	• Is it easy to delegate capabilities to new users / devices? • Any improvement points?	Yes	Yes	Yes. Provided that the person/devices first establish trust with Igor. I don't really see this as "delegation" if the trust is not fully established first with Igor. External token delegation makes sense as it is given to an external device.
Revocation				
c) Requirement 3 : Access control can be revoked by the owners when not needed anymore				
Login as admin. Revoke the capability that you just delegated to Pauline. Logout. Login as Pauline. Go to the capabilities screen. Check if the capability is now removed from your capability list. Note: There are 2 types of capabilities: internal and external capabilities. External capabilities are assigned to external IoT devices.	• Is it useful to revoke access? • Any improvement points?	Yes	Yes. The revocation list for external capability is a good idea. However, how do we handle if the revocation list gets corrupted or lost? There is another option to keep a "key list" and delete the list if not needed anymore?	Yes, good design.

Fine-grained access control				
d) Requirement 4 : Access control of data should be to the detailed level (fine-grained)				
Login as ID: admin Go to the Capabilities list screen. Grant access to Pauline the information in /data/people/jack/device Logout. Login as Pauline. Check if Pauline is able to see the information in /data/people/jack/device	• Is the access control model fine-grained enough? • Any improvement points?	Yes	Yes. Good fine-grained. Suggestion to use template for access. E.g. a device comes with a "set of capabilities" granted to it or to its owners. New devices will have list of actions it can perform, along with list of capabilities	Yes
Easily Managed and Modified				
e) Requirement 5 : Access control should be easily managed and modified for normal users				
Login as ID: pauline Go to the Capabilities list screen Try to modify access for the capabilities granted to Jack Access need to be revoked, then, delegate a new capability with new obligations	• Is it useful to manage and modify access? • Is this function simple and easy? • Any improvement points?	Yes	Yes, simple	Yes
Lightweight – suitable for IoT devices				
f) Requirement 6 : Access control solution should be lightweight as IoT devices have low processing power (with acceptable performance)				
The implementation part of the proposed framework is carried out by setting up Igor on a Raspberry Pi 3. The core of Igor are an XML database, a web server and Xpath 1.0.	Is it performing well? Any improvement points?	Yes, performance is good	Yes, performance is good. Don't really need to do performance test as the hardware will always change when we want to go to production. We can choose the higher spec for example, as price drops all the time.	Suggestion to use lower end hardware for Igor, e.g. Arduino instead of Raspberry Pi. Performance is good now.
Other design choices				
Design choice 1 - Separation of policy administration and checking from the XML database. A separate Python module is used (access.py) to function as the Policy Decision Point (PDP) and PAP (Policy Administration Point). This is to enable better code management and reduce errors when replicating capabilities and performing policy checks	Do you think this design is useful? Why? Any improvement points?	Yes, design is useful	Yes, good design	No comment
Design choice 2 - There are several approaches to perform delegation. Gong's approach is for the initiator to create a delegation message and pass it to the person who need the capability. He/she will then request access from the access server by presenting the message from the owner and the access server will then generate new capability and assign it to the requestor. We did not go with that approach as it presents too many steps and is cumbersome for the users. Instead, the capability owner can login to Igor using his/her credentials and delegate the capability through a user interface and the receiver of the capability will be notified through email that he/she has been granted access to certain objects. We have also given the choice to the token owner to specify if the newly delegated token can be further delegated by the new owner or not.	Do you think this design is useful? Why? Any improvement points?	Yes, design is useful	Good, simple design	Yes

Design choice 3 - We have also made the choice to enforce that all external tokens have limited lifespan (1 year) and a renewal process is forced upon the token holder after the expiry date.	Do you think this design is useful? Why? Any improvement points?	With condition its autonomous. If not, its not a good idea. Perhaps to give the owner options to specify if expiry date is required. I do not want to renew the tokens even once a year	Yes, its ok. Renewal should be automated. Owner should be able to choose the lifespan of the token during creation.	This should be dynamic, not tied to a specific duration. But the lifespan concept is good.
Design choice 4 - Each agent (device, people, plug-ins, other Igors) have their own profiles or listing in the XML database where the assigned capabilities and other credentials (e.g. logon ID, passwords, secret keys, etc.) are stored for ownership tracking purpose	Do you think this design is useful? Why? Any improvement points?	Yes, it is easy to maintain	Yes	Yes
Design choice 5 - We have chosen to have two types of capabilities: internal and external. Internal capabilities are used to control access to data, actions and plugins within the Igor's database. For better performance, internal capabilities do not to use any encryptions.	Do you think this design is useful? Why? Any improvement points?	Yes, very obvious, common sense	Yes	Yes
Summary				
What is your overall feedback about the usefulness of the design of the fine-grained access control for Igor?		Useful	Good, very useful	Useful
Other comments		I feel this is a good design - if there are good UI and enough plug-ins to connect to Igor. Currently, its not very user friendly.	The prototype works well with centralized home environment, within the same network. How about devices spread across different networks? Have multiple Igors across different locations and different access technology (Bluetooth, Zigbee, Z-Wave, 6LowPAN)? Maybe to have Igor sitting in the gateway? How about working with other off the shelf harware like Printer? I need to write my own plug-in?	Problem with centralized control is that if Igor is hacked, all the secret keys are being compromised? To improve on the current design, instead of having one Igor as token generator (centralized), why not use multiple Igors (group of Igors) and these collectively create a secret key for new persons/devices or any Igor can be randomly appointed to create the secret key? For example, with the concept of blockchain, secret key can be generated by any node in the chain. The demonstration can be better with more user-friendly interfaces and more automation – future work.
How would you rate whether the project meets the overall objectives in addressing ethical/privacy issues we are facing regarding information generated by IoT devices?	Rating from 1 – 5 (where 1 is the lowest and 5 the highest)	5	5	4.5

Appendix C – Detail Results of the Expert Evaluation (Part 2: Expert 4-5)

		Expert 4	Expert 5
Profile			
What is your education level		PhD	MSc
How do you describe your knowledge level on IoT and Access Control security models		Expert	Moderate - Expert
Adding New Device / User			
a) Requirement 1: New users or devices can be easily added to be IoT solution			
Adding a new user pauline	• Is it easy to add a new user? • Is it easy to add capabilities to new user? • Any improvement points?	Yes. User friendly interfaces are not available now, but can be future work and it's not in the current requirement	Yes, but lack friendly UI
Adding new devices: Button1 and Ringer	• Is it easy to add a new device? • Any improvement points?	Yes	Yes
Delegation & Transfer			
b) Requirement 2 : Access control can be delegated from the owner to others			
Login as admin. Go to the Capabilities list screen. Delegate the capability to view /data/profiles/Pauline to user Pauline Login as Pauline, check whether you have access to information in Pauline's profile. Transfer a capability to jack. Once transferred, the capability will no longer available to the first owner. Login as jack, check whether jack is able to access to the information that you just transferred. Note: Delegation/transfer can only be done for users/devices that are already registered with Igor	• Is it easy to delegate capabilities to new users / devices? • Any improvement points?	Yes. Question: For transferring capabilities, how about when the owner or the Admin "accidentally" transferred wrong rights to the wrong person/device? Currently, there is no way to take the capability back? Suggest to have a "special privileged" role (administrator, etc) to be able to recover / revoke transferred capabilities – For future work	Yes
Revocation			
c) Requirement 3 : Access control can be revoked by the owners when not needed anymore			
Login as admin. Revoke the capability that you just delegated to Pauline. Logout. Login as Pauline. Go to the capabilities screen. Check if the capability is now removed from your capability list. Note: There are 2 types of capabilities: internal and external capabilities. External capabilities are assigned to external IoT devices.	• Is it useful to revoke access? • Any improvement points?	Yes. Revocation list is acceptable solution, but do bear in mind there are weaknesses in revocation list and assume it is never corrupted, hacked, etc.	Yes. Revocation list is a good solution
Fine-grained access control			
d) Requirement 4 : Access control of data should be to the detailed level (fine-grained)			
Login as ID: admin Go to the Capabilities list screen. Grant access to Pauline the information in /data/people/jack/device Logout. Login as Pauline. Check if Pauline is able to see the information in /data/people/jack/device	• Is the access control model fine-grained enough? • Any improvement points?	Yes	Yes
Easily Managed and Modified			
e) Requirement 5 : Access control should be easily managed and modified for normal users			

Login as ID: pauline Go to the Capabilities list screen Try to modify access for the capabilities granted to Jack Access need to be revoked, then, delegate a new capability with new obligations	• Is it useful to manage and modify access? • Is this function simple and easy? • Any improvement points?	Yes	Yes
Lightweight – suitable for IoT devices			
f) Requirement 6 : Access control solution should be lightweight as IoT devices have low processing power (with acceptable performance)			
The implementation part of the proposed framework is carried out by setting up Igor on a Raspberry Pi 3. The core of Igor are an XML database, a web server and Xpath 1.0.	Is it performing well? Any improvement points?	Yes. For the scenario given (use case), using a normal mobile phone can be supported by the current setup of Igor (on Raspberry Pi), which I think meets your project requirements. However, I do not believe it can cover all ranges of IoT. For practical implementation, there are many other "ranges" of IoT – e.g. higher range / very demanding IoT machines, e.g. Smart cars, oil rigs, refineries?	Yes. But I would like to see how to implement this in large scale?
Other design choices			
Design choice 1 - Separation of policy administration and checking from the XML database. A separate Python module is used (access.py) to function as the Policy Decision Point (PDP) and PAP (Policy Administration Point). This is to enable better code management and reduce errors when replicating capabilities and performing policy checks	Do you think this design is useful? Why? Any improvement points?	Yes, good design. The XML database is readable, easy to understand and for support/troubleshooting. At the same time, the "capabilities" are hidden.	Yes, I think this is usually the design I've seen
Design choice 2 - There are several approaches to perform delegation. Gong's approach is for the initiator to create a delegation message and pass it to the person who need the capability. He/she will then request access from the access server by presenting the message from the owner and the access server will then generate new capability and assign it to the requestor. We did not go with that approach as it presents too many steps and is cumbersome for the users. Instead, the capability owner can login to Igor using his/her credentials and delegate the capability through a user interface and the receiver of the capability will be notified through email that he/she has been granted access to certain objects. We have also given the choice to the token owner to specify if the newly delegated token can be further delegated by the new owner or not.	Do you think this design is useful? Why? Any improvement points?	I cannot comment much which approach is better as it is not clear to me how effective the steps of key exchange and trusts are being established (in actual application), user interface, method of the transfer of keys and if the mechanism is flawless cannot be assessed now. I do understand currently, a lot of "copy and paste" exercise in the secret key exchange, which is acceptable now, as authentication and security threats analysis is not in scope for this project.	Yes, its good
Design choice 3 - We have also made the choice to enforce that all external tokens have limited lifespan (1 year) and a renewal process is forced upon the token holder after the expiry date.	Do you think this design is useful? Why? Any improvement points?	It is useful design as long as you have only one Igor, and revocation list not be shared with multiple Igors.	It is useful design as long as you have only one Igor, and revocation list not be shared with multiple Igors.
Design choice 4 - Each agent (device, people, plug-ins, other Igors) have their own profiles or listing in the XML database where the assigned capabilities and other credentials (e.g. logon ID, passwords, secret keys, etc.) are stored for ownership tracking purpose	Do you think this design is useful? Why? Any improvement points?	Yes, great design	Yes, great design

Design choice 5 - We have chosen to have two types of capabilities: internal and external. Internal capabilities are used to control access to data, actions and plugins within the Igor's database. For better performance, internal capabilities do not to use any encryptions.	Do you think this design is useful? Why? Any improvement points?	If the project is about authorization, this is actually not really in scope of this project but I appreciate that this has been thought of. There is indeed if anyone hacks Igor and access to its database directly (standard services server), it will be compromised.	Yes
Summary			
What is your overall feedback about the usefulness of the design of the fine-grained access control for Igor?		Yes, very useful. A good start for many future work by other students.	Yes, the demand for this kind of solution will be more as more IoT devices entering the market
Other comments		I think the PAPA model and the use case are great and it is clear on what you want to achieve. Well done, you have done a very detail software design and development, careful thoughts have been put into the design with different user scenarios being taken care of. Extra efforts have been put into the implementation which is not in project requirement, e.g. the https secure protocol, hiding of the capabilities from the web interface. For future work, more thoughts should be made on how multiple Igors interacting with each other: How to manage each Igor has different data structures but they need to share information, updating each other, and key generations can be very complex without compromising security and open to attacks. Example, when one Igor in the group got D-DOS attack, you will have problem if you have many capability issuers and many revocation lists – the task of updating each other will be jammed. Great work! Do share your paper and git hub after your project. I would like to ask my students to try out your codes and experiment in our lab and continue future work.	I have reservation about the centralized design. What if Igor is down and not available, this means no one can access to all the IoT devices? Should look at more de-centralized solution or how to ensure continuous availability of the service. I feel user friendly interface is important for this to be adopted by normal users.
How would you rate whether the project meets the overall objectives in addressing ethical/privacy issues we are facing regarding information generated by IoT devices?	Rating from 1 – 5 (where 1 is the lowest and 5 the highest)	5	4