Authentication and Access Control for Open Messaging Interface Standard

Narges Yousefnezhad, Roman Filippov, Asad Javed, Andrea Buda, Manik Madhikermi, Kary Främling
Department of Computer Science
Aalto University
Espoo, Finland
firstname.lastname@aalto.fi

ABSTRACT

The number of Internet of Things (IoT) vendors is rapidly growing, providing solutions for all levels of the IoT stack. Despite the universal agreement on the need for a standardized technology stack, following the model of the world-wide-web, a large number of industry-driven domain specific standards hinder the development of a single IoT ecosystem. An attempt to solve this challenge is the introduction of O-MI (Open Messaging Interface) and O-DF (Open Data Format), two domain independent standards published by Open Group. Despite their good compatibility, they define no specific security model. This paper takes the first step of defining a security model for these standards by proposing suitable access control and authentication mechanisms that can regulate the rights of different principles and operations defined in these standards. First, a brief introduction is provided of the O-MI and O-DF standards, including a comparison with existing standards. Second, the envisioned security model is presented, together with the implementation details of the plug-in module developed for the O-MI and O-DF reference implementation.

CCS CONCEPTS

• **Security and privacy → Security services**; **Systems security**;

KEYWORDS

Internet of Things, Open Messaging Interface (O-MI), Open Data Format (O-DF), Messaging Standards, User Authentication, Access Control, Security, Certificate

ACM Reference format:
Narges Yousefnezhad, Roman Filippov, Asad Javed, Andrea Buda, Manik Madhikermi, Kary Främling. 2017. Authentication and Access Control for Open Messaging Interface Standard. In *Proceedings of MobiQuitous 2017, Melbourne, VIC, Australia, November 7–10, 2017,* 8 pages.
https://doi.org/10.1145/3144457.3144461

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144461

1 INTRODUCTION

Internet of Things (IoT) is a vision of future Internet where the barrier between the physical world and digital information will be removed [17]. This concept includes heterogeneous objects such as light bulbs, microwave ovens, smart phones or any intelligent products which are used in our daily life in diverse situations and different applications. Giving the fact that the IoT will potentially connect billions of devices, there is a reasonable concern regarding network capacity and suitable communication patterns and protocols. Requiring a new web browser for each web site would be complicated; in a similar vein, it would be hard to imagine the current approach to IoT connectivity. A similar concern is expressed in [22], who states that "The future is not going to be just people talking to people or people accessing information. It's going to be about using machines to talk to other machines on behalf of people with totally new communication pattern arising between people to machines and machines to machines". Without a clear standardization between different organizations and countries, the expansion of a truly worldwide IoT can be practically impossible to achieve. Obviously the IoT community must follow the example of the World Wide Web, in which TCP/IP and HTTP/HTML boosted the Internet to spread across the world.

There are many protocols designed and used for machine-to-machine (M2M) communication specially in IoT platforms. Message Queue Telemetry Transport (MQTT) designed by IBM was proposed for unreliable networks with high latency and low bandwidth [18]. Constrained Application Protocol (CoAP) was proposed with the purpose of efficient M2M communication in restricted environments with a limited amount of available memory [21]. Advanced Message Queuing Protocol (AMQP) provides services for the middleware to administer the queuing, routing, orientation, reliability and security of the messages [19].

However, sufficiently generic and standardized application-level interfaces are still lacking for exchanging information required by IoT infrastructures. These interfaces must be as complete and flexible as possible to support variety of organizational needs and structures. Open Messaging Interface (O-MI) and Open Data Format (O-DF) messaging standards were proposed as a standard application-level interface for fulfilling such requirements [13]. Nevertheless, these two standards define no specific security model while they clearly state how suitable security mechanism can be applied "on-top" of these standards. This paper examines the development of a security model for these IoT standards considering authentication and access control requirements.

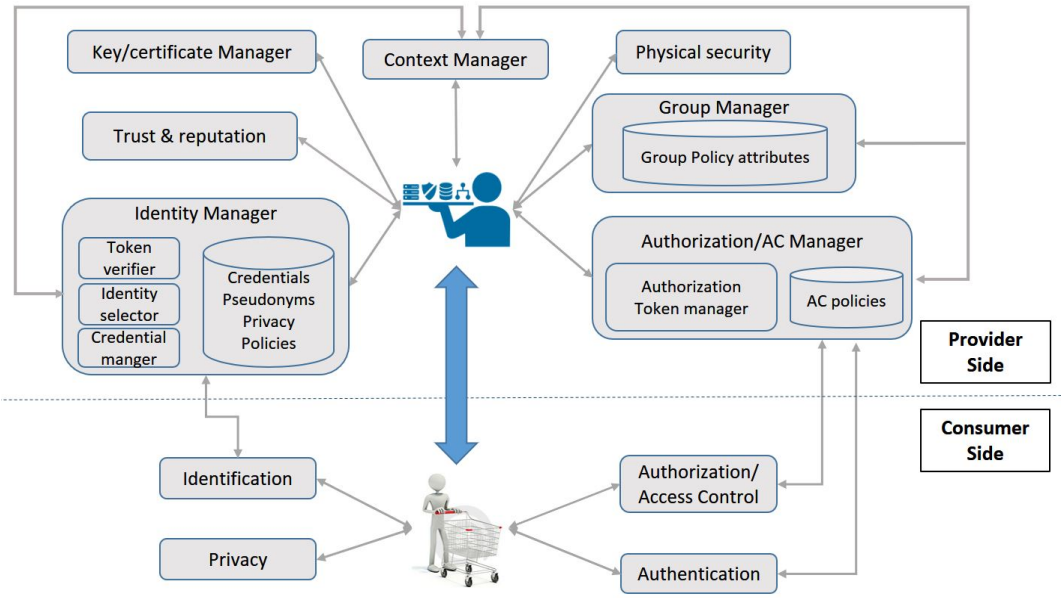

Figure 1: Security requirements for provider and consumer in IoT platform

The rest of this paper is structured as follows. First, a literature review is presented over the existing security modules for IoT protocols. Section 3, 4, and 5 explore the structure of O-MI and O-DF standards, an application for their implementation and their essential security, and privacy requirements respectively. Considering these requirements, a list of design principles and the database structure is demonstrated in Section 6. Section 7 displays the proposed authentication and access control interactions between two devices. After explaining implementation features in Section 8, we conclude with outlines of future work in Section 9.

2 RELATED WORK

Security in IoT implementations is critical during the device design and manufacturing phase, the initialization phase and product update. Correctly implemented, secure IoT deployments should ensure the proper configuration of the basic security requirements needed for data confidentiality, data integrity, and data accessibility as part of the solution. For this purpose, security requirements could be categorized into two groups: provider-specific and consumer-specific. Some of them are necessary for sensors as data providers and others should be fulfilled for users as data consumers (see Figure 1). Users do not need to be worried if their devices are compromised. Service providers are also assured that unauthorized credential sharing is prevented [15].

As a primary requirement for security, an efficient Identity Management (IdM) system is required which could dynamically assign and manage an unique identity for the large number of objects and users [12]. Comparison of five available IdM technologies, that is, OpenId, Liberty Alliance, Card-Space, Shibboleth and Higgins shows that none of them conform the IoT requirements. therefore, new IdM systems should be proposed [15]. On the other hand, to prevent attackers from inserting a malicious sensor node to the network, the identity of the sensor nodes should be authenticated by other nodes. For this purpose, Zhao et al. [25] propose a mutual authentication for IoT nodes. In addition, sensors must convince service provider (or cloud) that they are authenticated to store information there [10].

Furthermore, the provider enables privacy-preserving data sharing, with groups of entities which satisfy specific identity attributes values. It is also responsible for key exchange and providing interoperability between peers. It should also prepare a trusted and reliable IoT environment where users can interact securely with IoT services [9]. In order to provide end-to-end data protection, both in transit and in storage, Wrona [24] proposes an approach based on cryptographic access control and Object Level Protection standard. Mahalle et al. [16] also present an integrated approach to authentication and access control for IoT devices called Identity Authentication and Capability based Access Control (IACAC) model.

Finally, since context-awareness contribution critically to deciding what data needs to be processed for each service [20], for presenting a context-aware security service on IoT scenarios, the detected contexts should be defined and maintained before each security management decision. Before consuming the data (or service), the user needs to run identification, authentication, and authorization. Different kinds of identification technologies such as passive RFID chips and 2D-barcodes are used for IoT environment [5, 11]. To enable user (data owner) to estimate the privacy risk of sharing his sensor data to third party, a privacy management scheme is proposed by Ukil et al. [23]. Overall, there are few integrated solutions available for authentication and access control in IoT environment. Liu et al. [14] propose a new scheme on authentication and access control for IoT users. Mahalle et al. [16] present an authenticated

and access control approach for IoT devices. Hence, this paper analyzes these two requirements.

3 O-MI/O-DF STANDARDS

O-MI and O-DF were created with the same purpose as HTTP and HTML was for the web. For product lifecycle applications, O-MI provides a way for communication framework between products and distributed information systems that consume and publish information on a real-time basis. O-DF is defined as a simple ontology, specified as an extensible XML Schema, for representing the payload in IoT applications. It is intentionally defined in a similar way as data structures in object-oriented programming [7]. It is structured as a hierarchy with an "Objects" element as its top element. The "Objects" element could contain any number of "Object" sub-elements. "Object" elements could have two sub-elements including properties called "InfoItem" and other "Object". Figure 2 shows the possible O-DF hierarchy for these elements and sub-elements.

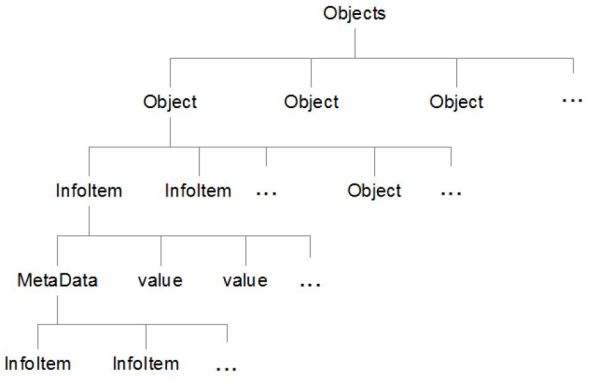

Figure 2: O-DF hierarchy

O-MI can be seen as a transportation mechanism for any payload, exchanging information between different O-MI Nodes across the network. The information can be encoded using most of the widely used formats such as XML, JSON and CSV. Since O-MI Nodes do not have predefined roles, the interaction is performed on "peer-to-peer" basis where every node can act both as a "server" and as a "client" with the other O-MI nodes or with other systems [8]. Furthermore, in order to use an open messaging interface like O-MI in real-time environments, security in terms of authentication and access control are fundamental. In other words, O-MI node administrators must be able to verify the identity of O-MI node users and specify the roles and permissions for every O-MI operations and O-DF Object (s) and InfoItem (s).

4 O-MI REFERENCE IMPLEMENTATION

To understand and learn the standards specifications, it is imperative to associate standards with an implementation and a sandbox environment. The reference implementation (or sandbox) acts as some sort of executable documentation, with request and response examples covering essentially every aspect of the standards. In this way, developers can jump straight into action in understanding what standards supposed to do in reality.

The current reference implementation[1], developed at Aalto University, School of Science, Department of Computer Science consists of three modules:

- *O-MI Node Server:* The server implements all O-MI basic operations and maintains a database where the information about O-DF data model, consisting of Object (s) and InfoItem (s), is stored.
- *Webclient:* This module [1] provides a graphical interface to guide users and developers working as a numbered step-by-step tutorial.
- *Agents:* The agent subsystem provides a mechanism to interact programmatically with the core of an O-MI node. Agents are employed as an intermediary between the hardware (e.g. sensors) and O-MI node to fetch data from data sources.

5 MODULE REQUIREMENTS

One of the main requirements for the security module is to impact as little as possible the current core implementation while providing the desired functionalities. Therefore, a separate self-contained module is created which can be integrated into the existing implementation. The security module needs to fulfil the following requirements:

- Preventing unauthorized access to the resources
- Setting group based rules assigning each user to a special group
- Differentiating permissions according to the O-MI verbs including read and read/write
- Minimizing the server-side account maintenance by using OAuth2 authentication model (e.g. Facebook login)
- Applying recursive permissions mechanism similar to file systems
- Setting an authentication mechanism treating humans and devices in a universal way
- Implementing rules management interface for controlling access policies by system administrator

These requirements are quite common in every security mechanism. Therefore, it is decided to mimic as much as possible the well-known security models such as the one implemented by Unix File System [3]. In comparison with Unix security module, O-DF structure is very similar to a DIRECTORY structure, where objects are folders and infoitems are files.

6 DESIGN PRINCIPLES

Based on the above listed set of requirements, it is possible to select certain technologies, design an overall architecture, and define the features which are needed to be implemented. The main design decisions are summarized in the following list:

- *Two main submodules.* The security module consists of two main parts: Registration (Authentication) submodule and the Access Control submodule. The first one is responsible for handling the registration of new users and their information. It will also manage the authentication process and session handling. The latter module consists of two essential parts: administrator console and access control middleware. The

[1] https://otaniemi3d.cs.hut.fi/omi/node/html/webclient/index.html

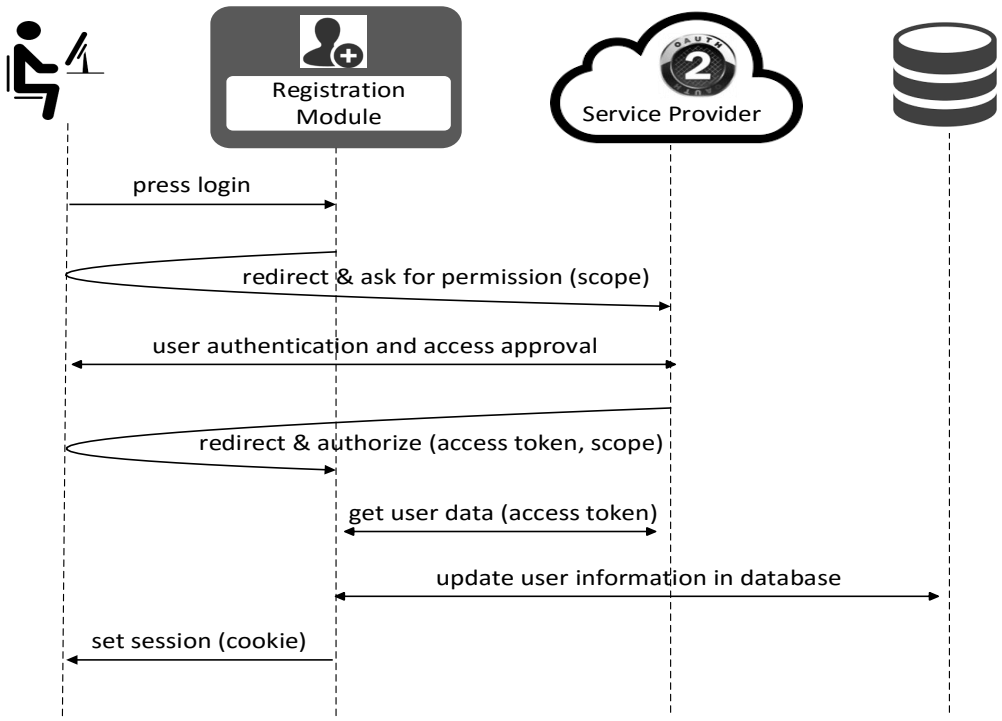

Figure 3: Authentication interaction

first module is a tool for system administrators to manage user groups and policies. The second module processes and authorizes requests made by the users.

- *Separate Database.* To satisfy the requirements of the existing core implementation, it is decided to manage users (groups) and the related policies in a separate database. Although, this choice has negative effect in terms of memory and overall performance, it ensures code modularity and drastically simplifies code management.
- *Servlet based.* The modules are written in Java by utilizing Servlet technology.
- *OAuth2 service provider.* This feature is applied for registration (authentication) module which supports login to the service using user credentials (e.g. Facebook).
- *Certificates Extension.* Since the "Things" does not have dedicated user profile, these devices use client side certificates containing the necessary credentials verifiable by the server.

6.1 Database Schema

At the core of the security module, there is a database, which is used to store and manage users, groups, and the associated access policies (see [4] for more details). The database contains 3 main tables: user, group, and rule. When registering new users, the module obtains username and email only. Users belong to a "default group" after their registration. The default behaviour and access policy for the "default group" can be customized by the O-MI node administrator. The group table contains the list of groups. Users and groups have many-to-many relationship that is implemented using a helper table named USER_GROUP_RELATION. Access rules stored in the rule table, are arguably the most suitable approach for implementation of the security module, as they maintain the association between groups, data objects, and the operations they are allowed to perform (Read/ReadWrite).

7 INTERACTION PRINCIPLES

Given the decision of developing the security module as a separate plug-in for the O-MI reference implementation, the interaction between these two softwares has to be planned. Two main interaction types are identified: 1) User Registration or Authentication 2) Access Control.

7.1 User Registration or Authentication

When a user opens the O-MI webclient interface for the first time (or in case his session has expired), the system redirects the user to perform an authentication process. Since OAuth2 is used for the reasons noticed above (deducting the server burden), the credentials are obtained from a third party website without the need for the user to type them manually. The interaction scheme is shown in Figure 3.

In this scenario, the user interacts with the Registration Module through the web-browser. On the login web-page, the user selects to register using an OAuth provider (in this case Facebook). Registration Module redirects the user to the service provider's web-page where he is asked for a permission of using his personal data. While

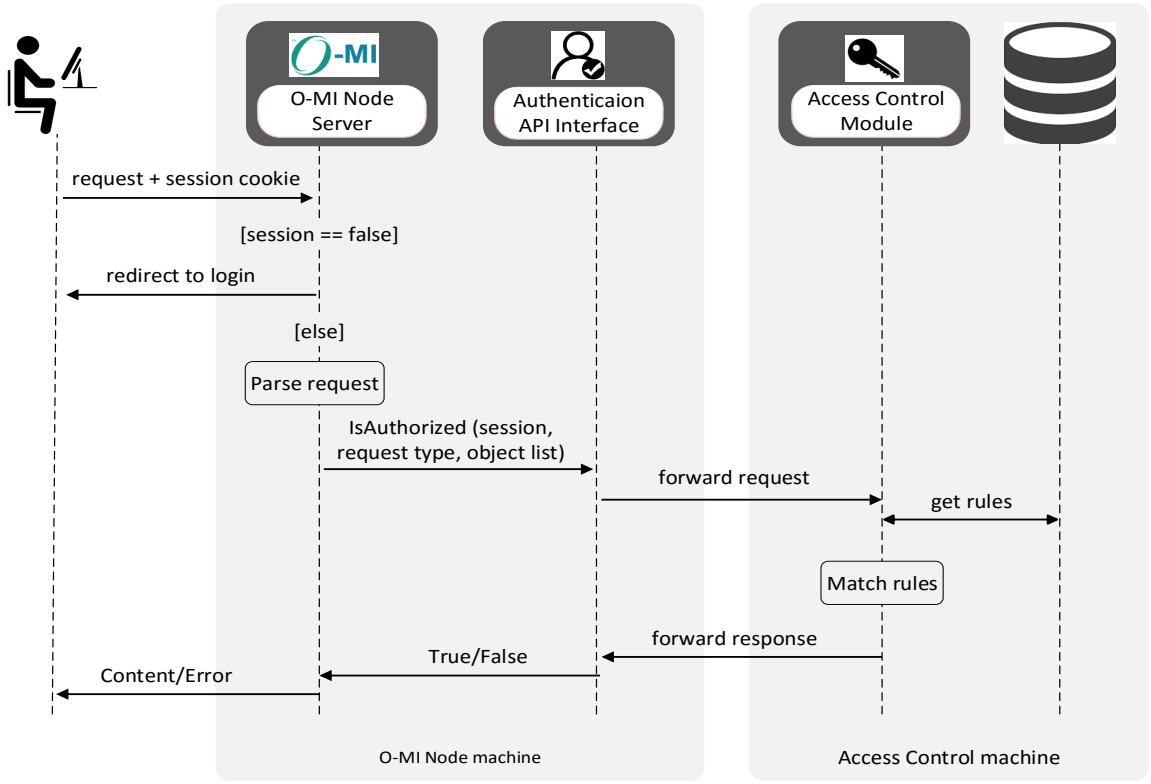

Figure 4: Access control interaction

launching the login dialogue, *scope* parameter defines what kind of data we would like to access in our application. Once the user approved, the service provider redirects him back with the *access token* plus *scope* parameters, which are forwarded to the Regulation Module. Afterwards, the module is able to get personal data of the user by making HTTP queries to the service provider calling vendor-specific Application Programming Interface (API). After getting the data, Regulation Module checks if the user is already registered; if not, user information will be stored in the database. In case no error occurs, the module sets the session *cookies* in the user's browser. Finally the user is authenticated and able to perform queries to the O-MI Node Server using the webclient.

7.2 Access Control

Once a user is registered to the system, it is possible to associate his account to particular groups restricting (or granting) the access to particular data objects. For this purpose, a dedicated user interface, called access management tool, is developed (additional information regarding this user interface can be found in [4]). This user interface interacts with the Access Control Module in the backend, which essentially manages and stores access rules on the database described above.

Access management tool. The user interface of this tool partially resembles the O-MI Node, extending its functionality. This tool is supposed to be used by the administrative staff in charge of the O-MI node. In the user interface, administrators can manage groups and users and set special rules for them. They can create, modify or delete groups and add or remove particular users to given group(s). Mimicking the same user interface used in the reference implementation webclient, it is possible to retrieve all the objects available in the O-MI Node (ReadAll operation) and set access policies for every node of the O-DF tree.

Access policies are simple flags (no-access/read/read-write), which are associated with every node in the O-DF tree. These rules are forwarded to Access Control Module on the backend which is in charge of storing them in the database. When a certain group is selected from the list, the system automatically loads the rules for that group from the database and shows them in the tree. More details regarding the access control module user interface can be found in [4].

Backend. The access control backend, besides storing the configuration set in access management tool in the database, has the fundamental function of interacting with the O-MI node core. The interaction is extremely simple. Essentially the O-MI node asks: "Is the access to the resource X for the user Y and request type Z allowed?", and the module replies "Yes or No" based on rules which are set by the administrators. The module interaction scheme is shown on Figure 4.

This scenario usually starts after the authentication process is completed and the user has a valid session cookie. Once the user is authenticated, he can start to interact with the O-MI Node Server through the webclient. When the user sends a request (e.g. read request for a particular object or set of objects) to the O-MI node, a session cookie is also forwarded. The Server receives and parses the request in case the session is valid, otherwise sends back the user to re-login. It then invokes one of the Authentication API methods, which is part of the O-MI core implementation passing the list of objects as parameters. The invoked method forwards the request to the Access Control Module, by sending an HTTP POST to the service running in localhost.

The Access Control Module selects the appropriate rules from its database, matching them with the request permissions. If the user has appropriate access right on every items of the hierarchy_id list, the service replies True back to Authentication API. In case one or more item violate access rules stored in the database, the service replies False scrapping the entire request. Once the Authentication API has received the True/False answer from the Access Control service, the O-MI Node finally replies to the user with either the requested data or an access denied error.

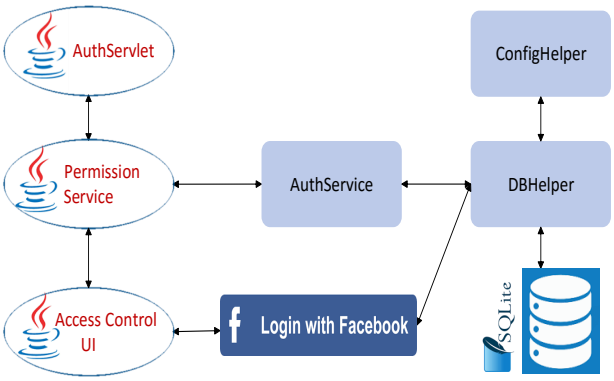

Figure 5: Module class diagram

8 IMPLEMENTATION

In the previous section, the overall design of the security module was presented, while this section focuses on the technologies and implementation details of the developed module. Despite the fact that the O-MI Node Server is written in Scala it is decided to implement security module in Java. This decision is made mostly due to the familiarly with the language and the available server-side framework. In addition it is worth to notice that it is possible to execute Java code from Scala applications as they are both compiled as byte code for the same Java virtual machine. Therefore there was no real concern if a tighter integration between the modules is required.

Since the module is designed to be standalone, an embedded Jetty Servlet container was used to implement the communication (HTTP) with the O-MI Node. Figure 5 depicts the overall code organization. There are three ovals representing the servlets: AuthServlet, PermissionService, and Access Control User Interface (ACUI). AuthServlet is an authentication servlet that handles user authentication and sets up a session. PermissionService, the core servlet of the module, is responsible for majority of functions implemented by the Access Control Module including the backend service for ACUI tool and enforcing access control on behalf of the O-MI Node.

DBHelper is a wrapper class for managing table structure and entities for the SQLite database. The ConfigHelper class contains basic configuration parameters, such as the database name and server URL. The AuthService is an intermediary class responsible for writing object permissions from the O-DF tree structure (using the ACUI tool) into the database.

Finally, the last component interacting with the Access Control Module is the Authentication API which is also known as Facebook login. Essentially it is an external class, which is included in the O-MI Node implementation, providing an abstracted and uniform way to perform authentication and authorization, hiding the implementation details regarding how these functions are performed. This allows future updates to the Access Control Module which will be completely transparent to the O-MI Node.

8.1 SSL Certificates Extension

To test the model, Smart home installation is used as a real-world use case in which the O-MI Node and Access Control Module are applied. Essentially it consists of various sensors connected together to central gateway. The gateway connects the house to the Internet and the Internet Service Provider (ISP) assigns a dynamic IP which might change over time. The usage of dynamic IPs is very common and that was one of the reason why the previous authorization method based on IP whitelist was abandoned. In this scenario, OAuth won't work because the user agent is not a browser and it does not make sense to authorize a physical device (the home gateway) using a Facebook account. To address this issue, it is decided to use client SSL certificates.

Normally, when using HTTPS protocol, the server buys a certificate from an authorized Certificate Authority (CA). When the client connects to the server through HTTPS, it receives the server certificate and checks within the CA if the certificate is valid [2]. For the O-MI Node, mutual authentication is needed, meaning both gateway and O-MI server need to be sure that they are talking to the correct server or right client (see Figure 6). In practise, the O-MI Node creates a certificate and signs it using the server private key. Network distribution of such certificate is also possible, if a trusted software is already running on the target machine, otherwise the certificate has to be physically installed in the device. At this point, when the home gateway establishes a normal HTTPS connection with the O-MI Node, it sends its certificate to the O-MI Node. The O-MI Node using its public key, is able to verify that the received certificate is signed by the server itself, and it can finally authenticate the device.

Obviously, the identity of the device and its access rights must be also configured beforehand using the ACUI tool. In this case, an e-mail address is used as "user-id", which is stored upon registration in Access Control Module database. This e-mail address is one metadata of the client certificate, which is extracted by the O-MI Node

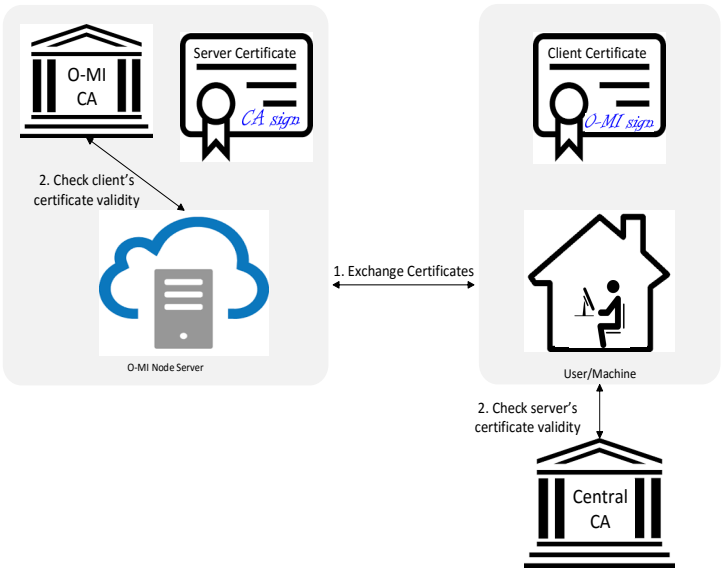

Figure 6: Certificates exchange diagram

and forwarded as "user-id" to the Access Control Module that can finally check if the requests performed by the device, comply with policies stored in the access control database.

It is worth to mention that a real manufacturer could use the product serial number, instead of an email, as "device/user-id", or even better a globally unique identifier, such as the ID@URI concept proposed by Främling et al. [6]. The Uniform Resource Identifier (URI) is the Internet domain of the manufacturer (e.g. samsung.com) whose uniqueness is guaranteed by the DNS, while the Identifier (ID) can be an unique identifier such as a product serial number or a Global Trade Identification Number (GTIN).

9 CONCLUSIONS AND FUTURE WORK

The main focus of this research is to develop a suitable security model for the Open Messaging Interface (O-MI) and Open Data Format (O-DF) standards. We described the design and implementation principles of access control module. The integration with the existing O-MI reference implementation is presented using a smart-home scenario as a testbed. The proposed security module developed can be only partially reused as such for the integration with other systems than the O-MI reference implementation. However, the requirements, the core design decisions, and the code structure are conceived to be generally applicable to other systems, providing a solid foundation for a further abstraction and generality of the used approach.

Although this paper focuses on a particular security problem for messaging interface, authentication and access control, we believe that our notion of using reference implementation to infer security solutions is a powerful tool. In future, it would help the sensor owners to assure about the identity of their devices using device fingerprinting. Furthermore, in our current implementation, client certificate is signed by the self signed CA. In future, we will extend our security module to support automatic creation and management of client side certificate. The new model will be able to handle the regeneration of certificates after expiration or when the certificates have been added to the certificate revocation list.

10 ACKNOWLEDGEMENTS

The research leading to this publication is supported by the European Union's Horizon 2020 research and innovation programme (grant 688203) and Academy of Finland (Open Messaging Interface; grant 296096).

REFERENCES

[1] AaltoUniversity. 2017. *O-MI Node Reference Implementation.* https://otaniemi3d.cs.hut.fi/omi/node/html/webclient/index.html.
[2] Hartley Brody. 2013. *How https secures connections: What every web dev should know.* https://blog.hartleybrody.com/https-certificates/.
[3] David A Curry. 1992. *Unix system security: a guide for users and system administrators.* Addison-Wesley Longman Publishing Co., Inc.
[4] Roman Filippov et al. 2016. *Security model for the open messaging interface (o-mi) protocol.* Aalto University.
[5] Kary Främling, Mark Harrison, James Brusey, and Jouni Petrow. 2007. Requirements on unique identifiers for managing product lifecycle information: comparison of alternative approaches. *International Journal of Computer Integrated Manufacturing* 20, 7 (2007), 715–726.
[6] Kary Främling, Jan Holmström, Timo Ala-Risku, and Mikko Kärkkäinen. 2003. Product agents for handling information about physical objects. *Report of Laboratory of information processing science series B, TKO-B* 153, 03 (2003).
[7] The Open Group. 2014. *Open Data Format (O-DF), an Open Group Internet of Things (IoT) Standard.* http://www.opengroup.org/iot/odf/.
[8] The Open Group. 2014. *Open Messaging Interface (O-MI), an Open Group Internet of Things (IoT) Standard.* http://www.opengroup.org/iot/omi/.
[9] José L Hernández-Ramos, M Victoria Moreno, Jorge Bernal Bernabé, Dan García Carrillo, and Antonio F Skarmeta. 2015. SAFIR: Secure access framework for IoT-enabled services on smart buildings. *J. Comput. System Sci.* 81, 8 (2015), 1452–1463.
[10] Susmita Horrow and Anjali Sardana. 2012. Identity management framework for cloud based internet of things. In *Proceedings of the First International Conference on Security of Internet of Things.* ACM, 200–203.
[11] Mikko Kärkkäinen, Timo Ala-Risku, and Kary Främling. 2003. The product centric approach: a solution to supply network information management problems?

Computers in Industry 52, 2 (2003), 147–159.

[12] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. 2012. Future internet: the internet of things architecture, possible applications and key challenges. In *Frontiers of Information Technology (FIT), 2012 10th International Conference on*. IEEE, 257–260.

[13] Sylvain Kubler, Manik Madhikermi, Andrea Buda, and Kary Främling. 2014. QLM messaging standards: introduction and comparison with existing messaging protocols. (2014), 237–256.

[14] Jing Liu, Yang Xiao, and CL Philip Chen. 2012. Authentication and access control in the internet of things. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 588–592.

[15] Parikshit Mahalle, Sachin Babar, Neeli R Prasad, and Ramjee Prasad. 2010. Identity management framework towards internet of things (IoT): Roadmap and key challenges. In *International Conference on Network Security and Applications*. Springer, 430–439.

[16] Parikshit N Mahalle, Bayu Anggorojati, Neeli R Prasad, and Ramjee Prasad. 2013. Identity authentication and capability based access control (iacac) for the internet of things. *Journal of Cyber Security and Mobility* 1, 4 (2013), 309–348.

[17] Gerben G Meyer, Kary Främling, and Jan Holmström. 2009. Intelligent products: A survey. *Computers in industry* 60, 3 (2009), 137–148.

[18] MQTT.org. 2014. *The MQTT protocol official website.* http://mqtt.org/.

[19] John O'Hara. 2007. Toward a Commodity Enterprise Middleware. *ACM Queue* 5, 4 (2007), 48–55. https://doi.org/10.1145/1255421.1255424

[20] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 414–454.

[21] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. *The constrained application protocol (CoAP).* https://tools.ietf.org/html/rfc7252.

[22] Lu Tan and Neng Wang. 2010. Future internet: The internet of things. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, Vol. 5. IEEE, V5–376.

[23] Arijit Ukil, Soma Bandyopadhyay, and Arpan Pal. 2014. Iot-privacy: To be private or not to be private. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 123–124.

[24] Konrad Wrona. 2015. Securing the Internet of Things a military perspective. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. IEEE, 502–507.

[25] Guanglei Zhao, Xianping Si, Jingcheng Wang, Xiao Long, and Ting Hu. 2011. A novel mutual authentication scheme for Internet of Things. In *Modelling, Identification and Control (ICMIC), Proceedings of 2011 International Conference on*. IEEE, 563–566.