# 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
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 of a standardized technology stack, following the model of the world-wide-web, 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 the Open Group. Although the standards are well compatible, they do not define any 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 of the O-MI and O-DF standards, including a comparison with existing standards is provided. Second, 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 → Systems security**; **Security services**;

## KEYWORDS

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

## 1 INTRODUCTION

Internet of Things (IoT) is a vision of future Internet where the barrier between physical world and digital information will be removed [16]. 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. Imagine how difficult it would be to require new web browser for each web site; in the same sense, it would be hard to imagine the current approach to IoT connectivity. Similar concern is expressed in [21], which 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 truly worldwide IoT can be practically impossible to achieve. It is obvious that IoT community must follow the example of the World Wide Web, in which TCP/IP and HTTP/HTML helped 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 [17]. Constrained Application Protocol (CoAP) was proposed with the purpose of efficient M2M communication in restricted environments with a small amount of memory available [20]. Advanced Message Queuing Protocol (AMQP) provides services for the middleware to take care of the queuing, routing, orientation, reliability and security of the messages [18].

However, there is still a lack of sufficiently generic and standardized application-level interfaces for exchanging information required by IoT infrastructures. These interfaces must be as complete and flexible as possible to support changing organization needs and structures. Open Messaging Interface (O-MI) and Open Data Format (O-DF) messaging standards were proposed as a standard application-level interface that would fulfil such requirements [12]. Nevertheless, these two standards do not define any specific security model while they clearly state how suitable security mechanism can be applied "on-top" of these standards. This paper focuses on the development of a security model for these IoT standards considering authentication and access control requirements.

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 discuss the structure of OMI and ODF 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.


*Mobiquitous'17, November 7–10, 2017, Melbourne, Australia*

DOI: 10.1145/nnnnnnn.nnnnnnn

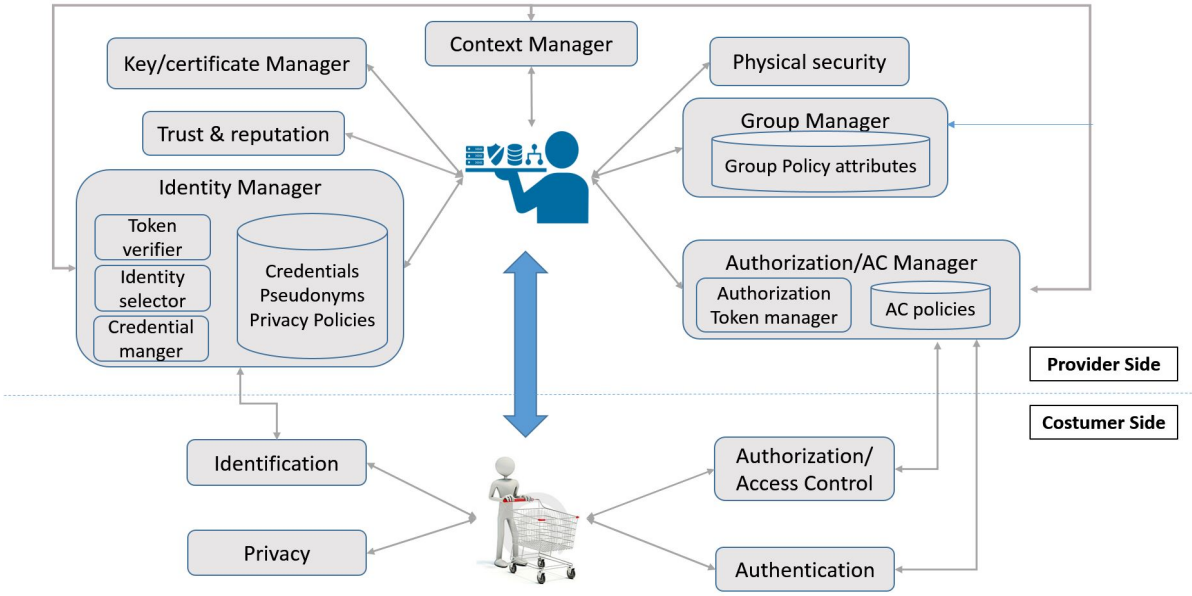


**Figure 1: Security requirements for provider and customer in IoT platform**

## 2 RELATED WORK

Security in IoT implementations is critical either during the device design and manufacturing phase or during the initialization phase or a product update. Correctly implemented, secure IoT deployments should ensure that the basic security requirements needed for data confidentiality, data integrity, and data accessibility are properly configured as part of the solution. For this purpose, security requirements could be categorized into two groups: provider-specific and customer-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 don't need to be worried if their devices are compromised. Service providers are also assured that unauthorized credential sharing is prevented [14].

As a primary requirement for security, an efficient Identity Management (IdM) system is required which could dynamically assign and manage unique identity for large number of objects and users [11]. Comparison of five available IdM technology including OpenId, Liberty Alliance, Card-Space, Shibboleth and Higgins shows that none of them conforms the IoT requirements and new IdM systems should be proposed [14]. 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 [9].

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 [8]. 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. [15] also present an integrated approach of authentication and access control for IoT devices called Identity Authentication and Capability based Access Control (IACAC) model.

Finally, since context-awareness plays a critical role in deciding what data needs to be processed for each service [19], 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, 2D-barcodes, and so on are used for IoT environment [4, 10]. 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. [22].

There are few integrated solutions available for authentication and access control in IoT environment. Liu et al. [13] propose a new scheme on authentication and access control for IoT users. Mahalle et al. [15] present an authenticated and access control approach for IoT devices. This paper focuses on 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

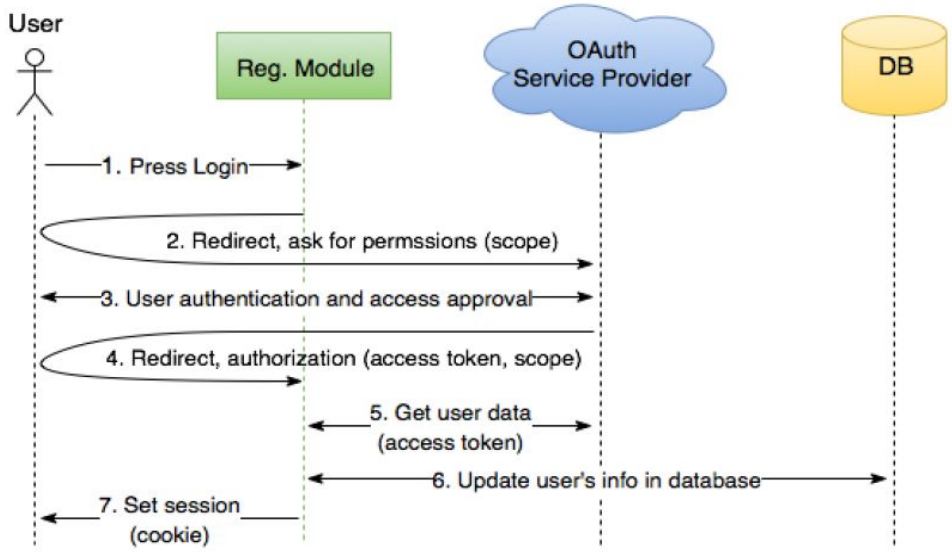


**Figure 2: Authentication interaction**

as data structures in object-oriented programming [6]. 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 3 shows the possible O-DF hierarchy for these elements and sub-elements.

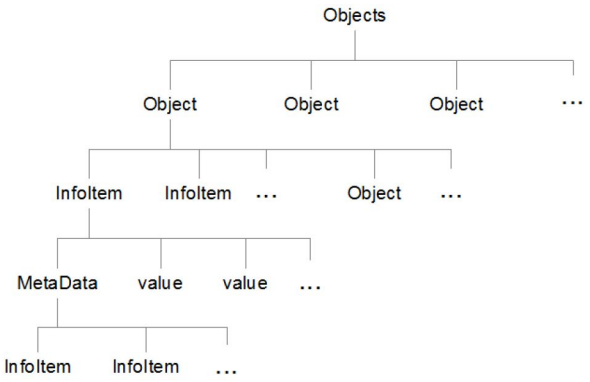


**Figure 3: O-DF hierarchy**

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 [7].

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. O-MI node administrators must be able to specify the roles and permissions for every O-MI operations.

## 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 [23] 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 used 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:

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

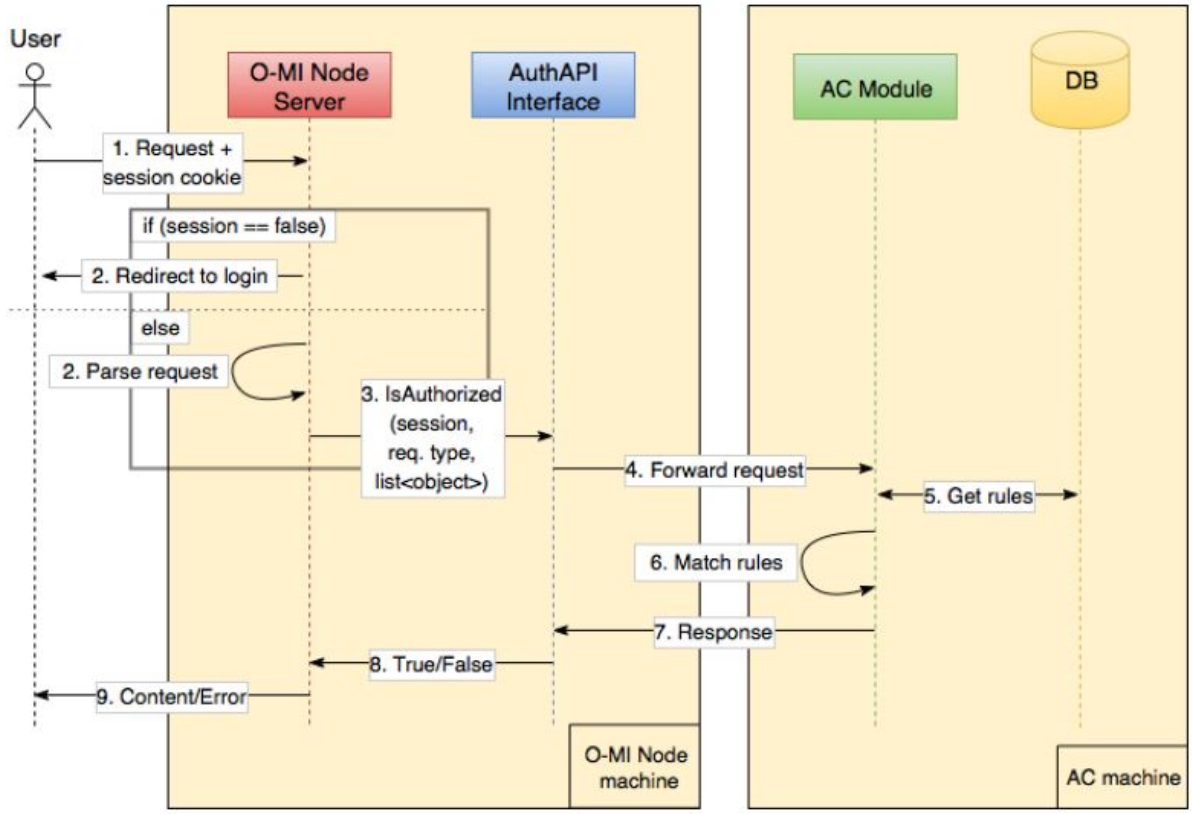


**Figure 4: Access control interaction**

- 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 server-side account maintenance using OAuth2
- 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 [2]. 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 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 module is written in Java by utilizing Servlet technology.
- *OAuth2 service provider.* The feature for registration/authentication module that support the login to the service using user credentials (e.g. Facebook).

- *Certificates Extension.* Since the "Things" do 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 [3] 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 it maintains 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/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. Because OAuth2 is used for reasons explained above, the credentials are obtained from a 3rd party website without the need for the user to type them manually. The interaction scheme is shown in Figure 2.

In this scenario, the user interacts with the Registration (Reg) module through the web-browser. On the login webpage, the user selects to register using an OAuth provider (e.g. Facebook). Registration module redirects the user to the service provider's webpage where he is asked for a permission of using his personal data. After the user agrees, the service provider redirects him/her back with the access token, which is forwarded to the Reg Module. Afterwards, the module is able to get user's personal data by making HTTP queries to the service provider calling vendor-specific APIs (in this case Facebook). After getting the data, Reg Module checks if the user is already registered; if not, user information will be stored in the database. If no error occurs, the module sets the session cookies in the user's browser. Finally the user is authenticated and is 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/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 [3]). 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 [no-access/read/read-write] flags, which are associated with every node in the O-DF tree. These rules are forwarded to Access Control (AC) 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 [3].

**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's 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 O-MI Node receives and parses the request, it then invoke one of the AuthAPI 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 AC Module, by sending an HTTP POST to the service running in localhost.

The AC Module selects the appropriate rules from its database, matching them with the request permissions. If the user has appropriate access right on every item of the hierarchy_id list, the service replies True back to Auth API. In case 1 or more item violate access rules stored in the database, the service replies False scrapping the entire request. Once the Auth API has received the True/False answer from the AC service, the O-MI Node finally replies to the user with either the requested data or an access denied error.

## 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, since Java codes can be executed from Scala applications.

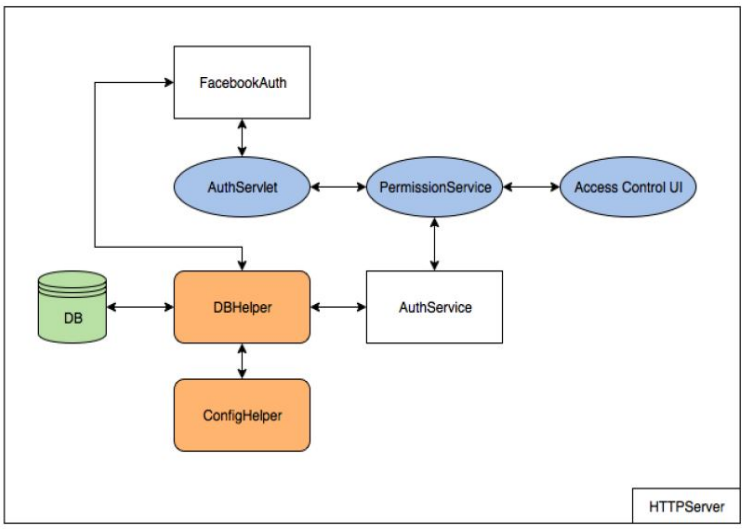


**Figure 5: Module class diagram**

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 blue ovals representing the servlets: AuthServlet, PermissionService, and access management User Interface (UI). 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 AC module including the backend service for access management UI 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 access management UI tool) into the database.

Finally, the last component interacting with the AC module is the AuthAPI. Essentially it is an external class, which is now included in the O-MI Node implementation, providing an abstracted and uniform way to perform authentication and authorization, hiding the implementation details regarding how this functions are performed. This allows future updates to the AC 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 AC 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. 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 [1]. 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 AC management UI tool. In this case, an e-mail address is used as "user-id", which is stored upon registration in AC module database. This e-mail address is one metadata of the client certificate, which is extracted by the O-MI Node and forwarded as "user-id" to the AC module that can finally check if the requests performed by the device, comply with policies stored in the AC 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. [5]. 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).

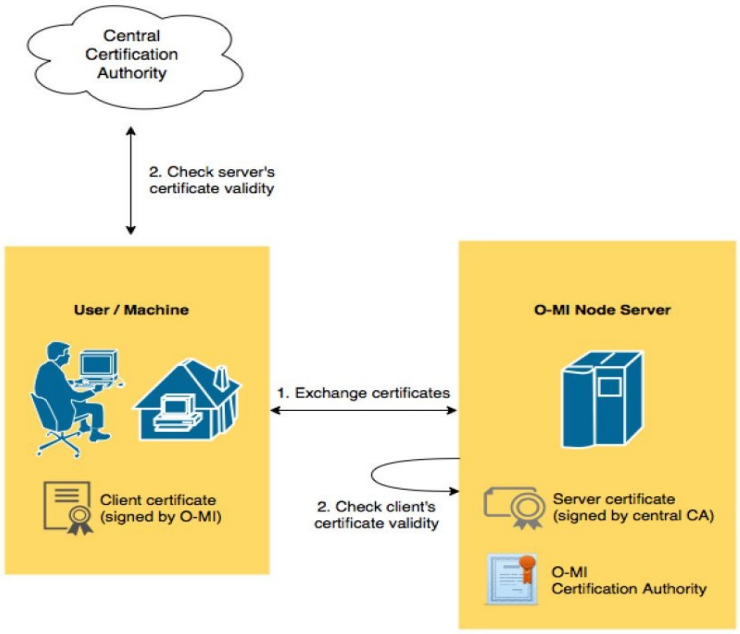


**Figure 6: Certificates exchange diagram**

## 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.

## REFERENCES


[1] Brody, H. How https secures connections: What every web dev should know, 2013.

[2] Curry, D. A. *Unix system security: a guide for users and system administrators*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[3] Filippov, R., et al. Security model for the open messaging interface (o-mi) protocol.

[4] Främling, K., Harrison, M., Brusey, J., and Petrow, J. Requirements on unique identifiers for managing product lifecycle information: comparison of alternative approaches. *International Journal of Computer Integrated Manufacturing 20*, 7 (2007), 715–726.

[5] Främling, K., Holmström, J., Ala-Risku, T., and Kärkkäinen, M. Product agents for handling information about physical objects. *Report of Laboratory of information processing science series B, TKO-B 153*, 03 (2003).

[6] Group, T. O. Open data format (o-df), an open group internet of things (iot) standard, 2014.

[7] Group, T. O. Open messaging interface (o-mi), an open group internet of things (iot) standard, 2014.

[8] Hernández-Ramos, J. L., Moreno, M. V., Bernabé, J. B., Carrillo, D. G., and Skarmeta, A. F. Safir: Secure access framework for iot-enabled services on smart buildings. *Journal of Computer and System Sciences 81*, 8 (2015), 1452–1463.

[9] Horrow, S., and Sardana, A. Identity management framework for cloud based internet of things. In *Proceedings of the First International Conference on Security of Internet of Things*, ACM (2012), 200–203.

[10] Kärkkäinen, M., Ala-Risku, T., and Främling, K. The product centric approach: a solution to supply network information management problems? *Computers in Industry 52*, 2 (2003), 147–159.

[11] Khan, R., Khan, S. U., Zaheer, R., and Khan, S. Future internet: the internet of things architecture, possible applications and key challenges. In *Frontiers of Information Technology (FIT), 2012 10th International Conference on*, IEEE (2012), 257–260.

[12] Kubler, S., Madhikermi, M., Buda, A., and Främling, K. Qlm messaging standards: introduction and comparison with existing messaging protocols. In *Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics*. Springer, 2014, 237–256.

[13] Liu, J., Xiao, Y., and Chen, C. P. Authentication and access control in the internet of things. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, IEEE (2012), 588–592.

[14] Mahalle, P., Babar, S., Prasad, N. R., and Prasad, R. Identity management framework towards internet of things (iot): Roadmap and key challenges. In *International Conference on Network Security and Applications*, Springer (2010), 430–439.

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

[16] Meyer, G. G., Främling, K., and Holmström, J. Intelligent products: A survey. *Computers in industry 60*, 3 (2009), 137–148.

[17] MQTT.org. The mqtt protocol official website., 2014.

[18] O'Hara, J. Toward a commodity enterprise middleware. *Queue 5*, 4 (2007), 48–55.

[19] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys &*

*Tutorials 16*, 1 (2014), 414–454.

[20] Shelby, Z., Hartke, K., and Bormann, C. The constrained application protocol (coap).

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

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

[23] University, A. O-mi node reference implementation, 2017.

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

[25] Zhao, G., Si, X., Wang, J., Long, X., and Hu, T. A novel mutual authentication scheme for internet of things. In *Modelling, Identification and Control (ICMIC), Proceedings of 2011 International Conference on*, IEEE (2011), 563–566.