An AAA Solution for Securing Industrial IoT Devices using Next Generation Access Control

Katyayani Kiranmayee Kolluru*†, Cristina Paniagua*, Jan van Deventer*, Jens Eliasson*,
Jerker Delsing* and Rance J.DeLong‡
*Dept. of Computer Science, Electrical and Space Engineering
Luleå University of Technology, Luleå, Sweden 97187
†Email: katyayani.kolluru@ltu.se
‡The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire RG1 1AX, UK

***Abstract*—Industry 4.0 is advancing the use of Internet of Things (IoT) devices in industrial applications, which enables efficient device-to-device (D2D) communication. However, these devices are often heterogeneous in nature, i.e. from different manufacturers, use different protocols, etc. and adds requirements such as security, interoperability, etc.**

To address these requirements, the Service-Oriented Architecture-Based (SOA) Arrowhead Framework was previously proposed using the concept of local clouds. These local clouds provide a set of mandatory and support core systems to enable industrial automation applications. One of these mandatory core systems is an Authentication, Authorisation and Accounting (AAA) system, which is used to authenticate and provide access control to the devices in a local cloud. In an industrial context, with multiple stakeholders, the AAA must support fine-grain access control. For example, in a distributed control loop, a controller should only have read access to its sensor such as a flow meter and write access to its actuator, such as a valve. The controller should not have access to any other information besides what is needed to implement the desired functionality.

In this work, an NGAC-based AAA solution to achieve fine-grain service level access control between IoT devices has been proposed and implemented. The solution is presented using a district heating use case.

I. Introduction

THE Fourth industrial revolution came about when components or devices in an industrial setting were connected to the Internet and communicate to other machines or persons. These components comprise sensors (e.g. temperature sensors, flow meters, etc.), actuators (e.g. pumps, valves, etc.) that have an electronic unit and can connect to the Internet. These components or systems, comprising physical parts and embedded systems, are often designated as Cyber-Physical Systems (CPS). When connected to the Internet, these "things" form an "Internet of Things" (IoT) network. The architects of Industry 4.0 [1] and the Industrial Internet Consortium (IIC) [2] were able to recognise this early on and proposed reference architectures such as RAMI 4.0 [3] and IIRA [4].

These new opportunities to communicate freely have enabled many solutions that were not feasible until now, which is why it is referred as a "revolution". At the same time, there is a dark side to these opportunities. Confidential information must be vigilantly guarded and commands to actuators should not be compromised. Therefore, stakeholders (machine or person) must be authenticated and authorised to communicate with other stakeholders. Additionally, their interactions should be accounted.

One proposed solution is the Arrowhead Framework [5] which offers a suite of systems that enable secure and scalable industrial interoperability with runtime binding. One of the basic modules of the Arrowhead Framework is the mandatory core system that handles Authentication, Authorisation and Accounting. The granularity of access control within this framework was more or less coarse, i.e. at the system level; any system that was allowed to consume services from a service producer could consume all the services it produced. A system in the Arrowhead Framework is a software module hosted on IoT devices, whereas the hardware unit itself is referred to as a "Device". The purpose of this work is to show that a fine-grain access control can be achieved with the integration of the Next Generation Access Control (NGAC) standard [6]. It also discusses the secure authentication procedure along with accounting of service exchanges in a local cloud.

This work continues with the background to the Arrowhead Framework, AAA system and NGAC standard followed by the proposed solution and its application to a district heating use case.

II. Background

In this section, the background to the Arrowhead Framework, AAA System, NGAC standard and the motivation for this work are described.

A. The Arrowhead Framework

The open source Arrowhead Framework is an SOA-based framework for industrial monitoring and control. Its central concept is to support most types of devices and systems that interact with each other and to enable the exchange of information regardless of underlying protocols and semantic solutions. The Arrowhead Framework as per SOA paradigm enable loosely coupled services to communicate in a late-binding fashion. It specifies a number of core systems such as ServiceRegistry, Orchestration, Authorisation, etc., which provide sufficient functionality to enable the formation of local clouds. A local cloud in the Arrowhead Framework is a self-sustained system of systems with required functionality to

978-1-5386-6531-2/18/$31.00 ©2018 IEEE

support IoT automation tasks and maintain low latency in the communication between the IoT devices. Besides the core systems, there are a number of support systems for data and protocol translation, data storage, quality of service, etc. On top of the core system services, users can develop application-specific services that together address some specific tasks. By using the distributed nature of the Arrowhead Framework, the core and support system services can be used in a very high number of applications. This saves a substantial amount of engineering and development time since only the application-specific parts need to be implemented. All other functionally can be largely reused.

The Arrowhead Framework mandates an integral Authorisation system as a core system for authenticating application systems and authorising them to consume services from other systems within the local cloud. Optionally, it should be able to account service exchanges between different systems within the local cloud. The Arrowhead Framework defines two different systems for this purpose, namely the Authentication and Authorisation (AA) system and the Authentication, Authorisation and Accounting (AAA) system. An AA system only verifies the identities of service consumers and grants access rights to the requested service producer based on the stored access rules, whereas the AAA system additionally accounts these service exchanges.

Many off-the-shelf solutions like RADIUS [7], TACACS+ [8], etc., provide centralised AAA services, which are mostly used for the authentication and authorisation of users connecting to a network and also to account their network service usage. However, these solutions cannot be directly adopted by the Arrowhead local clouds as the AAA requirements in a local cloud vary from network access context as discussed previously. Hence, a AAA system that performs mandatory core services as defined in the Arrowhead Framework for the Authorisation has been designed in the current work. To perform the tasks of an Authorisation system, the AAA system must communicate with the devices connected to the local cloud. The IoT devices generally use Constrained Application Protocol (CoAP) [9] or Message Queuing Telemetry Transport (MQTT) [10] protocols to interact with each other as they are lightweight compared to the Internet standard protocol HTTP. The MQTT protocol uses a publish/subscribe communication pattern and is only used for machine-to-machine communications, while the same cannot be used by IoT devices to communicate with the AAA system. Another alternative, CoAP, can handle request/response messages between any two systems. This protocol is most suitable for IoT devices to communicate with the AAA system. Thus, an AAA system has been designed in this work using the CoAP protocol for communicating with devices connected to a local cloud in a light weight manner.

B. Authentication, Authorisation, and Accounting System

The tasks designated to the AAA system and the potential ways in which they could be performed are explained in this sub-section.

1) Authentication: Authentication refers to the process or action of proving one's identity. This is generally performed by using passwords. The IoT devices in a local cloud could adopt this approach to authenticate themselves to the AAA system, but this alone will not be sufficient as these IoT devices might be exposed to external environments and hence, the local cloud's security could be enhanced by adopting a mutual authentication between any two systems that interact with each other in addition to their authentication with the AAA system. An authentication process that uses passwords becomes cumbersome in this case as multiple devices may be connected to the local cloud and authentication must be performed between all the devices that interact with each other. Alternatively, this could be achieved using protocol-level security measures.

2) Authorisation: Authorisation refers to the function of specifying access rights/privileges to resources in the context of the IoT. A set of rules has been defined in the Authorisation system to govern device-to-device interactions. Many Access control solutions such as Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), Capability-Based Access Control (CapBAC), etc., exist to secure such service exchanges among IoT devices [11]. None of these solutions is the best fit for all IoT use cases. A comprehensive list of different access control models and the pros and cons of using them in an IoT environment is discussed in the work by Ouaddah et al. in [11]. In their evaluation, the RBAC has been found to be a poor fit for an IoT environment mainly due to the increased complexity of achieving interoperability and scalability of roles, whereas ABAC and CapBAC are seen as the next best fit for IoT environments that address the above issues. As an initial step, the authors of the current work considered ABAC for defining access control rules. There are two common standards in ABAC: Extensible Access Control Markup Language (XACML) [12] and NGAC. Due to the semantic complexity and the lack of user-driven nature of XACML, NGAC has been found to be the more suitable solution for defining Access control rules in the current work. An introduction to the NGAC standard has been presented in the next sub-section and the manner in which it is leveraged for IoT devices has been presented in Section IV.

3) Accounting: Accounting refers to the process of measuring the consumption of resources in a service exchange. Authentication and Authorisation of IoT devices gained a lot of interest in the research community whereas Accounting of service exchanges between the IoT devices is not much explored. This paper presents a simple accounting model within the IoT local cloud.

C. Next Generation Access Control

NGAC is a flexible infrastructure whose purpose is to provide access control services in many different environments. It offers the possibility of developing different types of policies simultaneously. According to [13], such an infrastructure is scalable and remains manageable in the face of changing technology, organizational restructuring and increasing data

volumes. NGAC has been introduced by NIST and approved as a standard by the American National Standards Institute [6] [14].

The NGAC standard expresses policies in terms of attributes [15]. These attributes characterise the users and objects of an access request. NGAC grants or denies access based on the relations defined between the attributes in the policies. The framework comprises of a set of basic elements, containers, and relations. The basic elements are:

- *Users* are unique entities that request access to resources.
- *Objects* are controlled resources.
- *Operations* are actions performed on policy elements.

Containers are terms used to group elements with the same access rights. They can be attributes of users and objects, or policy classes. The possible types of relations between the basic elements and containers are:

- *Assignment* that defines the containment of elements within attributes or policy classes.
- *Association* that defines the access rights.
- *Prohibition* that specifies denial of access.
- *Obligation* that dynamically alters access rights depending on the environmental conditions.

The way NGAC can be used to provide service level access rights to the consuming systems in a local cloud is of interest in this paper. Its implementation is further described in sections IV and V.

III. AAA in an Arrowhead Local Cloud

In this section, all the services (Authentication, Authorisation, and Accounting) of the implemented AAA system has been detailed. The aim of this system is to provide secure and fine-grain access control to the IoT devices in a local cloud. The sequence of interactions between a service consumer, service producer and the AAA system in a local cloud are shown in Figure 1.

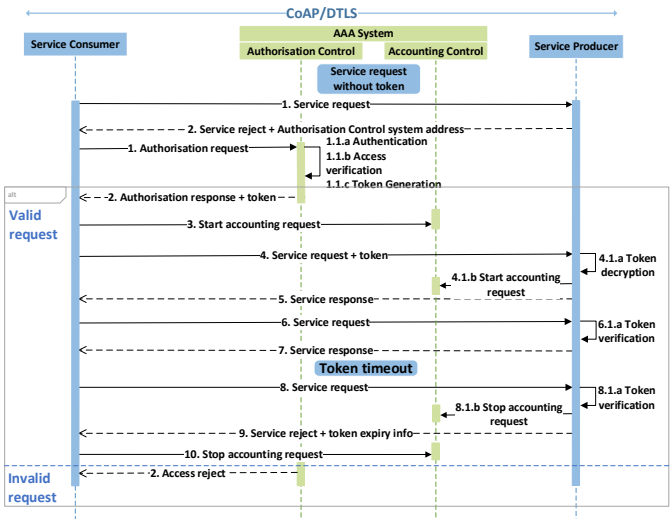

Fig. 1: Authorisation and Accounting process

A. Authentication

Communications between the IoT devices and the AAA system in the current work uses CoAP's Datagram Transport Layer Security (DTLS) [9] for secure communications. The CoAP protocol supports four security modes: NoSec, PreSharedKey, RawPublicKey and Certificate. The nature of authentication achieved using these modes is discussed below:

- NoSec: In this mode, security is not provided by the protocol. Therefore, it cannot be used for authentication.
- PreSharedKey (PSK): In this mode, a key has been shared between the two communicating entities and they authenticate each other based on that key. It is assumed that the key is securely communicated to both parties beforehand. The PSK is not a viable solution for the IoT local clouds as communicating the shared keys to all IoT devices in a local cloud is cumbersome.
- RawPublicKey (RPK): In this mode, the IoT devices possess an asymmetric key pair to establish DTLS connections, but the authentication of communicating parties cannot be verified.
- Certificates: In this mode, X.509 [16] certificates are generally used and are signed by a trusted root (e.g. Certificate Authority (CA)). The devices in a local cloud can authenticate each other if they possess a certificate from the CA specific to that local cloud.

Out of all the DTLS security modes, the certificate mode is perceived as an appropriate mode for achieving authentication in a local cloud context. Thus it has been used in the current work. In a DTLS connection, the peer initiating the connection is usually a client and the other peer is a server. CoAP's DTLS is similar to Transport Layer Security (TLS), except that it uses User Datagram Protocol (UDP). In TLS, a client (browser) verifies the server's certificate in a one-way handshake and a two-way handshake is optional where a server also verifies the client's certificate. Whereas, in the IoT environment, both communicating entities play an important role (e.g. the AAA system should verify that the service consumer is the same party who is requesting access and the service consumer should also ensure that it is requesting access from respective AAA system) and they need to authenticate each other using a two-way handshake.

In X.509 certificate generation, elliptic-curve cryptographic (ECC) keys are preferred to RSA keys as they provide the same cryptographic security but with smaller key sizes. This reduces the size of the certificate and IoT devices can store these certificates in their limited memory and makes communication faster due to the shorter handshake messages. A self-signed CA root certificate has been created in the current work using OpenSSL [17] and Java keytool. It is used to sign the AAA system's certificate and all other entities in the local cloud. All these systems have a truststore and the CA's certificate has been added to it. The DTLS connections are considered authenticated if they possess the certificate issued by the same CA. Additionally, the AAA system verifies the identity of the service consumer using the Common Name

(CN) field (which stores the unique system name of the service consumer in a local cloud) in its certificate.

B. Authorisation

Upon receiving an access request from a service consumer, the Authorisation service verifies the access rights as per the NGAC policies, which is detailed in the next section. If access is granted, the authorisation system issues an authorisation grant token to the service consumer. Upon receiving the token, the service consumer submits a service request to the service producer along with this token as shown in Figure 1. These tokens are JSON Web Encrypted (JWE) tokens that are signed by the AAA system (so that the service producer can verify token authenticity) and have been encrypted with the service producer's public key to keep them secure from any unauthorised access. They are self-contained tokens (i.e. the token itself contains information required for the service exchange so that the service producer does not need to validate the token with the AAA system). A validity period for these tokens can be use-case dependent. The producer decrypts this token upon receiving it from the service consumer, verifies its validity (i.e. it verifies whether this token has been issued to the same consumer or not and also verifies if the token is valid or has expired) and offers its services until the token's expiry. The service exchanges between a service consumer and a service producer during a token's validity period is referred to as a "session" in this work. When the intended service is no longer required, the service consumer can also submit a close service request to the service producer to close the session before token's expiry time.

C. Accounting

The service consumer and producer of a session should account the session-related information from both ends to the Accounting service of the AAA system in the local cloud. A sample set of information recorded during a session is given below:

- Number of service exchanges in request/response communications;
- session time (this can be either the token expiry time set in the AAA system or the time difference between the service consumer's first service request and the close service request in a session);
- maximum and minimum packet size of the request;
- IP address change during a session;
- reason for session termination.

This information is general to any local cloud and use-case specific accounting should be developed for more secure and robust accounting needs. The accounting information can be used to calculate payments to service producers, to measure statistics such as the performance of each device, security analysis, etc. Information such as number of service exchanges facilitates micro payments. Information such as packet size, session time, IP address change and termination cause can be used to determine any security breaches such as a Denial of Service (DoS) attack on constrained devices (service producer), compromise of service consumer, etc., during a session.

IV. Next Generation Access Control

In this work, an NGAC standard-based authorisation of IoT devices has been developed and integrated into the Arrowhead Framework to provide fine-grain access control to IoT devices in a local cloud.

A. NGAC Implementation

The NGAC functional structure is formed by the policy enforcement point (PEP), which intercepts users' access requests and enforces the decision taken by the Policy decision point (PDP). The PDP evaluates and issues authorisation decisions. It reaches a decision based on the elements and relations stored in the policy information point (PIP).

The PEP and PDP are implemented as Java programs integrated into the Arrowhead Framework's AAA system. The PIP has been developed in an SQL database in accordance with the graphical representation of the policies [14]. NGAC uses the policy classes to group and characterise collection of policies. Each element (users, objects, attributes, operations, etc.) is represented by a node in the graph representation of the policy. This is handled through Hibernate ORM to create the relations between the Java classes and the MySQL elements.

This is the first work thus far that authorises IoT devices using the NGAC standard. So, as an initial step, it has been developed by considering one unique policy class formed by users, objects and attributes connected by assignments and associations. The users and objects are connected to their attributes. The association of one attribute with an operation provided by an object ensures the access right to every element in that path. A path is defined as the connection between a user, a user attribute, an operation, an object attribute and an object that creates a chain of elements related to each other and with the same access rights.

B. NGAC in the Arrowhead Framework

For the development of the NGAC standard into the Arrowhead Framework, relations between Arrowhead elements and NGAC elements have been established. The common Arrowhead elements for representing service consumers and producers are "Systems" and the services they consume/produce are termed as "Services" [5]. Service consumer systems and service producer systems are represented in the policy by Users and Objects respectively and are grouped according to attributes. The services are represented by operations. The functional structure of the AAA system in conjunction with the NGAC structure is shown in Figure 2. The PEP (AAA system) receives an access request from the service consumer and forwards it to the PDP. The PDP retrieves the policy information stored on the PIP and makes a decision to grant or deny access to the requested consumer. The PDP communicates its decision to the PEP and PEP sends an access response to the consumer. If access is granted, PEP embeds a token in its response.

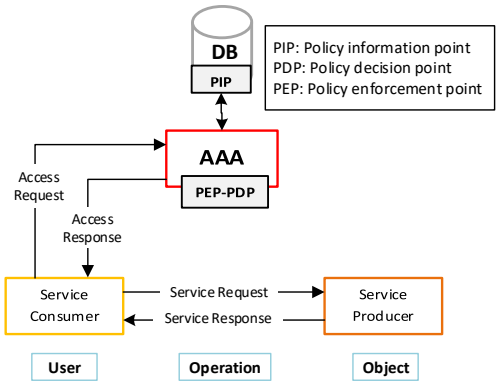

Fig. 2: NGAC functional structure in Arrowhead Framework

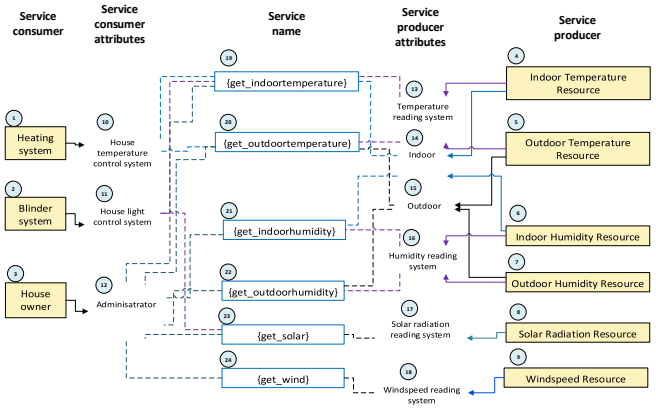

Fig. 3: NGAC policy for the use case

V. Use Case and Scenarios

A district heating use case has been utilised here to evaluate NGAC in an IoT context. The concept of district heating comprises of a smart grid in which heat from a centralised production plant is distributed to an entire district. The smart grid attribute is based on the need to balance production, distribution and consumption. The heating system in the current use case controls the temperature inside the building in accordance with the temperature (indoor/outdoor) measured by a weather station. The window blinds are controlled by a blinder system based on the indoor temperature and solar radiation information measured by the weather station. The main systems that are service consumers in this use case are the *heating system*, *blinder system* and *building owner*. Whereas, the weather station services are regarded as service producers.

In the current use case, a Davis Vantage Pro2 weather station has been used. It consists of two units, one outside and one inside the building. The outside unit comprises an anemometer, solar radiation sensor, temperature sensor, humidity sensor, etc. The inside unit comprises a console equipped with temperature and humidity sensors. These two units communicate wirelessly with each other and the readings of all sensors in these units are transmitted to the console. An application has been developed in Java that acts as a wrapper to the Davis Weather station. It is loaded on a BeagleBone Black device running Debian OS. This application receives readings from the Davis console via UART over USB. It acts as a service producer and responds to CoAP requests from the service consumers (heating system, blinder system etc.). Even though the weather station hosts different services, the service consumers should be restricted to consuming only certain services that are required for their functionality. NGAC is used in this use case to achieve this service level access control. NGAC policy developed for the selected use case has been graphically represented in Figure 3. Access mapping between different systems and diffent services in accordance with Figure 3 is shown in Table I and also explained in the following sub-section.

TABLE I: NGAC POLICY for DISTRICT HEATING USE CASE

Consumer systems	Weather station services					
	get_in-doortem-pera-ture	get_-out-doortem-pera-ture	get_in-doorhu-midity	get_-out-doorhu-midity	get_-solar	get_-wind
Heating system	✔	✔	✘	✘	✘	✘
Blinder system	✔	✘	✘	✘	✔	✘
House owner	✔	✔	✔	✔	✔	✔

A. NGAC policy implemented for the use case

In Figure 3, the systems are represented by yellow boxes: service consumers (users) on the left-hand side and service providers (objects) on the right-hand side. Each system is linked to one or more attributes. These links between service consumers, service producers and their attributes, called "Assignments" in NGAC notation, are depicted with arrows. Services (Operations) are shown in curly braces. The associations, represented by dotted lines, are always formed by joining one service consumer attribute to one service producer attribute. Every element linked in the policy, in the same path, has the same access right.

B. Implementation description

Both the Authorisation and Accounting modules have been implemented in Java. The Accounting system uses MySQL as a backend to store the accounting information. Upon bootstrapping the system, the AAA system reads the access control policies indicated in Figure 3 from a JSON file and populates the database. The Policy Information Point (PIP) is an SQL database with eleven tables containing information about users, objects, attributes, associations, etc. At runtime, the database can be updated or maintained with MySQL Workbench. Local cloud systems are authenticated using certificates

as detailed in Section III. Authorisation is granted based on the access rules in the NGAC database. Here are two scenarios that demonstrate NGAC usage.

In the first scenario, the heating system sends an access request to the AAA system to consume the get_indoortemperature service from the weather station. It does that by sending a CoAP POST request with a JSON payload to the Authorisation Control Service of the AAA system as shown below.

Scenario1

```
—— Authorisation Request——
{
  "AuthRequest":{
  "Consumer":"HeatingSystem",
  "Producer":"IndoorTemperatureResource",
  "ServiceName":"get_indoortemperature"
}}
```

As the heating system has access to the get_indoortemperature service as per the NGAC policies in Table I, an authorisation token is granted as shown below.

```
—— Authorisation Response——
{
  "AuthResponse":{
  "AccessResponse":"true",
  "AccountingID":"475360",
  "Consumer":"HeatingSystem",
  "Producer":"IndoorTemperatureResource",
  "ServiceName":"get_indoortemperature",
  "Token":"eyJjdHkiOiJK..."
}}
```

In the second scenario, the heating system sends an access request to consume the get_outdoorhumidity service. As the heating system attributes do not have association via the get_outdoorhumidity service to the weather station, the authorisation request has been rejected as shown below.

Scenario2

```
—— Authorisation Response——
{
  "AuthResponse":{
  "AccessResponse":"false",
  "Consumer":"HeatingSystem",
  "Producer":"OutdoorHumidityResource",
  "ServiceName":"get_outdoorhumidity"
}}
```

One limitation of the current work is that if the access rules corresponding to the token are changed after a token has been issued, measures to revoke the token are not implemented. To overcome this problem, the authors suggest to use shorter expiry period with the tokens.

VI. Conclusion

Industry 4.0 is incorporating the Internet of Things (IoT) in industrial applications. The use of IoT technologies enables efficient device-to-device (D2D) communication but also creates multiple requirements in terms of security, effective communication, etc. The SOA-based Arrowhead Framework provides features for addressing these requirements through the use of local clouds. One important requirement is security, including authentication, authorisation and accounting (AAA).

This paper has proposed an AAA solution that integrates the NGAC standard using a simple district heating use case that includes devices from multiple stakeholders. The experiments conducted show that NGAC can be used to provide fine-grain access control at service level.

VII. Future work

Future works include adding support for HTTPS, MQTT as well as other necessary protocols. The proposed solution needs to be further tested on resource-constrained devices such as wireless 6LoWPAN-based platform. Full integration with the other core systems of the Arrowhead Framework is also needed. In addition, complete support of the NGAC specification must be implemented.

Acknowledgment

The authors would like to thank the Horizon 2020 projects FAR-EDGE (grant agreement 723094), OPTi (grant agreement 649796) and, in association with the innovation programme ECSEL Joint Undertaking, the Productive 4.0 project (grant agreement 737459).

References

[1] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl, "Industry 4.0–An Introduction in the phenomenon," *IFAC-PapersOnLine*, vol. 49, no. 25, pp. 8–12, 2016.

[2] "Industrial Internet Consortium," the Industrial Internet Consortium is the world's leading organization transforming business and society by accelerating the Industrial Internet of Things (IIoT). [Online]. Available: http://www.iiconsortium.org/

[3] P. Adolphs, "RAMI 4.0 an architectural model for industrie 4.0," Jun. 2015. [Online]. Available: http://www.omg.org/news/meetings/tc/berlin-15/special-events/mfg-presentations/adolphs.pdf

[4] L. Shi-Wan, C. Mark, and M. Stephen, "The Industrial Internet of Things Volume G1: Reference Architecture," 2017, This document is a work product of the Industrial Internet Consortium (IIC) Technology Working Group and its Architecture Task Group.

[5] J. Delsing, *IoT Automation: Arrowhead Framework*. CRC Press, 2017, ISBN: 1498756751.

[6] D. F. Ferraiolo, L. Feldman, and G. A. Witte, "Exploring the next generation of access control methodologies," Tech. Rep., 2016.

[7] A. Rubens, C. Rigney, S. Willens, and W. A. Simpson, "Remote Authentication Dial In User Service (RADIUS)," RFC 2865, Jun. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2865.txt

[8] "Tacacs+ and radius comparison," Jan. 2008. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/13838-10.html

[9] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7252.txt

[10] A. Banks and R. Gupta, "MQTT Version 3.1. 1," *OASIS standard*, vol. 29, 2014.

[11] O. Aafaf, M. Hajar, E. A. Abou, and O. A. Ait, "Access control in The Internet of Things: Big challenges and new opportunities," *Computer Networks*, vol. 112, pp. 237–262, 2017.

[12] E. Rissanen *et al.*, "eXtensible Access Control Markup Language (XACML) version 3.0," *OASIS standard*, vol. 22, 2013.

[13] "Information technology - Next Generation Access Control - Functional Architecture (NGAC-FA)," American National Standards Institute, INCITS 499–2013, March 2013.

[14] D. Ferraiolo, R. Chandramouli, V. Hu, and R. Kuhn, "A comparison of attribute based access control (ABAC) standards for data service applications," *NIST Special Publication*, vol. 800, p. 178, 2016.

[15] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone *et al.*, "Guide to attribute based access control (ABAC) definition and considerations," *NIST special publication*, vol. 800, no. 162, 2013.

[16] R. Housley, W. Ford, W. Polk, and D. Solo, "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," 2008.

[17] The OpenSSL Project, "OpenSSL: The open source toolkit for SSL/TLS," April 2003, www.openssl.org.