Annals of Telecommunications
https://doi.org/10.1007/s12243-019-00709-7

Access control in the Internet of Things: a survey of existing approaches and open research questions

Emmanuel Bertin[1] · Dina Hussein[1] · Cigdem Sengul[2] · Vincent Frey[3]

Received: 11 July 2018 / Accepted: 11 February 2019

Abstract
The Internet of Things operates in a personal-data-rich sector, which makes security and privacy an increasing concern for consumers. Access control is thus a vital issue to ensure trust in the IoT. Several access control models are today available, each of them coming with various features, making them more or less suitable for the IoT. This article provides a comprehensive survey of these different models, focused both on access control models (e.g., DAC, MAC, RBAC, ABAC) and on access control architectures and protocols (e.g., SAML and XACML, OAuth 2.0, ACE, UMA, LMW2M, AllJoyn). The suitability of each model or framework for IoT is discussed. In conclusion, we provide future directions for research on access control for the IoT: scalability, heterogeneity, openness and flexibility, identity of objects, personal data handling, dynamic access control policies, and usable security.

Keywords Access control (AC) · Internet of Things (IoT) · Identity management · Security

1 Introduction

The Internet of Things (IoT) is an enabler for improving many different aspects of private and public life. Its applications range from healthcare to transport, and from environment and energy to business and culture. IoT platforms build on collecting data and often require users to grant permissions to applications, such as the ability to control lights in a house. In addition to monitoring and control, IoT facilitates the rise of a "sharing economy," as IoT objects become a facility that may be used by many individuals.

The IoT offers an enormous market opportunity for equipment manufacturers, Internet service providers, and application developers. Gartner reported that 8.4 billion of connected objects were in use in 2017 (up 31% from 2016) and forecasted 20.4 billion for 2020, so more than 240% of growth over the next three years ([online] https://www.gartner.com/newsroom/id/3598917). Previously, traffic monitoring of US cellular network already showed a year to year increase of 250% for M2M traffic volume [1]. Vertical businesses are the more impacted with a foreseen rise of IoT-based healthcare applications (e.g., mobile health, telecare enable medical wellness, prevention, diagnosis, treatment).

However, the only way to unleash this potential is through a trustful framework for granting access to the things that surround and make up the IoT environment. As it currently stands, consumer trust in IoT systems is effectively not improving, especially, it makes the news that unsecured IoT devices enable new security attacks. These concerns about security and privacy of IoT systems are not new. In 2015, 80% of the 1000 Internet users surveyed did not believe IoT's benefits outweigh any privacy concern about their personal data [2].

Privacy risks of any system are produced by three factors: personal data collected or generated, actions performed on that data, and the context surrounding its collection, generation, processing, disclosure, and retention [3]. Some privacy frameworks, standards, and legislation try to reduce the risk from these factors to address privacy concerns. These include Fair Information Practice Principles (FIPPs) [4], ISO/IEC 29100 Privacy Framework [5], Privacy by Design [6], and, for European residents, General Data Protection Regulation (GDPR) [7]. However, concerning IoT, while most of the regulations mentioned above focus on the first two factors, we believe that the third factor, "context," is critical to ensure

✉ Emmanuel Bertin
emmanuel.bertin@orange.com

[1] Orange Labs, 42 rue des Coutures, 14000 Caen, France

[2] Nominet, 6th Floor, 2 Kingdom Street, London W2 6BD, UK

[3] Orange Labs, 4 rue du Clos Courtel, 35510 Cesson-Sevigne, France

Published online: 02 March 2019

a trustful Internet of Things. IoT systems thus need proper access control to reduce their privacy risks [8, 9].

This paper reviews and discusses the various requirements and solutions to address the access control issue for the IoT. The rest of the article is structured as follows. We first discuss the main principles of access control in Section 2. We present access control as the exchange of and the reasoning on a set of assertions to decide access to a resource in a given context. We then detail the various methods for reasoning on the provided assertions and taking an access control decision (Section 3). In Section 4, we introduce the various protocols, languages, and architectures to build and convey access control related assertions. Many of these works stem from the Web. We finally discuss the future requirements to drive the research on access control for the IoT and a possible research agenda to address these open questions (Section 5).

2 Access control principles for the IoT

In this section, we first discuss the main principles of access control (AC) and various possible architectures when it comes to the IoT.

2.1 Main principles

AC enforces a selective restriction of access to protected resources, including data, IoT objects, and services. In the IoT context, access can mean performing a CRUD operation (create, read, update, delete) on a given data resource, but also performing operations on a physical resource (e.g., actuating). The decision to grant access to a resource is called access control [10]. Note that we need to differentiate between access control and authentication: the fundamental question of authentication is "who is speaking?" while the one of access control is "who is trusted?" (i.e., who is authorized to perform a given action) [11]. In a distributed environment, these questions can receive a variety of answers.

More precisely, Gusmeroli et al. [12] define AC as a method for controlling who (i.e., subject) can perform which access rights (i.e., actions) on which resource (i.e., object). Thus, AC is represented via a set of assertions involving subjects, actions, and objects [12]. In other words, AC rules or policies specify the conditions that must be fulfilled by a subject (e.g., user, service, device) to access an object (e.g., service, device) to perform an action (e.g., read).

The different steps that AC involves are shown in Fig. 1. First, the subject requests an action on the object to the AC system (or the request might be directly targeted to the object and intercepted by the AC system). This action may, for example, be a request to access an object's data or to actuate this object. This requested action comes along with attributes intended to assess the right of the subject to perform this action. Based on these attributes, as shown in Fig. 1, the AC system realizes the authorization process with three following logical steps. First, it selects the applicable rules or policies for this request based on the context of the request and the provided attributes. Second, it compares the provided attributes with those policies and decides to grant access or not. Third, it enforces this decision by transferring the requested action to the object.

These steps are implemented in various ways in the different AC architectures.

2.2 AC in different IoT architectures

From an architectural point of view, two main paradigms exist in the literature for IoT systems: centralized and distributed [13]. In the centralized IoT architecture, a central entity handles acquiring, processing, and transferring data between networked nodes as well as decision-making, including authorization decisions. In the distributed IoT architecture, entities at the edge of the network can exchange data and dynamically collaborate. Roman et al. [14] characterize distributed IoT by two main principles: (i) edge intelligence, which is the ability to delegate decision-making to entities at a lower level, and (ii) collaboration among diverse entities to reach a particular goal.

The variations in IoT network structure, communication patterns, heterogeneity, and potential mobility may dictate either a centralized or distributed AC implementation. Figure 2 shows the typical design patterns for IoT networking [15]. For instance, in Fig. 2b–d, they have centralized entities such as a service provider or a gateway. These entities may authorize access requests centrally. Note that, in this case, the endpoints (e.g., IoT devices) need to be able to connect to the centralized authorization entity at all times. However, in IoT networks connectivity may not always be guaranteed due to mobility or for energy saving. Also, the centralized AC approaches face the issues of having a single point of failure (SPOF) and scalability [16].

On the other hand, for the device-to-device communication patterns shown in Fig. 2a, distributed authorization solutions would be needed. Distributed authorization in the IoT context translates into pushing access control intelligence to the edge of the network [16]. However, resource limitations at the IoT edge are a significant challenge to achieve such edge intelligence. Furthermore, due to the lack of trust in distributed entities, enforcement of access control becomes a crucial security issue.

Depending on the network structure, and the underlying application, it is also possible to consider a semi-distributed AC approach. In such an approach, AC responsibility is split between a centralized decision-making authority and several distributed decision enforcement authorities.

These challenges need careful attention from the IoT research community, and they form the main motivation of this

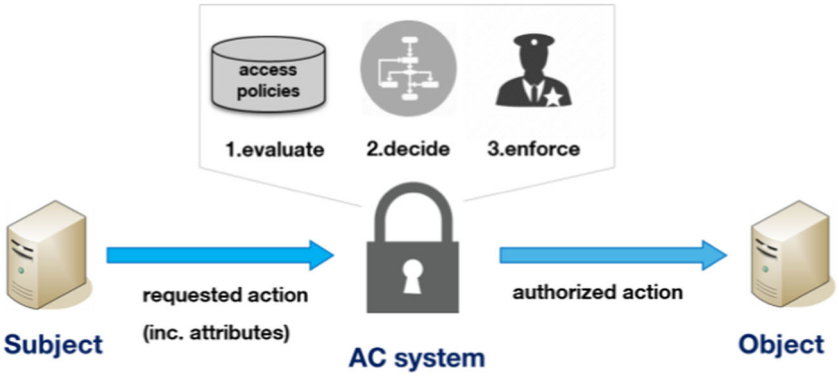

Fig. 1 Overview of an AC system. The subject requests access to perform an action or multiple actions over an access-controlled project. The request is evaluated based on policies. The access control decision—allow or deny access—is enforced by the object

paper. We present in the following section the various access control methods and how they fit with the IoT needs.

3 Access control models for the IoT

In this section, we present current authorization solutions in practice and standardization and discuss them in the context of IoT. These solutions cover a broad range intentionally, to enable an extensive discussion on the various solutions. Therefore, while some of these solutions specifically address IoT scenarios, others also target more general scenarios. We first detail the various AC policies, corresponding to the evaluation and decision steps introduced in the previous section. We then detail AC mechanisms corresponding to the enforcement step.

3.1 Evaluation and decision steps

AC policies can rely on several different models to evaluate an access request and decide whether the request is or not authorized. The most known and widespread models are the following: discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), and attribute-based access control (ABAC). These main AC models can be compared based on the linkage done between subject and object by the policies, as shown in Table 1.

More precisely, in *DAC*, the owner of an object sets access control policies on an object. The AC decision is based on the access rights of subjects, characterized by an identifier, e.g., IP or physical address. These rights are typically represented by an access matrix or access control lists (ACLs) assigned to each object, but DAC can also be implemented using capabilities (cf next section). Identity-based access control (IBAC) is a form of DAC, where the access control decision uses the authenticated identity of the subject.

Because of its simplicity, DAC remains the most used method for real-life IoT deployments, and notably in the case where an IoT Object, identified by its physical address (i.e., media access control address) or by credentials stored within the object, can access to resources situated on a cloud platform. This method has also been used as a basis for research works. For example, [17] proposes to rely on physical addresses to build a virtual software-defined LAN between IoT objects that have to exchange information. As the access

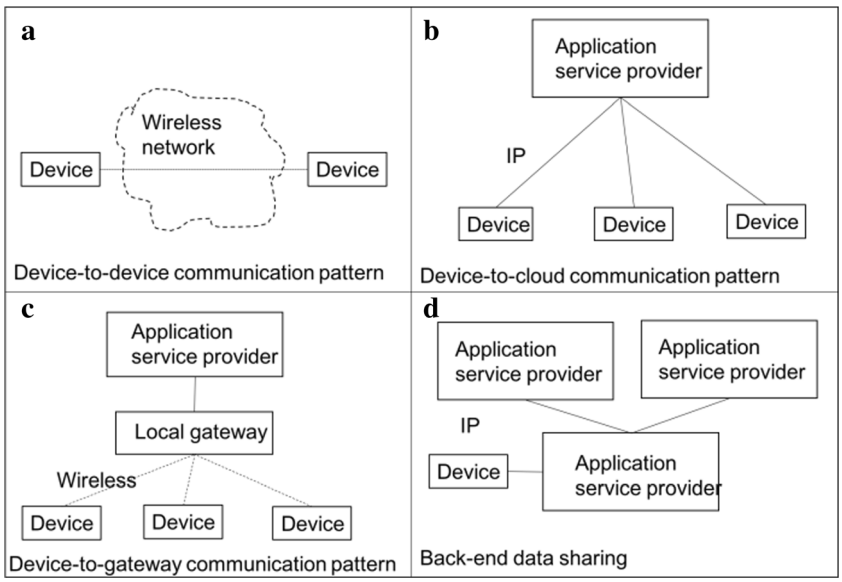

Fig. 2 Design patterns for IoT communication (source: IETF [15]). Four patterns emerge in IoT communications: device-to-device, device-to-cloud, device-to-gateway, and back-end data sharing

Table 1 Policy evaluation criteria of different AC models and applicability to IoT

AC model	Policy evaluation criteria	Applicability to IoT
DAC	Subject's identifier	Extensively used, but does not cover all IoT scenarios, e.g., does not work for use cases with no device identifiers
MAC	Subject's access to a security label	Generally considered as too rigid for IoT scenarios
RBAC	Subject's role	Variations of this model (e.g., smart OrBAC) are used for IoT scenarios
ABAC	Subject's attributes (inc. dynamic ones)	The most suitable for IoT scenarios as it can support flexible attributes

control is performed at the network level, this ensures extensive security, but at the apparent price of flexibility.

MAC is a security policy, where a central authority makes access decisions [18]. MAC restricts access to objects based on the sensitivity of the information they contain, represented by a security label [19]. The formal authorization of subjects (i.e., clearance, formal access approvals, and need-to-know) determines whether a subject can access information of such sensitivity (i.e., with this security label) [20]. A system administrator should perform the necessary labeling and clearance processes.

MAC is suitable for centralized environments with rigid access control policies, and where it is possible to distinguish subjects and objects with the necessary security clearance. Therefore, simple MAC-based approaches are too constraining for IoT environments, and no significant results have been achieved for adapting the MAC approach to IoT.

RBAC groups permissions into roles. Groups of permissions can then be provided to users with the simple operation of assigning roles [21]. A limited number of roles can represent many users or user types, and non-expert personnel can assign these roles to users. Therefore, it becomes easier to audit which users have specific permissions and what permissions have been granted to a given user. However, roles must be engineered before RBAC can be used [22]. Furthermore, RBAC must be constrained to handle dynamically changing attributes, such as time of day and location, as core RBAC cannot handle such attributes. Organization-based access control (OrBAC) [23] aims at simplifying the expression of security policies, by introducing a higher level of abstraction than RBAC. It allows modeling an AC policy independently of its contextual implementation within an organization.

Many works adapt the RBAC model to IoT [24–27]. However, the use of resource-constrained devices is rarely addressed. In [28], the authors focus on this issue and propose an implementation of the OrBAC model named SmartOrBAC. For this purpose, SmartOrBac introduces different functional layers:

- A constrained layer that groups the subjects and the objects (as defined in Fig. 1)
- A less constrained layer that contains the AC system
- An organization layer that is responsible for allocating roles and privileges

Also, SmartOrBAC introduces the concept of context for policy evaluation.

ABAC grants or denies requests based on subject and resource attributes, environmental conditions, and a set of policies specified concerning those attributes and conditions [29]. With ABAC, dynamically changing attributes, such as time of day and location, can be accommodated in access control decisions. There is no need to engineer roles unless role names are used as attributes. Essentially, ABAC is capable of enforcing DAC, MAC, and RBAC. For instance, DAC is ABAC with the "identity" attribute and RBAC has "role" attribute. However, in ABAC, a potentially large number of attributes must be managed, and expert personnel must select the attributes.

Due to its flexibility, the ABAC model has been widely used for implementing access control for IoT [30–33]. Indeed, relying on identities for authenticating a subject is complex in IoT scenarios where device identities are hard to maintain and assert. Instead, in distributed environments, using a combination of several attributes for authenticating a subject may be more reliable. For instance, an IoT device has a manufacturer model number, a software product number, and an IP or physical media access control address. Thus, identity may become a mere parameter in a list of attributes needed to evaluate an access request [34].

Also, this model has also been used for domain specific access control, for example, in [35], where an ABAC model is used for access control to a personal healthcare monitoring service. Attributes are here allocated to subjects as well as to objects (as defined in Fig. 1). Subjects' attributes mainly consist of the identity and the function of the subject, while objects' attributes define the type of medical record (public information, physical or mental diseases). Rules are then built to correlate subjects' and objects' attributes for defining access rights.

3.2 Enforcement step

Two main concepts enable the enforcement of the policy models: access control matrix and capabilities. These two different mechanisms can be compared based on the way AC policies are enforced, as shown in Table 2 below.

Table 2 Policy enforcement criteria for different AC mechanisms and applicability to IoT

AC mechanisms	Policy enforcement criteria	Applicability to IoT
AC matrix based	Mapping with a valid matrix entry	Not suitable for decentralized IoT architecture. More suitable when access policies do not need to be updated often.
Capability based	Presenting a valid capability token	Easier to decentralize. Tokens provide a more dynamic view of the access rights.

Access control matrix specifies the rights each subject possesses for each object. More precisely, this matrix specifies the linkages between three kinds of entities: (a) protected objects, (b) subjects, and (c) access rights (which specify the operations subjects can perform on objects). The access matrix [36, 37] provides a useful framework for describing resource protection in any system. In its typical basic implementation, an object is associated with an ACL (access control list) storing all subjects that can access it and its access rights.

This mechanism is easy to implement, especially in the case of a centralized approach where an access control system stores and enforces the access matrix. However, decentralized implementations have also been proposed for the IoT [38].

Capabilities are either a token, ticket, or key that grants to its owner the permission to access an entity or an object in a computer system [39]. The capability can be thought of as a pair (*object*, *actions*) stored by the subjects and presented with each access request. Capabilities need to be unforgeable, and the possession of a capability authorizes a subject for the *actions* on the *object*. With this mechanism, the access control system can be more easily distributed. The potential for a distributed implementation has raised a lot of interest in the IoT community [40–43].

For example, the European FP7 IoT@Work project has developed a capability-based access control mechanism (CapBAC) for IoT [44]. Figure 3 presents an example use case. When Bob goes on a business trip for a long time, he asks Dave to house sit. However, Bob wants to limit Dave's access to only housekeeping. At this time, Bob has issued a token with only the housekeeping permission, and the token can be used only during the period specified in the token.

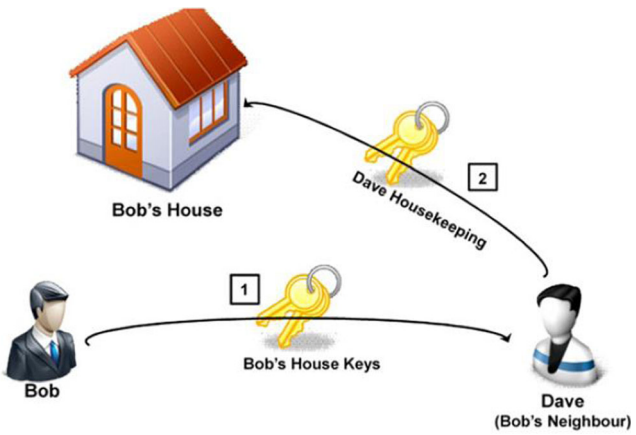

Fig. 3 CapBAC home keeping use case (source: [12]). Bob grants access to his neighbor Dave for housekeeping. The capability tokens allow Dave to enter Bob's house only in designated hours with access limited to housekeeping functions (e.g., Dave cannot change settings for house alarms etc.)

Moreover, the aforementioned AC models and mechanisms that cover the evaluation/decision and the enforcement step are not independent one from another. Table 3 synthesizes the usual mappings between both.

However, besides these AC models and mechanisms, new general-purpose identity management protocols and frameworks have emerged from the Web to implement end-to-end AC architectures.

4 Identity management and access control architectures: from Web standards to IoT

While identity management is treated as a separate concern from access control in literature, it acts as a pillar that supports authorized access.

Early approaches in the Web relied on site-specific identity management schemes where user identities were asserted via site-specific usernames and passwords. The vast proliferation of Web services, as well as the need to share and manage protected resources across several Web services, demanded building authentication mechanisms across security domains [45]. Federated identity management (FIdM) and single sign-on/single log-out (SSO/SLO) were introduced to address those problems [46].

In IoT, the challenge is not to access to one single physical object at a time but also to a mashup and composition of devices, services, and Web applications to accomplish a particular goal [47], in a context of complex relationships between objects and services as well as between owners and users.

Table 3 Mapping between AC models and mechanisms

AC mechanisms\models	DAC	MAC	RBAC	ABAC
AC matrix	x	x	x	x
Capability				x

The adaptation of Web identity management and access control mechanisms for IoT have focused either on the SOAP-based Security Assertion Markup Language (SAML) and eXtensible Access Control Markup Language (XACML) [48], on the RESTful-based OAuth 2.0 [49], or on the User-Managed Access (UMA) [50].

Table 4 provides an introduction to these mechanisms along with their applicability to IoT. These results are then detailed in Sections 4.1 to 4.6 below.

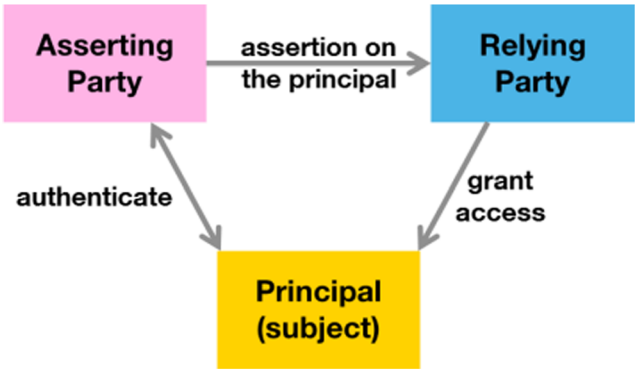

Fig. 4 SAML main components include the asserting party, the relying party, and the principal. The asserting party can authenticate the principal to the relying party

4.1 SAML and XACML

SAML is an OASIS open standard for exchanging authentication and authorization data between parties. One of its popular usages is for identity federation. Using SAML, clients need only to assert their identities once in a federated environment before accessing services across security domains [48].

SAML is a markup language as well as a communication mechanism. SAML has four XML-based mechanisms: security assertions, protocols, bindings, and profiles [51–53]. Figure 4 shows the main entities of SAML and relationships between these entities. An asserting party (AP) is the administrative domain that hosts one or more SAML authorities that issue assertions. A relying party (RP), which, as its name suggests, relies on AP for receiving assertions about a subject. The principal is an entity whose identity can be authenticated. In the single sign-on (SSO) profile, the AP is an identity provider (IdP), which manages identity information about principals and provides principal authentication. The relying party is a service provider (SP). It is assumed that there is a trust relationship between RP and the AP.

SAML assertions relay security information about a particular subject or entities. There are three kinds of assertions: authentication, attribute, and authorization decision. Assertions are secured using digital signatures and encryption.

SAML defines a number of request/response protocols. Protocols specify how SAML elements (encompassing assertions) are requested and received. With specific bindings, SAML protocol messages can be embedded and transported over protocols such as SOAP.

A SAML profile describes how assertions, protocols, and bindings combine for specific business use cases. For security, PKI (public key infrastructure) and TLS are recommended. Mutual authentication and digital signatures are also recommended measures.

Table 4 Various AC architectures and applicability to IoT

Architectural framework	Definition and main use case	Applicability to IoT
SAML	OASIS open standard for exchanging authentication and authorization data. Enables identity federation (i.e., single sign-on)	Too verbose and heavy-weight for most IoT scenarios.
XACML	OASIS standard which implements ABAC (attribute-based access control). Enables sophisticated access control policies.	Too verbose and heavy-weight for most IoT scenarios. Still used as an architectural pattern.
OAuth 2.0 framework	Enables a third-party to get limited and controlled access to a resource on behalf of the user. Widely used for web authorization.	While traditional OAuth 2.0 authorization flows are not very suitable for IoT, the OAuth 2.0 device flow may apply to IoT use cases.
ACE	Extends OAuth 2.0 to constrained environments. IoT scenarios are the main driver for the work.	This authorization solution is specifically developed to address various IoT use cases.
UMA	Introduces a new OAuth 2.0 grant, so that a resource owner can grant access to a different requesting party. Designed for web authorization.	While UMA is designed for web authorization, it may be adapted for some IoT use cases.
LMW2M	Provides identity-based access control for machine-to-machine communication. Targets machine-to-machine communication environments.	Presents a straightforward solution, but its ability to handle dynamic IoT environments is limited.
AllJoyn	Provides ACL-based access control. Applicable to device-to-device and device-to-gateway communication environments.	Offer some flexibility in how many users control the same devices, but with increased complexity.

The initial purpose of SAML was to design a generic language for security assertions. However, SAML profiles have been practically oriented towards authentication and SSO use cases, so authorization of entities and protocols remained out of the scope of SAML standardization process. This was the main driver for starting the XACML standardization.

XACML [54] is an OASIS standard which implements both ABAC model (for policy evaluation) and access control matrix (for policy enforcement). It offers a language as well as a reference architecture, which includes functions such as Policy Decision Points (PDPs), Policy Enforcement Points (PEPs), Policy Administration Points (PAPs), and Policy Information Points (PIPs). As seen in Fig. 5, in an XACML system, a request to, for instance, a file system or a web server, is made to the PEP protecting the resource. The PEP forms an XACML request based on the requester's attributes, the resource in question, the action, and other information about the request. The PEP then sends this request to a PDP, which returns an answer to the PEP whether access should be granted based on policies that apply to the request. PEP respects this answer in allowing or denying access to the requester.

XACML enables sophisticated policies that are formed by rules, obligations, and advice. In XACML, attributes (which are characteristics of the subject, resource, action, or environment in which the access request is made) form the basis of policies. For example, a user's name, their group membership, a file they want to access, and the time of day are all attribute values. The access request sent from PEP to PDP is also formed almost exclusively of attributes, which are then compared to the attribute values in a policy to make the access decisions. The rule and policy combining algorithms define a procedure to reach an authorization decision

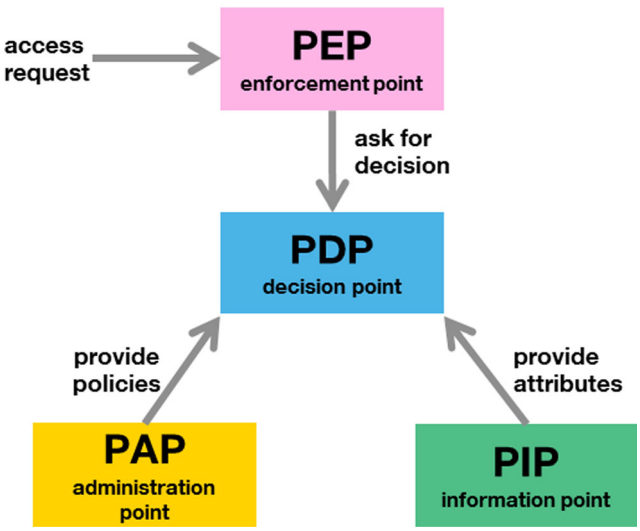

Fig. 5 XACML architecture. Access request is made to PEP, which is relayed to PDP for decision-making. PDP relies on the information from both PAP and PIP to grant or deny access to the requested resource

given the evaluation results of a set of rules and policies. However, as a price of such expressiveness, XACML is quite complex.

Various research works have proposed building an authorization framework for IoT devices upon SAML and XACML [55, 56]. At the time of that research (mainly from 2010 to 2013), SAML and XACML were indeed the most popular Internet and access control standards used in industry.

Of significant interest, the authorization framework for the Internet of Things proposed by Ludwig Seitz et al. [56] allows fine-grained and flexible access control to connected devices with insufficient processing power and memory. To that end, authors have designed profiles and adaptations of XACML and SAML to enable or optimize their use with constrained devices. Their work includes a complete reference architecture and some optimizations to adapt the protocols to constrained devices. As an example, one optimization addresses the verbosity of the full syntax of XACML responses and SAML assertions by defining a lightweight JSON-based notation for an authorization assertion.

Similarly, more recent work focuses on defining a self-understandable JSON authorization assertion [57] computable by a constrained device and compliant with an existing authorization protocol. Both approaches rely on IETF JSON Web Encryption (JWE) mechanisms [58]. However, the results of both research conclude that JWE is adding too much overhead for payloads of only a few bytes, which are common in constrained IoT protocols such as CoAP.

4.2 OAuth

The OAuth 1.0 protocol [59], and its successor, the OAuth 2.0 framework (RFC 6749) [49], and the OpenID Connect framework ([online] http://openid.net/specs/) are designed to solve authorization delegation issues in large-scale Internet applications, with the original design coming from Facebook. OAuth enables a third-party to get limited and controlled access to a resource on behalf of the user. This type of authorization facilitates use cases where a person (e.g., Jane Doe) grants an authorization to access her personal data (e.g., photos stored at a drive) to another service (e.g., print service) that is also used by the same person.

More precisely, as shown in Fig. 6, OAuth defines four different roles:

- The resource owner (RO) is the party capable of granting access to a protected resource. In the case of a person, the resource owner is the user.
- The resource server (RS) hosts the protected resources and controls the access to these resources.
- The client (C) is an application making a request to a protected resource on behalf of the RO or its behalf.

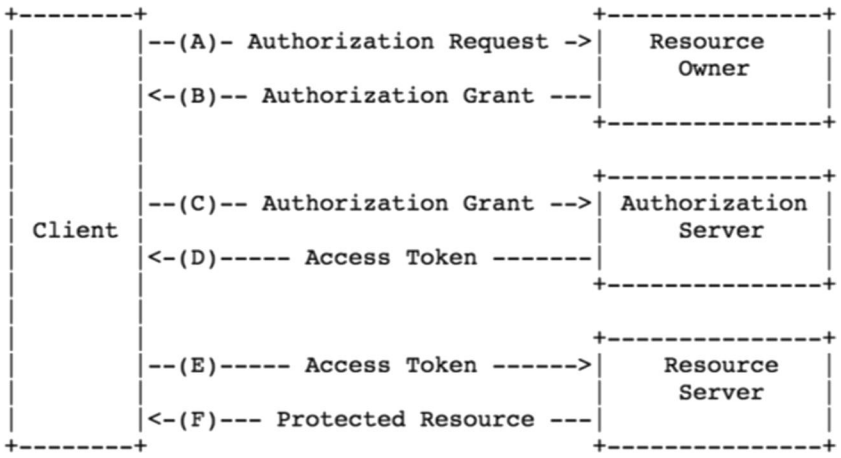

Fig. 6 The abstract OAuth 2.0 authorization flow (source: [49]). The client (C) may make a direct authorization request to the resource owner (RO) and receive an Authorization Grant, which represents the RO's authorization. Next, C requests an access token by authenticating with the authorization server (AS) and presenting the authorization grant. AS authenticates C, validates the grant, and if valid, issues an access token. C uses this access token to request the protected resource from the resource server (RS)

- The authorization server (AS) issues clients access tokens to the protected resource after having authenticated the RO and having obtained its authorization.

An access token is a string denoting a specific scope, lifetime, and other access attributes. The token represents the subject's capability and can also be interpreted as a summary of the RO policy on the resource. The client uses an access token to get access to protected resources on the resource server [60]. The resource server processes the access token and allows or denies access according to its contents.

OAuth can, therefore, be considered as implementing a DAC model for policy evaluation (as the access is based on the consent of a validly identified RO) and a capability-based mechanism for policy enforcement (access token).

OAuth 2.0 is designed for HTTP and may apply to web-based IoT solutions. For instance, in a scenario, as shown in Fig. 2d, where Alice grants access to a mobile health app to use data coming from her wearable and stored in another application.

However, Web services rely mainly on user-owned credentials (e.g., username and password, hardware token, or mobile phone) for asserting a user identity. These types of credentials may not exist in the IoT world where other sorts of identities for physical objects should be considered [61]. Moreover, provisioning user credentials to objects should be considered as a major security flaw, even though commonly done in IoT implementations.

For IoT applications, ongoing IETF work on OAuth 2.0's device flow for browserless and input constrained devices [62] provides a solution for clients running on devices that do not have a straightforward data entry method. Figure 7 depicts this device authorization flow. When the device requests access from the authorization server, the authorization server returns a device code, a verification code, and a verification URI to the user. Visiting the verification URI, the user can grant access to the device by validating the verification code. Meanwhile, the device continually polls the authorization server to check whether its verification code has been validated. In the worst case, the device polls the authorization server until its device code expires. In the case of success, the authorization server returns an access token, which gives access to the protected resource.

This is quite a pragmatic approach, and its main drawback is its purely centralized model.

4.3 ACE

In addition, the IETF Authentication and Authorization in Constrained Environments (ACE) workgroup is working towards extending OAuth 2.0 to provide a solution for IoT. The work considers the very different and limited capabilities of IoT devices concerning the available processing and message exchange, in supporting different authorization use cases [63]. The ACE group builds on:

- CoAP for messaging (without excluding the possibility of using other protocols such as MQTT or QUIC)
- Concise Binary Object Representation (CBOR) and CBOR Web Token (CWT) [64] for token representation
- CBOR Object Signing and Encryption (COSE) and DTLS, for application and transport layer security, respectively

Interestingly, CWT solves the issue of the heaviness of JWE for constrained devices, i.e., objects of IoT.

ACE also uses a different token type, PoP (proof of possession) tokens compared to OAuth 2.0 device flow. PoP tokens are access tokens with a PoP key associated with the

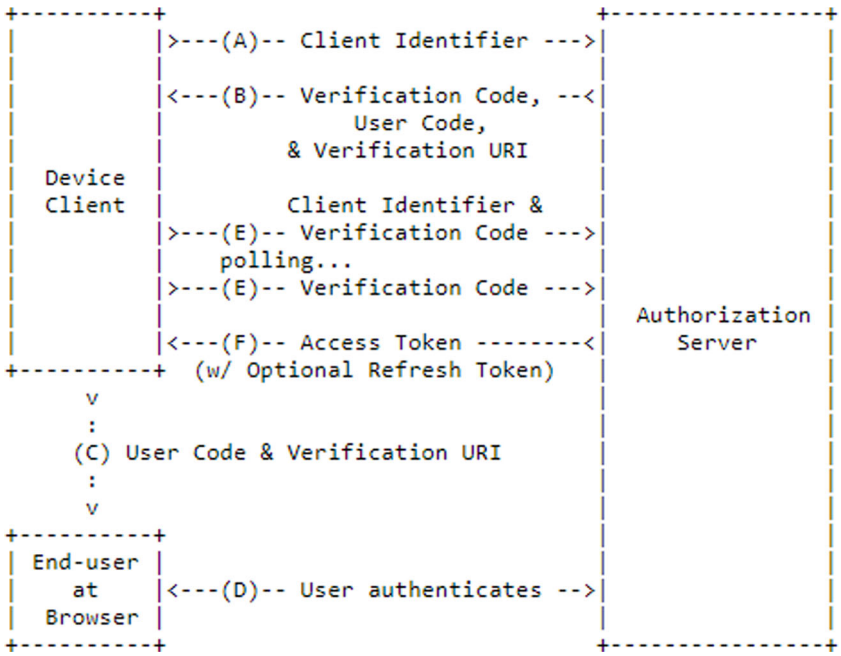

Fig. 7 OAuth 2.0 device flow (source: [62]). The client request to the authorization server (AS) includes a client identifier. The AS issues a verification code, user code, and verification URI. The client instructs to user to visit the URI (elsewhere, not on the device) and use the user code to grant access. The AS authenticates the end-user, and the user grants access by providing the user code. Meanwhile, the device polls the AS with its verification code to check whether the authorization process has completed. If successful, the AS responds with an access token

token. A PoP token allows a client to prove to the RS that it is indeed the intended authorized owner of the token and not merely the bearer of the token.

Assuming that the RS and client have registered with the AS, the following sequence of events needs to occur in ACE for the client to access a resource hosted by the RS. First, the client makes a request to the AS. The AS evaluates the client token request and grants or denies access by either returning a PoP token or an error [65].

The client includes the PoP token in its resource request. Using the PoP token, RS authenticates the client using the PoP token. RS also may locally evaluate the token if it has the capability. Otherwise, it may contact the AS to validate the token. This process is called token introspection. Depending on the token result, the RS may grant or deny access to the resource [65].

ACE handles the different communication patterns shown in Fig. 2, including some device-to-device communication support. Essentially, ACE expects that devices may frequently be offline, or they may not support IP-based communication, and therefore, they may not always be able to communicate with the AS [63]. Then, a more capable client authorization server can request tokens from the AS.

If a device is acting as an RS but does not have continuous Internet connectivity, then it may need to verify tokens locally. However, if the user policies have changed and the token is not valid anymore, there is no simple way to revoke the token. This issue may harm the entire security of the system, underlining the difficulty of providing meaningful authorization in limited and distributed IoT systems.

4.4 UMA

UMA working group has developed UMA (User-Managed Access) under the Kantara Initiative [50]. UMA defines a new authorization grant which enables resource owners to manage access to their resources by clients operated by arbitrary requesting parties (e.g., Alice authorizing access to her resources by Bob's services, compared to OAuth2 model where Alice may only authorize access to her resources by her services). It implements capability-based policy enforcement, while the policy evaluation phase remains out of its scope.

Here, the resources may reside on any number of resource servers, and a centralized authorization server governs access to the resources based on user-defined policies. Compared to OAuth 2.0 flows, these access policies are used to handle asynchronous authorization grants [16]. A typical example is the following: a web user (an end-user resource owner) can authorize a web or a native app (a client) to gain one-time or ongoing access to a protected resource containing his home address stored at a personal data store (a resource server). To do this, the user instructs the resource server to respect access entitlements issued by the authorization server.

Figure 8 depicts the UMA flowchart. The resource owner manages online resources at the resource server. To protect these resources, the resource owner introduces the resource server to the authorization server. The resource server, then, registers resources to be protected with the authorization server. Also, out of band, the resource owner configures the authorization server with policies associated with these registered resources.

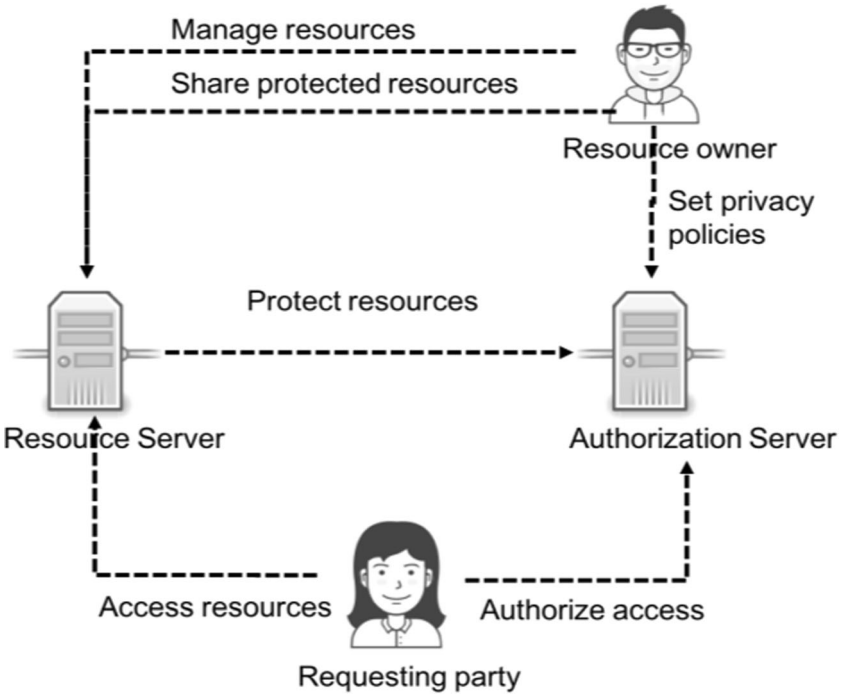

Fig. 8 UMA (User-Managed Access) flowchart (source: [50]). The resource owner (RO) manages its resources in a resource server (RS). RO introduces RS to the authorization server (AS), as the entity for the RS to register protected resources. RO also sets policies at the AS. The requesting party approaches the RS to access the resources. RS registers the access attempts without a token as permission requests at the AS. The client next approaches AS, which uses the permission request and the RO policies to issue a Requesting Party Token (RPT) to the client

To get authorization, the client issues a request for a resource at the resource server. In case of an unauthorized access attempt, the resource server registers a permission with the authorization server and returns a permission ticket to the client. To trigger the authorization process, the client must present this permission ticket to the authorization server. If successful, the authorization server returns a requesting party token (RPT) to the client. To access the protected resource, the client presents this RPT to the resource server. After the resource server verifies the token, the client gains access to the protected resource.

UMA is a centralized solution allowing complex and attribute-based policies. The solution is appropriate for back-end data sharing, as shown in Fig. 2. UMA has several IoT case studies that focus on healthcare scenarios [66].

4.5 Lightweight machine-to-machine protocol

Lightweight machine-to-machine protocol (LWM2M) is developed by OMA (Open Mobile Alliance) and provides identity-based access control for machine-to-machine communication [67]. LWM2M is principally a device management solution but includes authorization, authentication, and channel security protocols. It implements the DAC model for policy evaluation and access matrix mechanism for policy enforcement.

LWM2M follows a gateway-based communication pattern, shown in Fig. 2b, c. The LWM2M server is typically located in a private or public data center, and the LWM2M client resides on the device. M2M applications may contain multiple servers and clients. Also, LWM2M clients may become LWM2M servers in different communication set-ups, which may enable distributed scenarios.

LWM2M defines objects as collections of resources. A resource is an atomic piece that can be read, written, and executed. For instance, a location object may have multiple resources, including latitude, longitude, altitude, uncertainty, velocity, and timestamp. Each resource can have multiple instances.

LWM2M uses access control list object instances that contain ACLs. These ACLs define which operations are allowed on a given object instance for which LWM2M server(s). Bootstrap servers distribute ACLs to LWM2M clients. Predefined ACLs work well when the future interactions with objects are known. LWM2M also provides some dynamicity to authorization decisions by provisioning ACLs real-time from the LWM2M bootstrap server. This capability may allow for more complex authorization policies.

4.6 AllJoyn

AllJoyn is an open-source software framework, aiming to facilitate discovery and communication between devices and applications ([online] https://openconnectivity.org/developer/reference-implementation/alljoyn). It is developed by the AllSeen Alliance, which merged with the Open Connectivity Foundation in 2016. Similar to LWM2M, AllJoyn has ACL-based security features worth discussing in this section.

AllJoyn caters to the device-to-device communication pattern and optionally, supports the device-to-gateway communication pattern (Fig. 2a, b). To enable peer-to-peer communication, AllJoyn implements a “distributed software bus”

that enables AllJoyn devices to advertise and share their abilities with other devices around them [68].

The Security 2.0 feature of AllJoyn allows an application to validate access to interfaces or objects based on the ACLs installed by the owner. In the AllJoyn framework, the security manager plays a similar role to the authorization server in OAuth-based systems. It is the entity that provides certificates, and maintains ACLs, and enables managing security groups, identities, and keys [69]. The owners (i.e., "administrators") define the ACLs. They can also create security groups which allow multiple users to control the same devices. The complexity of this solution increases with how fine-grained the desired access control is. More groups need to be created for more levels of access.

5 Open problems for research

While there is much work on access control, IoT-specific solutions are still in their infancy. In this section, we discuss future directions for research needed to address the unique challenges of IoT.

5.1 Scalability

IoT systems proliferate. Hence, an IoT system must be able to handle a large number of users, applications, and policy enforcement and decision points. An effective way of dealing with scale is decentralization of management and delegation of authority. Thus, an access control management system for large networks must be able to adapt to different management structures (web of trust, hierarchical management).

However, the majority of the works covered in this paper rely on a centralized entity for AC, which may include administering policies, handing out and introspecting credentials for authorization decisions (e.g., the AS in case of OAuth 2.0, UMA, and ACE or the centralized PDP/PEP in the case of XACML). This centrality raises concerns regarding the scalability and single point of failure (SPOF) issues. In contrast, a semi-distributed AC architecture delegates some of the responsibility to the edge: while a central authority may still provide a certificate or an access token to access a device or group of devices, these are enforced locally at the point of access (e.g., ACE and UMA provide these options). Concerning fully distributed approaches, a significant challenge lies in the resource limitations at the edge in IoT systems.

5.2 Managing heterogeneity

New AC protocols get deployed without necessarily deprecating old ones. Daily objects have a longer life cycle than tech objects, extending the duration where backward compatibility will be required from new objects.

Moreover, the same IoT networks may continue to be used in increasingly more complicated ways (e.g., from Intranet to Internet of Things [70]). AC mechanisms should thus be flexible enough to adapt to unplanned cases.

In addition, cross-domain resource sharing and collaborations have become pervasive in today's service-oriented organizations. However, cross-domain access control remains a research challenge for the IoT, with a lack of concrete implementation mechanisms to provide a sufficiently flexible framework.

5.3 Openness and flexibility

A general requirement for IoT is to support the integration of solutions from different industry players and third parties. For an example of an integrated infrastructure, consider the novel home automation solution "Celiane with Netatmo" [71] developed by Legrand along with Netatmo. This solution allows the control of a group of connected objects at home from a common controller. Additionally, users can personalize event-based scenarios. For example, turning on the TV may set off a dim light in the living room or the opening of the front door may control that of rolling shutters. In this example, the IoT architecture design should be an open framework to accommodate compatible components. Such openness is also necessary to achieve AC in IoT, enabling cross-application authorizations for services providers. This could also present an excellent opportunity for third-party developers to integrate their solutions into an open AC architecture. In the enterprise context, this openness will ease the integration of IoT devices in the corporate business processes [72].

5.4 Resolving object identities

A crucial part of proposed systems involves asserting the identities of access requesters, subjects. However, a pure ID-based approach may not be entirely suited for IoT. This is because the identity information may be hard to assert or may not be known or a device may not hold any identity.

Issues for establishing device identity are making attribute-based access control more interesting for IoT. This way a combination of attributes for asserting the authenticity of the requester, e.g., current location, owner, or manufacturer, may be used. For instance, an IoT device may have a model number, or a product key, and an IP, or physical address. These, when used in combination, may identify the device more reliably than a single attribute. UMAv2.0 with its claims-based approach allows this type of subject authentication.

However, identifying the user behind the device is still challenging. Today, there is little clarity as to who owns or should own an IoT device and its data [8]. Moreover, IoT

devices will generally not be single-user devices. A device will be controlled by a group of users, and administrators, with different, and potentially, competing claims over its data.

5.5 Managing personal data

End users should be put in charge of their personal data, whether or not it is machine-generated. IoT systems should be able to manage conflicting interests. Especially, when using the location information from devices, it must be remembered that location is also personal data. For consent-based personal data collection, the EU GDPR [7] requires consent to be as easy to withdraw as to give it, where consent is considered as valid only if the user has sufficient knowledge of the risks and benefits of disclosing information to make a reasonable evaluation [73]. Enabling dynamic consent may be a challenge in IoT-based systems where devices are embedded to the user's environment and interactions with an IoT system may be implicit by design.

5.6 Providing dynamic AC policies

A privacy policy needs to point out precisely who interacts with what data, when, where, how, and to what end. This may conflict with the usability of these systems. The aim should be here to build easy-to-understand policies, which is challenged with the increased combination of options for many data flows in an IoT context. Pointing out all possible interactions appears challenging at best, and detrimental to understanding at worst.

IoT systems, therefore, will need to be configured dynamically to provide the necessary middle ground between expressivity and simplicity by constraining initial policies to a small set of rules. Although there are not many examples in literature, an inspiring example of a dynamic policy is presented in [74] for RFID systems. The proposed system uses physical access control rules to ensure that (1) locations of users are revealed to one another if only they are co-located and (2) the location of an RFID object is revealed to its owner only when the object and its owner are co-located. These rules are relaxed by enabling scenario-specific rules. For instance, in a scenario, where the RFID object is borrowed by a user, the owner of the object may check whether the RFID object is carried by this user. Then, the system returns the information that "user x carries the object" without disclosing the location of user x. While this research constitutes a step in the right direction, IoT systems need "just-in-time" policies that resolve access control requests to a multiplicity of IoT objects.

5.7 Usable security

It is well-known that IoT has several security challenges. Securing the access to and usage of a vast number of vulnerable, connected objects are understudied [75]. Hence, to some, it was no surprise, when a botnet of thousands of connected cameras was used in a denial of service (DDoS) attack on the French host OVH in September 2016 [76].

Current security solutions apply cryptographic methods and disclose keys only to authorized users to protect sensitive IoT data against cyber-attacks. However, these solutions are susceptible to many types of cyber-attacks. They also do not prevent authorized entities from performing certain actions on the connected objects in question. Additionally, the cryptographic methods introduce a heavy computation overhead on the device as well as issues with key distribution and management. Automated and self-contained AC is therefore required in IoT, as recently exemplified in [77]. Such AC should reduce the computational load on IoT devices and usability issues such as password fatigue for the device owners.

6 Conclusions

Compared to other domains, IoT introduces specific challenges concerning access control. While in this paper, we surveyed several outstanding contributions for implementing AC in IoT, much work still remains to be done. None of the proposals presented in this paper is indeed able to cope with all the numerous challenges raised by the IoT. New research programs on innovative AC mechanisms are therefore required to provide a set of suitable solutions for deploying IoT services at a global scale.

References

1. Shang C, Zhou MC, Chen C (Mar. 2014) Cellphone data and applications. Int J Intell Control Syst 19(1):35–45
2. TRUSTe (2015) Majority of consumers want to own the personal data collected from their smart devices, availabile online: https://www.trustarc.com/blog/2015/01/05/majorityconsumers-want-own-personal-data-survey/. Accessed 28 Feb 2019
3. Brooks S, Garcia M, Lefkovitz N, Lightman S, Nadeau E (2017) "An introduction to privacy engineering and risk management in federal systems," National Institute of Standards and Technology Internal Report 8062
4. Hugo Teufel III CPO (2008) Fair Information Practice Principles: framework for privacy, Privacy Policy Guidance Memorandum
5. International Organization for Standardization (ISO), ISO/IEC 29100 (2011) Information technology, security techniques, privacy framework, Dec. 2011
6. Cavoukian A (2012) Privacy by Design and the emerging personal data ecosystem, accessible online: https://www.ipc.on.ca/wp-content/uploads/Resources/pbd-pde.pdf. Accessed 28 Feb 2019
7. General Data Protection Regulation (GDPR) (2018) General data protection regulation (GDPR) [online]. Available at: https://gdpr-info.eu/. Accessed 28 Feb 2019

8. Cerf VG (2015) Access control and the Internet of Things. IEEE Internet Comput 19(5):96
9. Sicari S, Rizzardi A, Grieco LA, Coen-Porisini A (2015) Security privacy and trust in Internet of Things: the road ahead. Comput Netw 76:146–164
10. Fagin R (1978) On an authorization mechanism. ACM Trans Database Syst 3, 3(September 1978):310–319
11. Abadi M, Burrows M, Lampson B, Gordon P (1993) A calculus for access control in distributed systems. ACM Trans Program Lang Syst 15, 4(September 1993):706–734
12. Gusmeroli S, Piccione S, Rotondi D (2013) A capability-based security approach to manage access control in the Internet of Things. Math Comput Model 58(5):1189–1205
13. Cirani S, Davoli L, Ferrari G, Léone R, Medagliani P, Picone M, Veltri L (2014) A scalable and self-configuring architecture for service discovery in the Internet of Things. IEEE Internet Things J 1(5):508–521
14. Roman R, Zhou J, Lopez J (2013) On the features and challenges of security and privacy in distributed Internet of Things. Comput Netw 57(10):2266–2279
15. Tschofenig H, Arkko J, Thaler D, McPherson DR (2015) Architectural considerations in smart object networking. In: RFC 7452, IETF
16. Ouaddah A, Mousannif H, Abou El Kalam A, Ouahman AAIT (2017) Access control in the Internet of Things: big challenges and new opportunities. Comput Netw 112:237–262
17. Bui DT, Douville R, Boussard M (2016) "Supporting multicast and broadcast traffic for groups of connected devices," 2016 IEEE NetSoft Conference and Workshops (NetSoft), Seoul, 2016, pp. 48–52
18. Bell DE, La Padula LJ (1975) Secure computer system: unified exposition and Multics interpretation. In: Technical report, Technical Report MTIS AD-A023588. MITRE Corporation, McLean
19. Denning DE (1976) A lattice model of secure information flow. Commun ACM 19(2):236–243
20. Sandhu RS, Samarati P (1994) Access control: principle and practice. IEEE Commun Mag 32(9):40–48
21. Sandhu RS (1998) Role-based access control. Adv Comput 46: 237–286
22. Samarati P, Di Vimercati SDC (2001) Access control: policies, models, and mechanisms. Found Secur Anal Des 2171:137–196
23. Kalam A, Baida R, Balbiani P, Benferhat S, Cuppens F, Deswarte Y, Miege A, Saurel C, Trouessin G (2003) Organization based access control. In: Proc. POLICY 2003. IEEE 4th Int. Work. Policies Distrib. Syst. Networks, IEEE Comput. Soc, pp 120–131
24. Zhang G, Tian J (2010) An extended role based access control model for the Internet of Things, 2010 International Conference on Information, Networking and Automation (ICINA), 1, IEEE, pp. V1–319
25. Jindou J, Xiaofeng Q, Cheng C (2012) Access control method for Web of Things based on role and SNS, in: 2012 IEEE 12th Int. Conf. Comput. Inf. Technol.,IEEE, pp. 316–321
26. Barka E, Mathew SS, Atif Y (2015) Securing the Web of Things with role-based access control. Springer International Publishing, New York City, pp 14–26
27. Liu J, Xiao Y, Chen CP (2012) Authentication and access control in the Internet of Things, in: 2012 32nd Int. Conf. Distrib. Comput. Syst. Work., IEEE, pp. 588–592
28. I. Bouij - Pasquier, A. Ait Ouahman, A. Abou El Kalam and M. Ouabiba de Montfort, SmartOrBAC security and privacy in the Internet of Things, 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1–8
29. E. Yuan, J. Tong, Attributed based access control (ABAC) for Web services, in: IEEE Int. Conf Web Serv, IEEE, 2005
30. Ye N, Zhu Y, Wang R-C, Malekian R, Qiao-min L (2014) An efficient authentication and access control scheme for perception layer of Internet of Things. Appl Math Inf Sci An Int J 1624(4): 1617–1624
31. Hemdi M, Deters R (2016) Using REST-based protocol to enable ABAC within IoT systems. In: 2016 IEEE 7th Annual Information Technology. Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp 1–7
32. Sciancalepore S et al (2017) Attribute-based access control scheme in federated IoT platforms. In: Podnar Žarko I, Broering A, Soursos S, Serrano M (eds) Interoperability and open-source solutions for the Internet of Things. InterOSS-IoT 2016. Lecture Notes in Computer Science, vol 10218. Springer, Cham
33. Ouechtati H, Ben Azzouna N, Ben Said L (2018) "Towards a self-adaptive access control middleware for the Internet of Things," 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, pp 545–550
34. Hussein D, Bertin E, Frey V (2017) "Access control in IoT: from requirements to a candidate vision," 2017 20th Conference on Innovations in Clouds. Internet and Networks (ICIN), Paris, pp 328–330
35. Salama U, Yao L, Wang X, Paik HY, Beheshti A (2017) "Multi-level privacy-preserving access control as a service for personal healthcare monitoring," 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, pp 878–881
36. Sandhu R (1992) The typed access matrix model. In: Proc. 1992 IEEE Comput. Soc. Symp. Res. Secur. Priv., IEEE Comput. Soc. Press, pp 122–136
37. Lampson B (1974) Protection, ACM SIGOPS. Oper Syst Rev 8: 18–24
38. Oh SW, Kim HS (2014) Decentralized access permission control using resource-oriented architecture for the Web of Things, 16th International Conference on Advanced Communication Technology, Pyeongchang, pp 749–753
39. Dennis JB, Van Horn EC (1966) Programming semantics for multiprogrammed computations. Commun ACM 9(3):143–155
40. Anggorojati B, Prasad NR, Prasad R (2012) Capability-based access control delegation model on the federated IoT network, 2012 15th Int'l. Symp. Wireless Personal Multimedia Commun, pp 604–608
41. Mahalle P (2013) Identity authentication and capability-based access control (IACAC) for the Internet of Things. J Cyber Secur Mobility 1:309–348
42. Hernández-Ramos JL et al (2013) Distributed capability-based access control for the Internet of Things. J Internet Serv Info Secur 3(3/4):1–16
43. Hernández-Ramos JL, Jara AJ, Marín L, Gómez AFS (2016) DCapBac: embedding authorization logic into smart things through ECC optimizations. Int J Comput Math 93(2):345–366. https://doi.org/10.1080/00207160.2014.915316
44. Gusmeroli S, Piccione S, Rotondi D (2012) IoT access control issues: a capability-based approach, 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, pp. 787–792
45. Lynch L (2011) Inside the identity management game. IEEE Internet Comput 15(5):78–82
46. Beltran V, Bertin E, Crespi N (2014) User identity for WebRTC services: a matter of trust. IEEE Internet Comput 18(6):18–25
47. Hussein D, Han SN, Lee GM, Crespi N, Bertin E (2017) Towards a dynamic discovery of smart services in the social Internet of Things. Comput Electr Eng 58, C(February 2017):429–443
48. Armando A, Carbone R, Compagna L, Cuellar J, Tobarra L (2008) Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps. In: Proceedings of the 6th ACM workshop on Formal methods in security engineering. ACM, New York City, pp 1–10

49. Hardt D (2012) The OAuth 2.0 authorization framework, IETF RFC 6749, Oct. 2012, IETF. [online] http://tools.ietf.org/html/rfc6749. Accessed 28 Feb 2019
50. Home - WG – User-Managed Access - Kantara Initiative, [online] Available at https://kantarainitiative.org/confluence/display/uma/Home. Accessed 28 Feb 2019
51. Zhang G, Liu J (2011) A model of workflow-oriented attributed based access control. Int J Comput Netw Inf Secur 1(February): 47–53
52. Hughes J, Maler E (2005) Security Assertion Markup Language (SAML) v2.0 technical overview. OASIS SSTC Working Group, Clovis
53. Hughes J, Cantor S, Hodges J, Hirsch F, Mishra P, Philpott R, Maler E (2005) Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, Burlington
54. Moses T (2005) eXtensible Access Control Markup Language (XACML), version 2.0. OASIS Standard, Burlington
55. Zhang G, Liu J (2012) The study of access control for service-oriented computing in Internet of Things. Int J Wireless Microwave Technol (IJWMT) 2(3):62–68
56. Seitz L, Selander G, Gehrmann C (2013) Authorization framework for the Internet of Things, 2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), Madrid, pp. 1–6
57. Hussein D, Bertin E, Frey V (March 2017) A community-driven access control approach in distributed IoT environments. IEEE Commun Mag 55(3):146–153
58. Jones M, Hildebrand J (2015) JSON Web Encryption (JWE). RFC 7516, IETF, available at http://tools.ietf.org/html/rfc7516. Accessed 28 Feb 2019
59. Hammer-Lahav E (2010) The OAuth 1.0 protocol. RFC 5849, IETF, available at http://tools.ietf.org/html/rfc5849. Accessed 28 Feb 2019
60. Leiba B (2012) OAuth web authorization protocol. IEEE Internet Comput 16(1):74–77
61. Fremantle P, Aziz B, Kopecký J, Scott P (2014) Federated identity and access management for the Internet of Things, 2014 International Workshop on Secure Internet of Things, Wroclaw, pp 10–17
62. Denniss W, Bradley J, Jones M, Tschofenig H OAuth 2.0 device flow for browserless and input constrained devices, Internet-draft, IETF, draft-ietf-oauth-device-flow-09
63. Seitz L, Gerdes S, Selander G, Mani M, Kumar S (2016) Use cases for authentication and authorization in constrained environments, RFC 7744, January 2016, IETF
64. Jones M, Wahlstroem E, Erdtman S, Tschofenig H (2018) CBOR Web Token (CWT), RFC 8392, May 2018. IETF, Fremont
65. Beltran V, Skarmeta AF (2016) An overview on delegated authorization for CoAP: authentication and authorization for constrained environments (ACE). In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), pp 706–710
66. Su X et al (2016) Privacy as a service: protecting the individual in healthcare data processing. In: Computer, vol. 49, no. 11, pp 49–59
67. OMA (2017) Lightweight machine to machine requirements, Candidate Version 1.1 – 08 Dec 2017, Open Mobile Alliance
68. Costa D, Mingozzi E, Tanganelli G, Vallati C (2016) An AllJoyn to CoAP bridge, 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, pp. 395–400
69. Fernandes E, Rahmati A, Jung J, Prakash A (2017) Security implications of permission models in smart-home application frameworks. IEEE Secur Priv 15(2):24–30
70. Zorzi M, Gluhak A, Lange S, Bassi A (2010) From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. IEEE Wirel Commun 17(6):44–51
71. Dillet R (2017) Netatmo is trying really hard to make the smart home happen. January 3, 2017, Techcrunch. Available at: https://techcrunch.com/2017/01/03/netatmo-is-trying-really-hard-to-make-the-smart-home-happen. Accessed 28 Feb 2019
72. Bertin E, Crespi N (2009) Service business processes for the next generation of services: a required step to achieve service convergence. Ann Telecommun 64(3–4):187–196
73. Sloan RH, Warner R (2014) Beyond notice and choice: privacy, norms, and consent. Suffolk Univ J High Technol 14(2):370–412
74. Rastogi V, Welbourne E, Khoussainova N, Kriplean T, Balazinska M, Borriello G, Kohno T, Suciu D (2007) "Expressing privacy policies using authorisation views," in Proc. of the 5th International Workshop on Privacy in UbiComp (UbiPriv'07)
75. Kolias C, Kambourakis G, Stavrou A, Voas J (2017) DDoS in the IoT: Mirai and other botnets. IEEE Comput 50(7):80–84
76. Kovacs E (2016) Hosting provider OVH hit by 1 Tbps DDoS attack. Retrieved from http://www.securityweek.com/hosting-provider-ovh-hit-1-tbps-ddos-attack. Accessed 28 Feb 2019
77. Tian Y, Zhang N, Lin Y-H, Wang XF, Ur B, Guo XZ, Tague P (2017) SmartAuth: user-centered authorization for the Internet of Things. In: USENIX security conference