IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 8, 2017, accepted July 8, 2017, date of publication July 24, 2017, date of current version February 1, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2730958

Expressive Policy-Based Access Control for Resource-Constrained Devices

MIKEL URIARTE[1], JASONE ASTORGA[2], EDUARDO JACOB[2], (Senior Member, IEEE), MAIDER HUARTE[2], AND MANUEL CARNERERO[1]

[1]Nextel S. A., 48170 Zamudio, Spain
[2]Department of Communications Engineering, University of the Basque Country, 48013 Bilbao, Spain

Corresponding author: Mikel Uriarte (muriarte@nextel.es)

This work was supported in part by the Training and Research Unit through UPV/EHU under Grant UFI11/16 and in part by the Department of Economic Development and Competitiveness of the Basque Government through the Security Technologies SEKUTEK Collaborative Research Project.

ABSTRACT Upcoming smart scenarios enabled by the Internet of Things envision smart objects that expose services that can adapt to user behavior or be managed with the goal of achieving higher productivity, often in multi-stakeholder applications. In such environments, smart things are cheap sensors (and actuators) and, therefore, constrained devices. However, they are also critical components because of the importance of the provided information. Therefore, strong security is a must. Nevertheless, existing feasible approaches do not cope well with the principle of least privilege; they lack both expressiveness and the ability to update the policy to be enforced in the sensors. In this paper, we propose an access control model that comprises a policy language that provides dynamic fine-grained policy enforcement in the sensors based on local context conditions. This dynamic policy cycle requires a secure, efficient, and traceable message exchange protocol. For that purpose, a security protocol called Hidra is also proposed. A security and performance evaluation demonstrates the feasibility and adequacy of the proposed protocol and access control model.

INDEX TERMS Access control model, authorization, resource-constrained device, expressive policy language, least privilege, message exchange protocol, mutual authentication, policy codification, sensor.

I. INTRODUCTION

The Internet of Things (IoT) concept involves an enormous collection of novel services and applications, including services that support dangerous processes such as critical infrastructure management [1]. In fact, the IoT concept conceives an interconnected network of things, the smarter the better, contributing to a higher awareness, enhanced decision making, and more adaptive behaviour of systems supporting any business process integrating pervasive and ubiquitous ICT technologies. IoT also implies a massive deployment of sensors and actuators, which, aiming at being cheap, are implemented in a range of constrained devices, constrained device sensors (CDSs) from now on, classified according to [2]. Specifically, the IETF defines a range of CDSs from class 0 (C0, less than 10KB of data and less than 100KB of code), class 1 (C1, approximately 10KB data and 100KB code), to class 2 (C2, 50KB, 250KB), with different purposes and different features, where Moore's law [3] is expected to move more slowly and will have a greater impact on price than on capacity [4], [5]. Moreover, depending on the use case and location, they may require power autonomy, and therefore, demand low power consumption mechanisms. However, in such IoT applications, security (and more specifically, access control) remains an insufficiently solved need.

Within the IoT context two behaviours are distinguished. On one hand, the usual behaviour of a CDS involves a push operation in which the CDS publishes measurements and events to a few large message brokers. Therefore, CDSs behave as information producers. Consumer applications query the message brokers rather than the CDSs directly to obtain field information. Then, because the set of communicating peers is reduced and known beforehand, the security challenges of CDS push mode operations can be easily overcome by preconfiguring static security associations.

On the other hand, more powerful IoT scenarios are also developed, in which sensors also behave as tiny information servers in pervasive computing environments in order to support smarter and more manageable applications. In such scenarios, requesting clients directly query the tiny information servers implemented in the CDSs, establishing a secure end-to-end (E2E) communication, and this proposal conveys

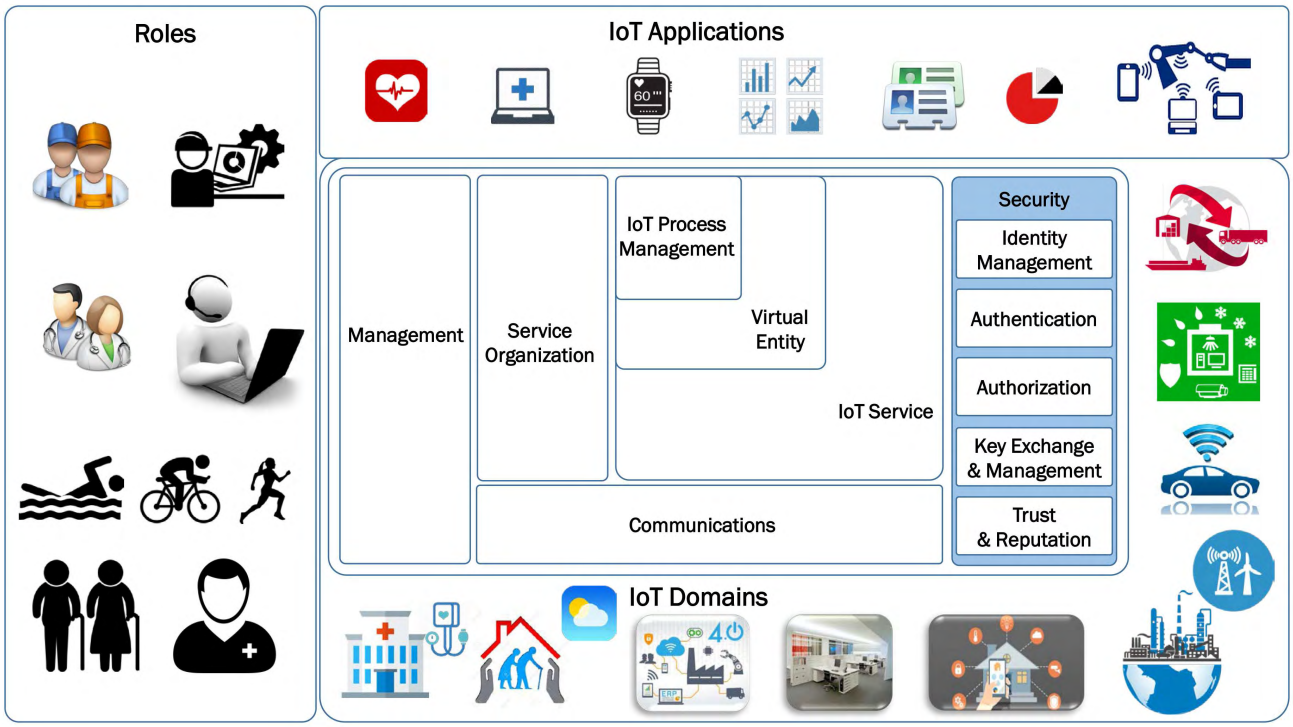


FIGURE 1. Scenario schema where several stakeholders playing different roles access IoT applications on different IoT domains. The core shows the functional decomposition view of the IoT reference architecture ARM [6], where security functional group is highlighted.

the mechanisms for the required fine-grained access control enforcement.

Examples of the considered adaptive environments are smart homes, smart offices, smart elite sports training, smart hydrotherapy rooms, *etc.* Figure 1 shows an IoT schema that conveys different roles in various deployed domains, monitored and controlled through applications and fully aligned with the functional decomposition view of the IoT architecture reference model (ARM) [6]. All these uses involve intelligent services on sensors accessible to either end-users or process managers. These services on the sensors allow some level of personalization to improve the user experience. For example, applications where a patient's profile is used to customize the sensitivity and configure the user identity in a CDS that bulk-monitors health parameters.

Additionally, apart from supporting configuration by end users, CDSs in such smart environments are also required to deliver a set of management and maintenance services to enable their operation to be tuned after deployment. Examples include the manufacturer updating the setup, owners configuring domains, supporters performing maintenance tasks, *etc.* to adapt the features of CDSs to their deployed context such as in the case of critical infrastructures or Industry 4.0 factories. These use cases illustrate both the opportunities and the challenges of the management services exposed by CDSs.

In such contexts, the accuracy and correctness of the information exchanged with CDSs is crucial. Protecting this information requires the implementation of appropriate security mechanisms that include fine-grained access control mechanisms based on expressive policies and that can guarantee essential security properties such as confidentiality, integrity, availability, authenticity and non-repudiation [7]–[10]. However, implementing these security mechanisms in resource-constrained CDSs is not straightforward [11]. Indeed, currently, one of the key challenges for enabling broader adoption of smart things is the availability of feasible access control solutions.

In fact, in 2016, the Dyn cyberattack [12] caused multiple DDoS attacks to many services originating from devices connected to the Internet such as printers, cameras, *etc.* This attack is a representative case that demonstrates the potential disastrous impact of improper access control enforcement on the management services on IoT devices, even if they are not precisely constrained. Actually, vendors and manufacturers are not paying enough attention to security and this issue is even worse in very constrained C0 and C1 CDSs. Furthermore, proposals from the research community, have addressed authentication in various ways, but authorization has received little consideration. In short, the existing feasible approaches for implementing access control mechanisms lack sufficient expressiveness and, therefore, granularity, as well as enforcement features.

Moreover, due to the extremely dynamic nature and purpose of applications based on services in sensors, policy-based security must be enforced locally in the CDSs. In fact, security policies [13] allow security objectives to be modified without changing the implementation of the involved entities.

For this reason, policy based security is often used in systems where flexibility is required because users, services and access rights change frequently, or where context-based conditions must be considered at enforcement time. A policy language in these sorts of smart environments must not only be highly expressive and easily extensible but also lightweight.

The main contribution of this paper is a highly expressive E2E access control model suitable for severely constrained devices (C0 and C1 CDSs). In fact, the proposed attribute based access control model (ABAC) conveys on one hand, a new policy language and codification that are specifically defined to provide expressiveness for authorization policies and to be feasible for CDSs; and on the other hand, a new protocol that enables secure provisioning and enforcement of dynamic security policies as well as an audit trail.

The rest of the paper is organized as follows. Related works are presented in Section II as the state of the art. The proposed ABAC model is specified in three sections, covering 1) the access control model architecture in Section III, 2) the policy domain model based on a specific policy language and policy codification definition in Section IV, and 3) the message exchange protocol for dynamic policy life cycle and runtime enforcement in Section V. The security evaluation is discussed in Section VI and the performance evaluation in Section VII. Finally, the main conclusions of the paper are presented in Section VIII.

II. STATE OF THE ART

In the envisioned IoT scenarios, CDSs behave as servers and requesting clients establish directly a secure E2E communication with these tiny servers. The use of intermediary proxies is avoided for several reasons. On one hand, proxies are specific for each protocol or application and lack flexibility for open scenarios. On the other hand, the use of an intermediary such as a proxy implies the establishment of two independent security associations, but for applications supporting critical processes or when information with high security considerations must be transmitted, it might not be considered acceptable.

In such envisioned IoT scenarios, the networking stack is supported by protocols adapted for network and transport layers such as CoAP [14], [15] and 6LowPAN [16], *etc.*, which mainly adhere to lightweight principles but do not always adhere to security principles. Therefore, security remains the main obstacle to the development of innovative and valuable services [7], [10], [17]. According to Gartner, the security problem is dissuading investors from full adoption of IoT solutions [18]. In fact, knowing that security must focus on network, transport and application layers with different approaches and resulting granularity and robustness, past research on security in wireless sensor networks (WSNs), has concentrated on communication security, such as key management, message authentication, intrusion detection, *etc*. [19], [20]. However, fine-grained data and service access control [21] has barely evolved until recently.

In the last few years, as IoT scenarios fostered by the increasing smartness of CDSs have evolved, new security requirements have arisen, among which fine-grained access control enforcement in CDSs plays a crucial role. Although this issue has been widely researched in the traditional security literature, techniques developed for powerful workstations cannot be directly applied to IoT scenarios due to the severe resource constraints of the devices acting as information servers. Despite that restriction, inspiring traditional approaches for non-constrained device sensors (Non-CDSs) are analysed. In fact, traditional authentication and authorization solutions such as SAML [22], OAuth [23], OpenID [24], Fido [25], [26], *etc.* solve the multi-factor multi-domain access control problem but are based on message exchanges that are not optimized for constrained devices.

Additionally, in the envisioned scenarios, flexibility is enabled by policy driven authorization approaches, and according to Sloman [27], security policies define the relationship between subjects and targets. But there is always a trade-off between the expressiveness of the policy and feasibility. Moreover, traditional implementations such as Extensible Access Control Markup Language (XACML) [28], Rei [29] and Ponder [30], although policy driven and very expressive, are not feasible in CDSs [31]. As represented by its level of adoption, XACML defines a generic authorization architecture and a policy language for expressing and exchanging access control policy information using XML. Unfortunately, although XACML is complete, it is too heavy for resource-deprived devices. For instance, specifying only a single policy rule can easily require more than 50 lines of text, which is a large amount of data for a policy that must be processed locally by a CDS.

Regarding access control approaches specifically tailored to IoT applications. Basically, these do not support expressive (and, therefore, fine-grained and tight) security policy enforcement. Furthermore, the currently implemented static and coarsely grained policies are not well-suited for service-oriented environments where information and management access is by nature dynamic and ad-hoc. For feasibility reasons, one approach could be a centralized architecture, where a central server with no resource constraints makes authorization decisions for each access request to a CDS based on traditional standard mechanisms and protocols. This approach could implement the most effective access control policy languages but it doesn't consider local context conditions in CDSs. In addition, according to [32], it implies high energy consumption as well as high network overhead because requests and responses to on-line authorization decisions must be transmitted through the network.

Analysing the distributed architecture approach, a recent alternative to ABAC or role based access control model (RBAC) is distributed capability based access control (DCapBAC) [33]. The authors of DCapBAC propose an unforgeable token of authority, exchangeable as a capability, that grants access to its possessor in a more agile way,

TABLE 1. Access control feasibility in CDSs classified by resource capabilities, where non-constrained devices are denoted by Non-CDS. An orange X denotes *not feasible*, a green checkmark denotes *capabilities that both are feasible and currently implemented*, and a yellow G denotes *unsolved gap*.

Access control model (sorted by tightness)	Feasibility by CDS classification			
	Non-CDS	C2	C1	C0
Fully expressive policy based access control [27,28,29]	✓	X	X	X
Expressive policy based access control [32]	✓	G	G	G
Simple attribute based access control [34,38,40]	✓	✓	G	G
Access control lists [46]	✓	✓	✓	✓
Just authentication [46]	✓	✓	✓	✓
Additional access control features	**Existence by CDS classification**			
	Non-CDS	C2	C1	C0
Usage based access control [41,42]	✓	G	G	G
Accounting [None]	✓	G	G	G
Flexibility of the policy [27-40]	✓	✓	G	G
Scalability [27-40]	✓	✓	G	G

as the capability based access control model implies [34]. This token includes a subject, permissions, a validity period, and the authorization chain. The token is designed based on an XML schema and the demonstrated scenarios are developed in Java. Therefore, the server side can be considered a small device but not a constrained one. Moreover, security aspects such as entity authentication or digital signatures are not addressed. Therefore, it is not feasible for application to CDSs in IoT resource-deprived sensor networks. Nevertheless, DCapBAC is an interesting model.

Capability based access control models have been adopted by some other initiatives involving technologies specifically designed for IoT environments. Some authors propose a fully distributed approach in which the CDSs can make authorization decisions autonomously [35], [36] that have been successfully applied to smart building scenarios [37]. These proposals are based on an optimized version that uses the elliptic curve digital signature algorithm (ECDSA) to enable public key cryptography (PKC) in CDSs to ensure E2E authentication, integrity and non-repudiation without requiring the intervention of any intermediate entity. Specifically, PKC enables autonomous key establishment among peers and fits well in large scale IoT E2E scenarios. The capabilities (also referred to as tokens or tickets) may additionally include some contextual conditions expressed as type, value, units tuples, that can be evaluated locally in the CDS after the capability has been validated.

Nevertheless, these approaches present several drawbacks. First, they do not sufficiently address the commissioning processes of the capabilities, namely, the lifecycle covering multiple generations, the delivery conditions and constraints, or usage and revocation. In any case, the required local conditions are static because the refresh of the capability is not addressed, and thus, changes in the status of resources in CDSs and changes in context are not covered. Furthermore, considering the policy language, obligations are not supported, and condition checking is limited to matching static values, since it does not support expressions. In addition, while successful validation has been conducted in C2 CDSs with 32-bit CPUs, due to the computational cost and the energy consumption related to the PKC and the length of the capabilities, this approach is not feasible in C0 and C1 CDSs [2], [38].

An aligned alternative approach by the IETF, is the delegated CoAP authentication and authorization framework (DCAF) [39]. This approach is based on using a token to distribute pre-shared keys. Then, if authorized, a handshake is performed to establish a DTLS channel. Authorization policies are specified as local conditions to be checked as attributes. Additionally, the JSON representation for policies has been enhanced through CBOR serialization [40] with the goal of compacting the payloads in the CoAP protocol, and specifically, to compress the length of the capabilities and the enclosed policies based on conditions on attributes. But because CBOR is a general-purpose serialization solution, the resulting compression is still not sufficient for the C0 and C1 CDSs considered in the envisioned scenarios; it is feasible only in C2 CDSs.

An object security architecture (OSCAR) for IoT was proposed in [41]. This scalable approach enables E2E secure access control mechanisms based on PKC that support multicast, asynchronous traffic, and caching. However, OSCAR suffers from the excessive overhead required by DTLS handshakes. Alternatives to avoid this excessive overhead imply delegating the session key establishment process to intermediate entities, which is undesirable from a security point of view. In addition, because a detailed overview of the access control mechanisms was only vaguely discussed, requirements such as local context checking or launching obligations were not addressed.

As a completely different alternative approach, the usage control model [42] and the attribute based policy schema [43] extend traditional non-constrained access control systems, in which the decision to grant access is made before the first access occurs rather than during it, by continuously protecting the resources during access and considering consumption activities. This approach includes obligations such as usage

control, but does not include a proposal addressing its feasibility in any CDS class.

Therefore, Table 1 conveys the feasibility of the aforementioned access control models ordered by their restrictiveness as well as some additional features that may or may not currently exist for each CDS class. To summarize, 1) traditional access control solutions are not feasible in all constrained devices (C0-C2 CDSs); 2) recent access control solutions designed for constrained devices can be implemented only in C2 CDSs and lack policy expressiveness; 3) access control solutions currently feasible in C0 CDSs are based on authentication and very coarse grained and static policies, scale badly, and lack a feasible policy based access control solution. Moreover, no access control solution exists with either accounting or usage control functionalities for any CDS level (C0-C2).

Therefore, no suitable solution currently exists that can provide expressive, policy based, fine-grained authorization services in the envisioned scenarios of constrained but manageable sensor networks. Thus, the aim of this work is to cover the gap in the current state of the art by defining an E2E access control model suitable for C0 CDSs and extensible to C1 and C2 CDSs that solves the poor expressiveness in the existing access control solutions. Additionally, the proposed access control model enables instant provisioning of a custom policy and adds some mechanisms for accurate accounting, with the goal of supporting complete tracking and further auditing as well as flushing local storage.

III. ACCESS CONTROL MODEL

As discussed above, the proposed E2E access control model consists of 1) a combination of a centralized and distributed enforcement architecture to enable multi-step authorization, 2) a policy language and codification that are specifically designed to provide expressiveness for authorization policies while maintaining feasibility in CDSs, and 3) a protocol that enables the provisioning and enforcement of a dynamic security policy as well as comprehensive accounting.

A. ACCESS CONTROL SCENARIO

The conceived network scenario consists of a set of severely constrained CDSs that publish some services. These services can be accessed by subjects through IP networks. Regarding the networking stack, at the link layer, these CDSs communicate with a more powerful base station using specific protocols such as IEEE 802.15.4 [44], which covers short ranges and enables low bit rates. At the network layer, each CDS implements IPv6/6LoWPAN, making it able to establish IP communications with any device in the Internet without the need for intermediary proxies.

There are three basic actors in the access control scenario: a *subject* 1), constrained or not, which intends to access a management service as a *resource* in a CDS 2), with the collaboration of a trusted third party in a step to establish a security association, namely an *access control server* (ACS) 3).

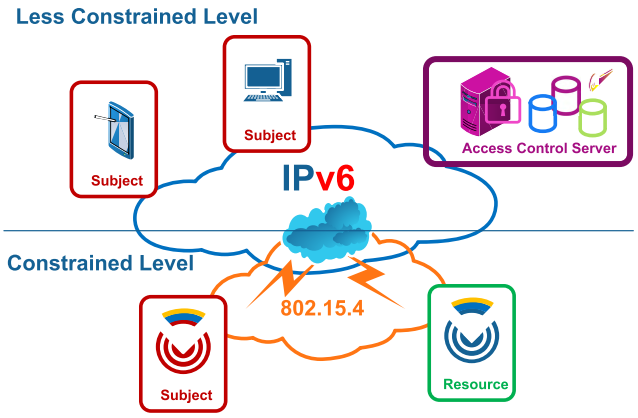


FIGURE 2. Access control scenario.

When a subject wishes to access a resource in a CDS, the well-known three-tier architecture explained in Subsection III-B is adopted. Figure 2 shows a simplified schema of the actors involved, aligned with the IoT reference stack [45].

The trusted third party, that is, the ACS, supports most of the features needed for a complete access control chain such as registration, identification, authentication, authorization, accounting, tracking, auditing and reporting. Specifically, most resource-intensive features rely on the ACS, which performs identity management, cross-domain federated authentication, preliminary authorization, policy life cycle management, and other tasks in the access control chain. This approach allows the usage of standard security and access control technologies that are not feasible given the severely constrained resources of CDSs. This approach also means that most unauthorized access attempts will be refused at this step, avoiding large numbers of unsuccessful message exchanges with the CDSs, and saving energy, which is a crucial aspect. Furthermore, this approach also enables unified and coherent policy management.

In addition to preliminary policy-based authorization, involving strong authentication, another notable feature of the ACS is to fetch, wrap, and inject the local context based policy to be enforced in the CDS. This policy, as well as the issued credentials, can remain valid for some defined period of time after the security association establishment, and it may cover granting rules related to a request, a session, a workflow, and so on in an E2E secure interoperability. Related to instant provisioning, the ACS also initiates accounting, tracking and auditing records based on the first interactions with subjects. These records are later completed by adding further records about E2E access sessions supplied by the CDS, which provides notifications of who performed what, where, when, and under which policy rule.

The CDS support the complementary security features that are needed to enable E2E dynamic and flexible access control based on local context. These features include authentication validation, secondary local-context based authorization, tracking and accounting communications. These features apply not only during the establishment of the security asso-

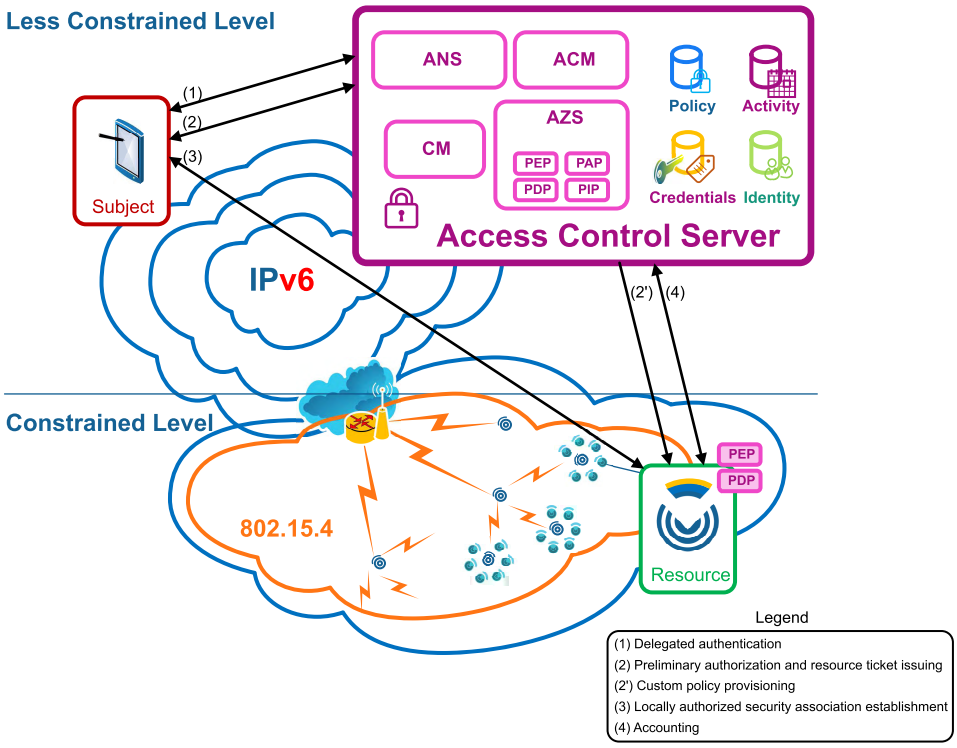


FIGURE 3. Access control architecture.

ciation but also for each derived resource access performed by the subject.

This proposed multi-step enforcement approach, which combines centralized and distributed access control architectures, adopts the benefits derived from each and makes E2E security both feasible and efficient in CDSs.

B. ACCESS CONTROL ARCHITECTURE

The proposed access control architecture consists of two parts: a standardized cross-domain access control central checkpoint as an initial mandatory filter, namely, the ACS, which is not resource-deprived, and a distributed checkpoint in each CDS, that enables dynamic and fine-grained access control, based on local context.

The aforementioned security features are covered by several components located in the three actors involved in the scenario. Figure 3 shows a more detailed view of the architecture of the proposed access control model, which is compliant with the IoT ARM [6], and whose concepts were inspired by non-constrained standards such as the Policy Core Information Model (PCIM), Kerberos, XACML, SAML, and OAuth.

In the ACS, strong authentication is performed by the authentication server (ANS), as depicted in Figure 3 (Step 1). For clarity, related registration and identity management components, which are responsible for the identity repository, are not represented in Figure 3. When a subject requests a ticket to access a resource on a CDS, a preliminary authorization is performed in the authorization server (AZS), where the granting decision is made in a policy decision point (PDP) decoupled from the policy enforcement point (PEP) as in XACML, and depicted in Figure 3 (Step 2). There are two other complementary components in the AZS that support PDP decisions: the policy repository administration point (PAP) and the policy information point (PIP). A repository for unified policies is also included. This repository stores policies for CDSs instantiated according to the policy language defined in this proposal. The proper policy instance is provisioned in the CDS as depicted in Figure 3 (Step 2'). In this manner, policy consistency is maintained but remains amenable to be edited, maintained, checked, changed and provisioned as needed. The policy language and the policy domain model, defined in this proposal to enable the instantiation of expressive but feasible policies to be enforced in the CDSs, are discussed in Subsections IV-A and IV-D respectively.

The credential manager (CM) component generates and releases resource tickets with the credentials of the authenticated and preliminarily authorized subjects. Repositories with credentials, namely tickets and cryptographic keys, and accounting records, which are initiated when tickets are issued, are also included in the architecture. The additional components of the ACS that cover accounting, tracking, and auditing are grouped in the Accounting Manager (ACM). A related repository to record any activity to be further analysed is also included.

All these highlighted components are integrated into the ACS, while tailored PDP and PEP are deployed in each CDS. In fact, a customized PDP engine is deployed in each CDS. When a security association request arises, the PDP checks the enclosed resource ticket and local conditions specified in

the related policy instance to make a local decision to grant access, as depicted in Figure 3 (Step 3).

PDP granting decisions, additional obligations and policy re-evaluation functions are then enforced by the PEP in the CDS and notifications are sent to the ACS as depicted in Figure 3 (Step 4). When deploying a PDP in each CDS, the PDP engine and the policy language can be at a lower abstraction level and, therefore, specific to the CDS system being protected, making implementation simpler and more efficient, and allowing more granular policies. Additionally, local PDP-PEP communications reduce the time required to obtain decisions granting access. The runtime overhead introduced by custom policy instance deployment is reduced because of the small size of policy instance codifications, as explained in Subsection IV-C. The impact of this approach is much lower than opting to store all the potential policies in the CDSs, which is in fact unfeasible. It is also a more efficient option than querying a central PDP for every resource request, an approach that would underperform due to network dependencies and delays and would consume much more energy. This is because, according to [32], sensor energy consumption related to network transmission/reception operations is three orders of magnitude higher than energy consumption related to logic processing operations, as explained in Section VII which presents validation measurements.

Deploying PDP in the CDS, combines the advantages of central unified policy administration, instantaneous policy instance customization, and local decision making. This enables the dynamic delegation, request, cancellation and revocation of permissions. This is performed by refreshing the instant policy instance during each E2E security association establishment phase and can be performed during the lifetime of a security association. The updated policy is then enforced autonomously in the CDS through the deployed PDP and PEP when any request arises.

These components of the access control architecture enable a feasible multi-step authorization in which the main contribution is the policy language, its codification and the policy domain model. The three of these, discussed in the remaining sections, empower the instantiation of the fine-grained policies to be enforced in CDSs to comply with the principle of least privilege, their feasibility in resource-constrained CDSs (C0, C1) and their scalability respectively. Therefore, they endow the proposed E2E access control model with the flexibility derived from an expressive policy driven authorization scheme.

IV. AUTHORIZATION POLICY LANGUAGE

This section defines an expressive policy language whose goal is to enable the enforcement of more restrictive access control policies in the CDS that overcome the resource constraints. In fact, this policy language definition enables CDSs to make granting decisions based on local context conditions and to react accordingly to requests.

This policy language adopts the deontic concepts of rights, prohibitions, and obligations; and it is declarative, (i.e. interpretation follows one previously declared logic) instead of the procedural approach, where a policy instance also includes the steps to follow to produce a result, as discussed in [13].

This policy language was inspired by the full expressiveness of the solutions applied to non-constrained devices. It can be considered a subset of the XACML policy language. Although it does not include all the features of XACML, it does include a simpler definition of the most useful ones. The design decisions were made by balancing expressiveness and feasibility in C0 CDSs.

A resulting policy instance is defined, as in the general event-condition-action approaches, as an optional set of rules that specify both the conditions to be checked and the related reactions at enforcement time. Specifically, this policy language stands for a sequence of constructs with specific meanings at decision-making and enforcement time. The order of the sequence is crucial to its correct interpretation.

A. AUTHORIZATION POLICY LANGUAGE CONSTRUCTS

On one hand, some of the constructs are defined as mandatory, and some others as optional, which can reduce the length of the policy when a use case requires only a simple policy instance. This defined policy elasticity is a crucial characteristic which supports adjusting policy length to use case requirements. Policy length is a key performance indicator of access control approaches in CDSs: the shorter a policy is, the better the performance is; therefore, policy length impacts feasibility and promotes higher autonomy.

On the other hand, some constructs are extended through other nested constructs, and some can be instantiated many times within a container construct.

The more constructs a policy supports, the higher its expressiveness is. Similarly, the more granular a policy is, the tighter its enforcement can be. Below, the defined constructs are described along with their nesting relationship. The main construct is the policy construct, which is mandatory in any policy instance.

1) POLICY CONSTRUCT

The *policy* construct enables a policy instantiation, which contains three nested constructs.

First, a policy instance identification, *id*, must be specified for logging, tracking and auditing purposes. Then, a default policy granting *effect* is specified. This effect prevails in the absence of other applicable rules, or when any rule evaluation conflict occurs. This construct is useful in most simple policy instances that have no rules, where authentication or even preliminary authorization in the ACS is sufficient to grant access for a request. It is also useful for revocation notification and related security association finalization. Finally, optionally, an array of rules comprising a *ruleset* may be instantiated to specify additional conditions and related reactions. Each *rule* in the array is an extensible construct that will be detailed in Subsection IV-A.2.

TABLE 2. EBNF representation of a schematic policy.

```
policy ::= '{' '"id"' ':' integer ',' '"effect"' ':' ( DENY | PERMIT ) ( ',' '"ruleset"' ':' '[' rule+ ']' )? '}'
```

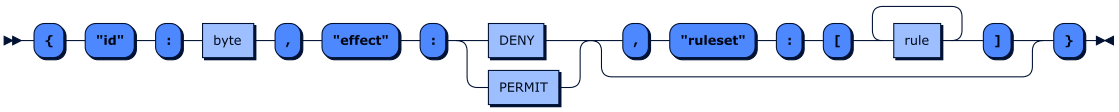


FIGURE 4. Policy construct definition.

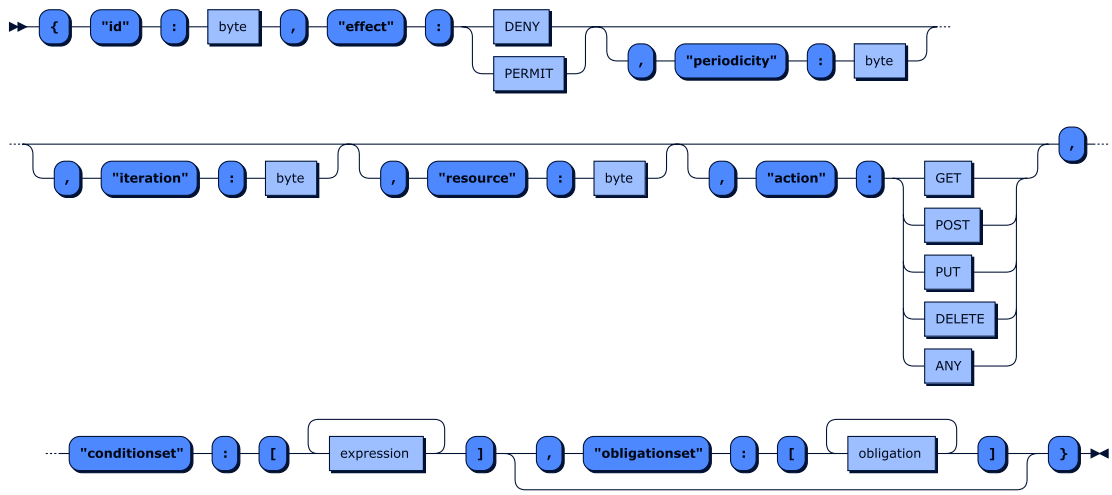


FIGURE 5. Rule construct definition.

The three constructs that constitute a policy, their content, and their optionality are also represented in Extended Backus-Naur Form (EBNF) notation, as listed in Table 2. EBNF is nowadays one of the two main notation techniques used to describe computer language syntax such as policy languages [46]. EBNF enables meaningful constructs to be represented along with expandable, nesting and repetitive constructs, as needed.

The three constructs that constitute a policy, their content, and their optionality are represented in a graphical view in Figure 4. The graphical view is an equivalent representation and it will be used hereafter to enhance clarity.

Note that the simplest policy instance is one that grants or denies access but requires no further rules. The length of such a simple policy instance is minimal, as discussed in Subsection IV-C.

2) RULE CONSTRUCT

A *rule* construct is defined as a sequence of eight nested constructs whose order is crucial. Some of these constructs, such as *id*, *effect*, and *conditionset*, are mandatory, and the rest, named *periodicity*, *iteration*, *resource*, *action* and *obligationset*, are optional. The sequence of the constructs, their content and their optionality in a rule are presented in Figure 5.

The *conditionset* and the *obligationset* are arrays of expressions and obligations, respectively. These repeatable and extensible *expression* and *obligation* constructs are described in separate subsections.

Starting from the top left, the mandatory constructs in a rule are the identifier *id* (1), and the granting *effect* (2) of the rule. These two constructs must appear first in any rule and are key values for further tracking and auditing purposes.

Then, the optional constructs named *periodicity* (3) and *iteration* (4) govern the rule's evaluation behaviour. Periodicity determines how often the rule must be evaluated, and iteration determines how many times it must be evaluated. The combination of these two constructs enables rule evaluation not only before an access request occurs but during or even after the request, and constitute a way to activate and expand the potential of obligations (explained later). In any case, they allow rules to require reiteration and support temporal constraints such as expiration checks. Both the timer and the counter for rule re-evaluation are implemented as part of the PEP, which will invoke the PDP for decision making and enforcement feedback.

Then, optional constructs related to the targeted *resource* (5) and requested *action* (6) identifications can be instantiated to govern rule matching. This matching feature allows specifying several rules in a unique policy specification that protect several different resources with different actions or intentions. Policy instances containing different rules are exchanged just once per authentication and preliminary authorization process in the ACS as explained in Section V. This approach not only enables evaluating the correct rule when establishing a security association but also evaluating the appropriate rule at

FIGURE 6. Definition of the expression in a condition set.

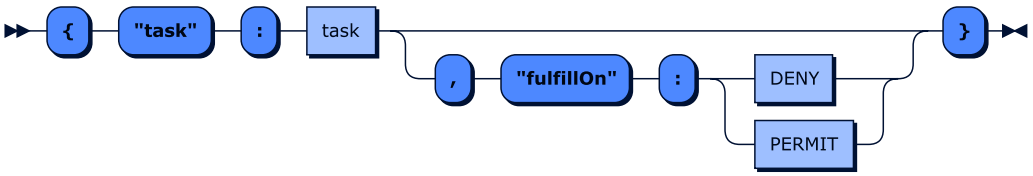


FIGURE 7. Obligation construct definition.

each further resource request, while the security association is still in effect. Any conflict that occurs between different rules is avoided by applying the most restrictive approach (i.e. final granting decision is *permit* only when the effect of all the evaluated applicable rules is *permit*).

Then, the *conditionset* construct (7) is mandatory. A *conditionset* is defined as an array of extendable expressions as detailed in Subsection IV-A.3. The definition of expressions as conditions to be checked enables the specification of more expressive conditions than do existing approaches, as explained in Section II.

Finally, the *obligationset* construct (8) provides a way to specify specific reaction to a rule match, allowing advanced features such as precise logging, active accounting, system blocking, transaction-level control implementation, usage-based authorization, *etc.* Note that even the simplest action such as updating a counter or an attribute value may modify the result of further rule evaluations specified by iteration constructs, as discussed above. The combination of both *obligationset* and *iteration*, enables the adaptation of CDS security to activity, use and context evolution, increases the expressiveness, and paves the way to smart access control enforcement in smart CDSs.

3) CONDITIONSET CONSTRUCT

This mandatory extendable rule construct enables defining an array of expressions that should be evaluated (as opposed to the currently available simple static matches of variable name, value and type tuples that imply longer strings with less expressiveness). In fact, an *expression*, equivalent to one condition, is a function identified by an identifier *id*, that accepts an array of optional attributes and can be instantiated as shown in Figure 6. These functions can be drawn from a rich library of logical, arithmetic, textual, relational, comparison, *etc.* operations supported by CDSs, as well as available custom functions developed for specific purposes; all of these are indexed in appropriate tables. Moreover, these reference tables are defined as part of the policy domain model (PDM), as explained in Subsection IV-D.

The *attribute* construct is defined as a nested construct of the *inputset* construct in Subsection IV-A.6.

4) OBLIGATIONSET CONSTRUCT

The *obligationset*, which is an optional extendable rule construct, enables the definition of an array of obligation tasks to be launched consecutively, depending on whether the granting decision of the rule is DENY, PERMIT, or for either case, ALWAYS, as shown in Figure 7.

The construct named *task*, which is nested in an *obligation* construct, is described in Subsection IV-A.5.

5) TASK CONSTRUCT

The *task* construct, which is mandatory in an *obligation* instance, enables the specification of a function to be executed as an obligation. A *task* is a function referred to by an identifier, *id*, that accepts an array of optional attributes, and it can be instantiated as an expression in a condition, which follows the structure shown in Figure 6. As is the case with *expressions* in a *conditionset*, the functions referred to by a task may belong to a rich library of logical, arithmetic, textual, relational, comparison, *etc.* operations supported by CDSs, as well as custom functions developed for specific purposes; all these are indexed in appropriate tables as shown in Figure 9. In fact, these reference tables are defined as part of the PDM, as explained in Subsection IV-D.

6) INPUTSET CONSTRUCT

An *inputset*, which is an optional construct invoked in an expression or in a task, enables the specification of an array of attributes considered as input arguments. An attribute is a type-value pair. Figure 8 shows a list of defined attribute types in which well-known types are complemented with some indirect references. These three reference types are address attributes related to a request, attributes related to the system, and attributes related to the results of current expression evaluations.

Request references enable addressing any additional attributes related with the subject, the action or the resource involved in a request. Related to the subject, not only the user name may be checked for accounting purposes, but additional attributes concerning the origin such as source IP, device type, location, authentication method, *etc.* can be checked during the granting decision. Similarly, during the request, input parameters related to the requested action or resource can be checked during the granting decision.

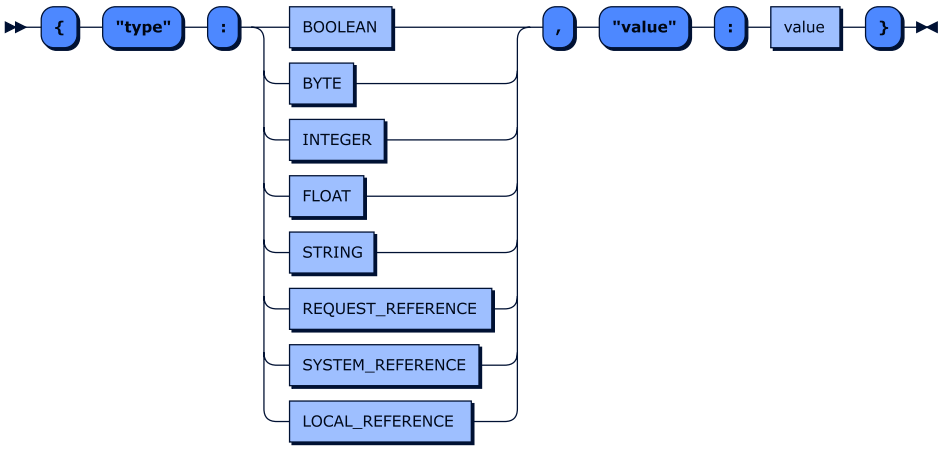


FIGURE 8. Attribute construct definition.

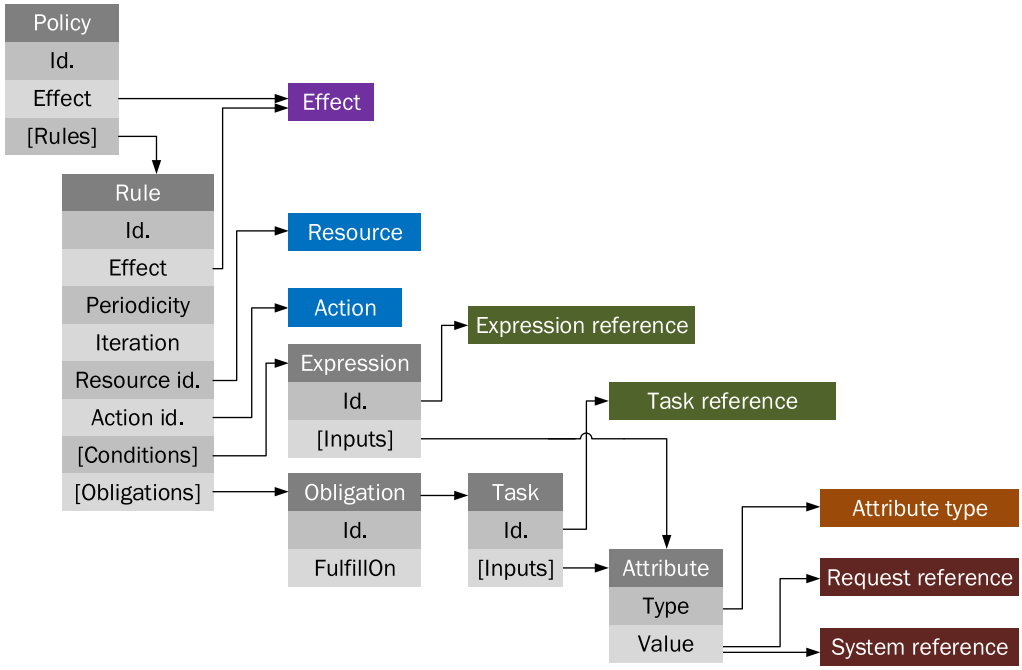


FIGURE 9. Policy domain model.

Local references enable the reuse of the results of previously evaluated expressions as inputs to expressions, or as inputs to tasks, enabling higher expressiveness through correlated conditions and obligations. Local references, complemented by system and request references enable higher expressiveness through more contextual conditions and obligations. These reference types also allow managing not only single-valued attributes (atomic) but also multi-valued attributes (set-valued), because expressions and tasks using these reference types as inputs can expect such input. The resulting output is expected to be an atomic Boolean. This feature is very useful when working with blacklists, revocation lists and so on.

These sets of attributes enable flexible and expressive ABAC model implementations, allowing checks not only of fully expressive local instantaneous context conditions but also historical conditions.

B. POLICY INSTANCE

All the nested constructs in the main construct named *policy* defined so far enable the instantiation of an expressive policy. This policy instance is evaluated in the PDP in the CDSs, triggered during the first occurrence of an event, usually a request. Consequently, local conditions are checked according to the matched rule. Then, proper reactions are performed, which include enforcing the granting decision and executing corresponding obligations.

This policy language can be used to implement most extended access control models (i.e., RBAC and ABAC models)

using the same sets of constructs in different domains, leading to simpler policy instances and more uniformity.

Any policy instance that follows the policy definition in this section is a text file that can be generated in different formats, stored and processed by information systems, and edited by human administrators.

The policy expressiveness proposed in this section would be useless if the length of the resulting policy instances were too long. This proposal provides a policy codification to compress the policy length. The compression supports a gain in expressiveness and limits the impact on storage and energy consumption. The codification of the policy instances is defined in the following section.

C. POLICY LANGUAGE CODIFICATION

This section specifies a codification of the policy instances that notably reduces the lengths of policy files. The goal of this codification is to generate the minimum possible bit stream, starting from a textual policy instance generated following the policy language format explained in Section IV.

Proposed policy codification is distinct from existing proposals that serialize policy instances through standardized open solutions such as CBOR. The available existing serialization approaches do not optimize the agreed-upon common understanding of the constructs by means of their sequence, their meaning, their type, their scope, or the policies' elasticity. Consequently, the compression ratio of currently available approaches is below the ratio resulting from the policy codification proposed in this section.

Therefore, this proposed policy instance codification takes advantage of pre-existing knowledge of the agreed-upon sequence of the constructs and their format. An additional crucial factor is not only the bit conversion of the content of the constructs, but the injection of some predetermined bit masks that specify whether optional constructs exist. This approach optimally considers the elasticity defined in the policy language; by avoiding unused optional fields of expressive policies, it greatly reduces policy length. Thus, the proposed policy bit codification significantly shortens the length of policy instances to be provisioned in the CDSs during the establishment of a security association as introduced in Subsection III-B, making dynamic policy cycles feasible and avoiding having to store large policies in the CDSs.

Because it accounts for policy formats, the policy codification specified in this subsection can easily be applied to any original policy instance format (XACML, JSON, *etc.*), from plain text files to structured policy instance representations. Policy codification is specified in the next subsections, taking as input a JSON representation of the policy, which is considered pertinent for two reasons. First, JSON is a broadly adopted format of the payload in CoAP, which is a specialized web transfer protocol designed specifically for constrained devices [14], [15]. Second is clarity, because JSON is simple and readable by humans.

To explain the specific codification and derived optimizations, the codification is explained in detail for the *policy* and *rule* constructs, and it can be looked up in the Appendix for the rest of the nested constructs.

1) POLICY CONSTRUCT CODIFICATION

The *policy* construct consists of three nested constructs, where *ruleset* is expandable. Table 3 shows the binary representation resulting from the codification of a human-readable and editable policy instance represented in JSON.

The policy related constructs are cited in the leftmost column in Table 3. The JSON representation is presented in the second column and has been split into rows for clarity. Each row is a field containing, on one hand, the construct label, and on the other hand, the characters used as JSON separators.

According to the policy language definition, some constructs are mandatory and others are optional, and so are the related fields (i.e., labels and separators), as presented in the third column. Due to the strict order in the construct sequence, most fields can be omitted in the resulting bit stream. For optional constructs, some additional bits are injected in the binary representation to indicate their existence, as specified in the third column. This bit injection allows compressing the policy stream to the minimum, enabling the policy elasticity as a key feasibility and scalability characteristic. Some other injected bits are used to specify the number of elements in the arrays of nested constructs. Again, codification adds only meaningful fields to the final policy bit stream.

The fourth column shows the codified binary representation, while the fifth column shows the description, including bit ranges; therefore, related length can be easily derived.

Any resulting policy bit stream will always start with the binary value of the policy identifier, followed by an effect binary code, and a mask named *RuleExistenceMask* that specifies whether rules have been instantiated. When *RuleExistenceMask* indicates that rules are present, an array with at least one rule is expected and codified. The maximum rule index is used to specify the number of rules within the array and codified through the injected *MaxRuleIndex* value. Rule codification is expanded in Table 4.

Note that this codification specification applies some agreed-upon conventions that are included in the PDM specification. In fact, the policy codification table shows that the maximum number of rules in a policy is eight. This decision, taken as a sample, may cover most use cases in CDSs, but in any case, it can be easily adjusted to particular use cases, by means of parametrization of the PDM, as explained in Subsection IV-D.

2) NESTED CONSTRUCTS CODIFICATION

Nested constructs such as rules are all codified in the same way and are specified in Table 4, which contains the same columns as Table 3, but some rows related to omitted texts are compressed.

Within the rule construct, the rule identifier is the first coded value, followed by the rule effect code, and a mask of five bits that specifies the existence of the five optional constructs.

TABLE 3. Policy codification.

	JSON	NATURE	BINARY	DESCRIPTION
POLICY	{	Mandatory		Omitted
	id	Mandatory		Omitted
	:	Mandatory		Omitted
	byte	Mandatory	[0b00000000 - 0b11111111]	Policy id. [0 - 255]
	,	Mandatory		Omitted
	effect	Mandatory		Omitted
	:	Mandatory		Omitted
	DENY/PERMIT	Mandatory	[0b0 - 0b1]	Policy effect [0 - 1]
		Injected	[0b0 - 0b1]	RuleExistenceMask [0 - 1]
		Injected	[0b000 - 0b111]	MaxRuleIndex [0 - 7]
	,	Optional		Omitted
	ruleset	Optional		Omitted
	:	Optional		Omitted
	[	Optional		Omitted
	rule+	Expanded	array of rules	At least one rule
	]	Optional		Omitted
	}	Mandatory		Omitted

TABLE 4. Rule codification.

	JSON	NATURE	BINARY	DESCRIPTION
RULE	{ id :	Mandatory		Omitted
	byte	Mandatory	[0b00000000 - 0b11111111]	Rule id. [0 - 255]
	, effect :	Mandatory		Omitted
	DENY/PERMIT	Mandatory	[0b0 - 0b1]	Rule effect [0 - 1]
		Injected	[0b00000 - 0b11111]	Logical jumpers
				PERIODICITY_MASK = 0b10000
				ITERATION_MASK = 0b01000
				RESOURCE_MASK = 0b00100
				ACTION_MASK = 0b00010
				OBLIGATIONS_MASK = 0b00001
	, periodicity :	Optional		Omitted
	byte	Optional	[0b00000000 - 0b11111111]	Time (minutes) [0 - 255] (>4h)
	, iteration :	Optional		Omitted
	byte	Optional	[0b00000000 - 0b11111111]	Repetitions [0 - 255]
	, resource :	Optional		Omitted
	byte	Optional	[0b00000000 - 0b11111111]	Resource id. [0 - 255]
	, action :	Optional		Omitted
	GET/POST/PUT/DELETE/ANY	Optional	[0b000 - 0b100]	Action id. [0 - 4]
		Injected	[0b000 - 0b111]	MaxExpressionIndex [0 - 7]
	, conditionset : [	Mandatory		Omitted
	expression+	Expanded	array of expressions	At least one expression
	]	Mandatory		Omitted
		Injected	[0b000 - 0b111]	MaxObligationIndex [0 - 7]
	, obligationset : [	Optional		Omitted
	obligation+	Expanded	array of obligations	At least one obligation
	] }	Mandatory		Omitted

When the optional periodicity construct exists, a value specifying the time (for example in minutes) between iterations is coded. And when the optional iteration construct exists, a value specifying the number of repetitions is coded. Similarly, when the optional resource and/or action constructs exist, the resource and/or action identifier is coded, respectively.

Conditions are mandatory and are codified as an array containing at least one expression. For clarity, as an example, eight conditions are estimated to be sufficiently expressive; therefore, the maximum index in the array of expressions [0-7], *MaxExpressionIndex*, is injected to specify the number of instantiated expressions [1-8]. In any other use cases, the maximum number of conditions could be easily adjusted, by means of parametrization of the PDM, as explained in Subsection IV-D.

When obligations exist, an array containing at least one is expected. The number of obligations is expressed before the details of the obligation are codified. The injected maximum index value, *MaxObligationIndex*, will be used by the decoder to deduce the number of obligations.

The rest of the nested constructs such as obligations, tasks, expressions, inputs and attributes are codified in a similar way and they are conveyed in Anex VIII.

TABLE 5. References and constants.

Concept	Type	Range	Description
System attributes reference	byte	[0 - 255]	It requires an additional table indexing available attributes in the system, to be used in expressions
Request attributes reference	byte	[0 - 255]	It requires an additional table indexing available attributes in the request, to be used in expressions
Local attributes reference	bit (3)	[0 - 6]	It addresses previous expressions' index to reuse the output value
Expression functions	byte	[0 - 255]	It requires an additional table indexing available functions to be used in expressions
Task actions	byte	[0 - 255]	It requires an additional table indexing available functions to be used in tasks
Target resource	byte	[0 - 255]	It requires an additional table indexing resources of CD

All of the agreed-upon definitions, such as the maximum number of elements in the arrays of rules, expressions, input attributes, obligations, and so on, which represent a good balance between functionality and performance, are also included in the PDM specification.

D. POLICY DOMAIN MODEL

There is a trade-off between the expressiveness and the length of the policy instance, because the policy length results from the expressiveness required by the use cases.

The maximum length of the codified policy bit stream, including the maximum possible rule length that recursively includes all optional constructs and the highest values, is 1024 bytes. Even though a length of 1024 bytes is not acceptable for every CDS, it covers the required expressiveness in most use cases. In any case, this total length of a codified policy could be assumable in less-constrained devices.

Moreover, the length can be modulated through some assumptions defined in the policy domain model. This feature supports different gradual implementations that could be patterned after most current CDS implementations.

In fact, in the aforementioned codification tables, some semantic conventions have been adopted as summarized in Table 19.

The adopted conventions define a short set of possible values in a construct where the universe of the set is well-known and predetermined. However, some fields are identifiers that refer to entities that can be values, types, labels, attributes or functions, instead of the entities themselves, enabling a much shorter codification and adding agility to the processing. The references related to attributes, functions, and resources are specified in Table 5. A range of up to 255 different values has been adopted as example in most cases. This range is likely larger than needed, and it can be shortened by design depending on the policy domain to balance scope and performance. Note that in the case of local attribute references, the range is related to the number of conditions. Here, the number of conditions has been set to eight in the example because 8 is sufficient for most use cases in severely constrained devices (e.g., C0 CDSs).

Therefore, each entry in this table requires an additional index table to be specified within the PDM. The PDM consists of the detailed definition of the policy language and the adopted conventions, as denoted by the tables depicted in Figure 9. Specifically, the set of constructs and their expansion is presented (grey), and the bindings with related reference tables (colourful), are summarized for clarity.

Note that the high compression factor of the policy codification is predicated on the concept that knowledge is built into the PDM.

This specific common understanding by E2E peers can be agreed upon as PDM specifications. In implementations where the PDM is specified in separate files, their versioning, modification, provisioning and activation is much more agile. That is, by separating the PDM specification files, better abstraction and multi-domain applicability can be obtained.

The separate specification of these agreements corresponding to a policy domain enables agile change management. When there is no explicit representation of its content, these tables are understood to be implicitly known by both ACS and CDS peers. However, this option would be rigid, and changes would be complicated, difficult, and, could easily become unmanageable.

On the other hand, the files constituting this PDM specification enable different profiles to be defined through patterning, which improves scalability and flexibility, because, depending on the resource constraints of the CDSs, resource granularity and access control policy expressiveness can be adjusted.

E. RESULTING POLICIES REVIEW

Thanks to the elasticity enabled by the policy language, when instantiating policies ranging from simple cases to highly sophisticated ones, the lengths rise progressively. As an example, four representative samples have been defined as listed below:

- Sample 1: A policy with no rules comparable in expressiveness to Ladon, which has been validated in C0 CDSs [47]. This example might be access granted to a subject initially authenticated and authorized in the ACS and then authenticated in the CDS.
- Sample 2: A policy containing one rule with conditions, somewhat comparable in expressiveness to the existing DcapBAC approaches The proposed approach enables

TABLE 6. Codification lengths of four instances of samples.

Sample	Instance of sample (IS) and description	Length		Comparison		
		bits	bytes	Class 0	Class 1	Class 2
1	IS1: No rules, just policy id. (for tracking) and granting effect	10	2	✓	✓	✓
2	IS2: One rule with one condition with one input	53	7	Beyond State of the Art	✓	✓
3	IS3: One rule with one condition and one obligation	67	9			✓
4	IS4: Two rules with three conditions and one obligation, and periodical re-evaluation during access session	258	32			✓

rich attribute expressions rather than only the simple matching of DcapBAC. For example, beyond initial authorization in the ACS and local authentication in the CDS, access for any maintenance action can be granted only after ensuring that battery level exceeds a given threshold.

- Sample 3: A policy with one rule containing both conditions and obligations. This level of expressiveness is beyond the existing feasible solutions and enforcement granularity. For example, after checking a local condition such as battery status while granting any maintenance action, the system status flag or semaphore can be updated as a reaction enabled by obligations.
- Sample 4: A policy with two rules containing conditions, obligations and periodic re-evaluation. This level of expressiveness is far beyond the capabilities of existing solutions. This example incrementally covers checking different conditions related to different actions on different resources. For example, it could enforce checking system status attributes before maintenance actions can occur and/or attempt counter-checking before granting administration rights, which both highlight its finer granularity. Additionally, each access may produce specific reactions through obligations such as updating system flags to enable transactional controls, updating counters to enable usage controls, or performing concrete remote notifications that enable instant awareness.

For such usable samples, the lengths of specific policy instances vary depending on the specific field values and iterations of nested constructs they contain. For each of the four samples, exemplary Instances of Samples (IS_i) have been considered to calculate specific lengths. Table 6 shows the detailed lengths of the sample instances and their expressiveness, as well as an overview of the gap with respect to the state of the art.

It is also pertinent to compare the resulting lengths with the existing representations and serializations of the policy instances. The samples as represented in JSON and CBOR result in the lengths shown in Table 7, where the authorization policy binary representation specified in Subsection IV-C is denoted by Authorization Policy Binary Representation (APBR).

Approaches for transmitting and processing JSON format policy instances, which is the de-facto standard adopted in CoAP, are much too long. Note that a slight reduction in JSON representation can be obtained when identifiers and

TABLE 7. Length comparison for different representations in four instances of samples.

REPRESENTATION	NATURE	LENGTH (bytes)			
		IS_1	IS_2	IS_3	IS_4
JSON	Human readable text	30	164	236	798
JSON	Pre-processed text	23	118	174	554
CBOR	Binary stream	14	81	123	391
APBR	Optimized binary stream	2	7	9	32

PDM related conventions are applied, but the lengths remain high. Existing approaches to transmitting and processing policy instances in CBOR format result in shorter lengths than JSON, but they are still longer than the proposed approach because they are not optimized for policy specification and interpretation purposes. Consequently, the proposed policy codification, APBR, results in the best compression factor. APBR is the shortest; therefore, it will have the lowest impact on storage, transmission and processing. Additionally, looking at Table 6, this proposal presents a feasible solution that significantly raises the expressiveness of authorization policies for CDSs.

V. HIDRA MESSAGING PROTOCOL

The components of the access control architecture need to enable the related authentication and fine-grained authorization features, where trusted instantaneous policy provisioning in the CDS is required to enable a dynamic life cycle of expressive policies, as well as to avoid local storage in the CDS. The protocol supporting such interactions, which integrates authentication and authorization, is specified in this section.

The proposed protocol, named Hidra, is based on Ladon [48], which is a formally validated solution for establishing E2E security associations through pair-wise keys, guaranteeing mutual authentication and authorization in severely constrained CDSs (C0). Ladon, which is based on Kerberos, also ensures security properties such as confidentiality, integrity and non-repudiation, and it is resistant to replay attacks. Additionally, this protocol addresses further specific requirements related to the envisioned scenario: independence from clock synchronization, energy efficiency that has been validated in severely constrained devices (C0), resistance to message losses and scalability. However, Ladon lacks policy provisioning and accounting functionalities. In fact, Ladon enables the transmission of authenticated values for a subject's role attribute,

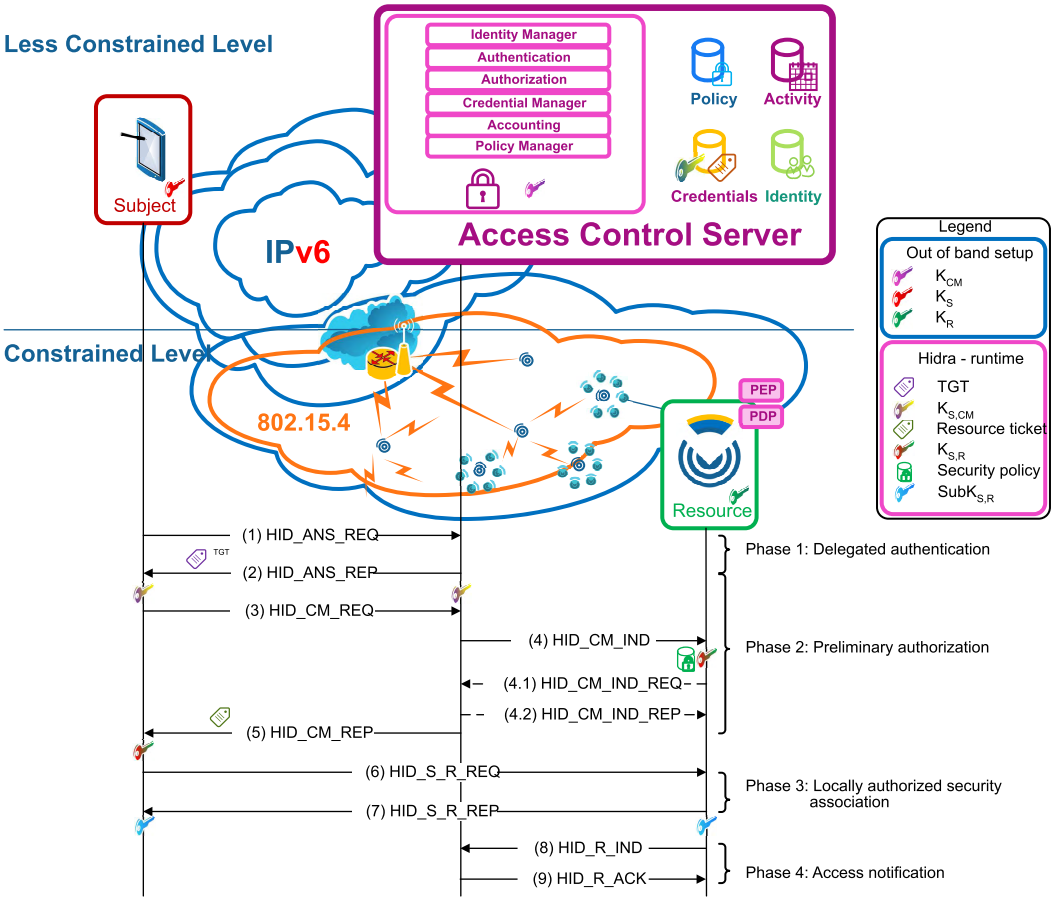


FIGURE 10. Hidra protocol.

but its policy is static: the CDSs have a preconfigured set of valid attributes. Thus, Ladon is not a flexible solution and scales poorly.

In the message exchange sequence of the Hidra protocol, depicted in Figure 10, a novel approach is proposed that can dynamically include an authorization policy instance to be enforced in a CDS through modifying one of the messages defined in Ladon. The second novelty in the Hidra protocol is the addition of a pair of messages to enable precise accounting. A CDS can provide notifications of details such as who performed what, where and when for every access request received from the subject. These notifications are collected, normalized, and treated appropriately by the ACM in the ACS.

Table 8 presents the terms used as abbreviations and the notation, while Table 9 details the contents of the exchanged messages. Because they are based on Ladon, the following description of Hidra omits a detailed description of the entire protocol; only the novelties are conveyed.

Both Hidra and Ladon are based on symmetric key cryptography, and they assume that each endpoint owns a secret key shared with the ACS. The configuration of this key is part of an initial configuration procedure performed during bootstrap and conducted in a controlled environment (typically out of band, for example, through a USB) in all CDSs before network deployment.

Hidra operations are based on the use of tickets, capabilities distributed by the ACS that contain proof of the identity of the subject making the request. Tickets are encrypted so that only the intended entities are able to decrypt them. Each subject needs a ticket issued by the ACS to request any resource on the CDS. Therefore, the subject first authenticates against the ANS in the ACS (see Phase 1 in Figure 3) and obtains a long-term ticket known as a ticket granting ticket (TGT), through HID_ANS_REQ/_REP messages (1 and 2). This ticket allows the subject to securely communicate with the credential manager CM in the ACS, which is responsible for issuing the actual resource tickets.

In the ACS, in Phase 2, resource tickets are generated only for authenticated and preliminarily authorized subjects through the HID_CM_REQ/_REP messages (3 and 5). Enclosed in its payload field, the resource ticket includes the proof of the subject's identity and any additional attributes for the subject and context ($Attr_S$, $Attr_C$) that would be useful for further authorization. Toward that goal, the ACS invokes two information stores: a repository of credentials and active connections used to deal with the freshness of

TABLE 8. Terminology and notation on Hidra messages.

Term	Description
S	Subject. An entity which attempts to access a resource on a CDS.
ACS	Access Control Server. Trusted third party supporting security mechanisms required for E2E security, when the resource is on a constrained device.
ANS	Authentication Server in the ACS.
CM	Credential Manager in the ACS.
R	Resource in a CDS.
K_{CM}	Secret key of the credential manager shared with the ACS.
K_S	Secret key of the subject shared with the ACS.
K_R	Secret key of the resource shared with the ACS.
$K_{S,CM}$	Secret key shared between the subject and the credential manager. Key $K_{S,CM} = K_{CM,S}$.
$K_{S,R}$	Secret key shared between the subject and the resource. Key $K_{S,R} = K_{R,S}$.
$SubK_{S,R}$	Secret key shared between the subject and the resource. Subkey $K_{S,R} = K_{R,S}$.
TGT	Ticket encrypted with $K_{S,CM}$ that allows a subject to authenticate to the credential manager.
$Resourceticket$	Ticket encrypted with K_R that allows a subject to authenticate to the resource.

TABLE 9. Detail of the content of Hidra messages.

Message	Direction	Content
HID_ANS_REQ	$S \rightarrow ANS:$	$ID_S \vert\vert ID_{CM} \vert\vert Lifetime_{TGT} \vert\vert Nonce_1$
HID_ANS_REP	$ANS \rightarrow S:$	$ID_S \vert\vert Ticket_{CM} \vert\vert \{K_{S,CM} \vert\vert Nonces_{S,CM} \vert\vert Nonce_1 \vert\vert ID_{CM}\}K_S$ *where,* $Ticket_{CM} = \{K_{S,CM} \vert\vert ID_S \vert\vert Nonces_{S,CM}\}K_{CM}$
HID_CM_REQ	$S \rightarrow ACS.CM:$	$ID_R \vert\vert Lifetime_{TR} \vert\vert Nonce_2 \vert\vert Ticket_{CM} \vert\vert AuthN_{CM}$ *where,* $Ticket_{CM} = \{K_{S,CM} \vert\vert ID_S \vert\vert Nonces_{S,CM}\}K_{CM}$ $AuthN_{CM} = \{ID_S \vert\vert Nonces_{S,CM} + i\}K_{S,CM}$
HID_CM_IND	$CM \rightarrow R:$	$ID_R \vert\vert ID_S \vert\vert Nonces_{S,R} \vert\vert Lifetime_{TR} \vert\vert K^i_{R,CM} \vert\vert AuthZ$ $\vert\vert MAC(K_R, ID_S \vert\vert Nonces_{S,R} \vert\vert Lifetime_{TR} \vert\vert K^i_{R,CM} \vert\vert AutZ)$ *where,* $AuthZ = \{Policy_R\}K_R$
HID_CM_IND_REQ	$R \rightarrow CM:$	$ID_R \vert\vert Nonce_3 \vert\vert MAC(K_R, ID_R \vert\vert Nonce_3)$
HID_CM_IND_REP	$CM \rightarrow R:$	$ID_R \vert\vert K^{i+1}_{R,CM} \vert\vert MAC(K_R, ID_R \vert\vert Nonce_3 \vert\vert K^{i+1}_{R,CM})$
HID_CM_REP	$CM \rightarrow S:$	$ID_S \vert\vert Ticket_R \vert\vert \{K_{S,R} \vert\vert Nonces_{S,R} \vert\vert Nonce_2 \vert\vert ID_R\}K_{S,CM}$ *where,* $Ticket_R = \{K_{S,R} \vert\vert ID_S \vert\vert Nonces_{S,R} \vert\vert Attrs_S \vert\vert Attr_C\}K_R$
HID_S_R_REQ	$S \rightarrow R:$	$Ticket_R \vert\vert AuthN_R \vert\vert Nonce_4$ *where,* $Ticket_R = \{K_{S,R} \vert\vert ID_S \vert\vert Nonces_{S,R} \vert\vert Attrs_S \vert\vert Attr_C\}K_R$ $AuthN_R = \{ID_S \vert\vert Nonces_{S,R} \vert\vert Subkey\}K_{S,R}$
HID_S_R_REP	$R \rightarrow S:$	$\{Nonces_{S,R} \vert\vert Subkey \vert\vert Nonce_4\}K_{S,R}$
HID_R_IND	$R \rightarrow ACS:$	$ID_R \vert\vert \{Nonce_5 \vert\vert ID_{Pol} \vert\vert Log_{S,R}\}K_R$ *where,* $Log_{S,R} = \{ID_S \vert\vert ID_{RR} \vert\vert ID_{RA} \vert\vert ID_{GE} \vert\vert ID_{Rl} \vert\vert ID_{Ob} \vert\vert Nonce_6 + i\}$
HID_R_ACK	$ACS \rightarrow R:$	$ID_R \vert\vert Nonce_5 \vert\vert MAC(K_R, ID_R \vert\vert Nonce_5)$

tickets and protocol messages, and a repository of authorization policies.

When a positive preliminary authorization occurs, before issuing the resource ticket to the subject, the credential manager in the ACS sends a message HID_CM_IND (4) to the targeted resource running on the CDS, specifying all the information it will need afterward to validate the request message from the subject. The message is renewed to include the fine-grained authorization policy with local context conditions, as detailed in the policy language described in Section IV.

To assure the freshness of this message (4), it embeds a new key value $K^i_{S,CM}$ from a previously generated one-way key chain $[K^1_{S,CM} ... K^N_{S,CM}]$. The purpose of these one-way functions on the enclosed key, $F(K^i_{S,CM}) = K^{i+1}_{S,CM}$, is to make it computationally unfeasible to calculate the inverse function using a transmitted and potentially sniffed key. When the CDS receives the message, it will compute the one-way function on the received key ($K^j_{S,CM}$) and check it against the last valid key in the previously received sequence $K^{i+1}_{S,CM}$, before validating it and storing it as $K^i_{S,CM}$. In fact, the CDS will successively compute a predefined (L) number of times on the received key ($K^j_{S,CM}$), $F_L(..F_1(K^j_{S,CM})) = K^{i+1}_{S,CM}$, to make the transmission resilient to eventual packet losses.

The messages named HID_CM_IND_REQ (4.1) and HID_CM_IND_REP (4.2), are exchanged to provide the last value ($K^N_{S,CM}$) of the key chain in a one-way function series of length N generated in the ACS every time the sequence is initiated. The sequence initiation is started by the ACS and can also be requested by the CDS as a recovery mechanism in cases where many packets are lost in the network, and the local attempts to compute consecutive one-way functions exceed a maximum limit (L).

The inclusion of the authorization policy in HID_CM_IND message, which is one of the smallest messages, also aims to achieve the minimum fragmentation of 6LowPAN IPv6 packets over IEEE 802.15.4 links by considering the remaining headers and the reduced remaining available payload. This design decision makes a difference with respect to the approaches for enclosing the policy in the ticket, which is included in a larger request message with higher probabilities of exceeding the IEEE 802.15.4 MTU and the consequent network overload. The resource in the CDS stores this information for a specified time, $Lifetime_{TR}$, and if the expected request from the subject is not received before this time elapses, the resource in the CDS deletes the corresponding information to avoid overflowing the limited capacity of the CDS. The authorization policy is codified according to Subsection IV-C and sent encrypted with the resource's secret key. This instant provisioning for every security association phase or triggered by any global context status assessed in the ACS enables novel features for the CDS such as policy updating, either by usage or revocation. The received policy is locally stored in the CDS until the security association expires or is revoked by a refresh message.

In Phase 3, the subject requests the establishment of a security association to access the resources on the CDS using a HID_S_R_REQ message (6). The subject sends the previously obtained resource ticket, which identifies the subject as an authenticated and preliminarily authorized party. The CDS validates the message using the information received from the ACS and makes a granting decision.

After a positive validation and authorization, the CDS responds to the subject with the message HID_S_R_REP (7) either accepting the key proposed in the request message (subkey) or proposing a new one. Then, the security association is established for further resource accesses. This key can subsequently be used to derive further encryption and integrity keys to be used in further resource accesses.

For each received security association request, the CDS also sends a newly defined notification message, HID_R_IND ((8) in Phase 4). This message is sent to the ACS along with the access details to support central logging and to flush any local record after the acknowledge message, namely, HID_R_ACK, is received (9).

After a security association is established, when the CDS receives any further resource access request using the established session key, the PDP in the CDS checks the policy instance to match the proper rule depending on the action requested on the resource. Then, the conditions (which are usually local) specified in the matching rule included in the policy are checked and a decision is made in the PDP. The resulting granting effect as well as correspondent reactions due to obligations are performed in the PEP in the CDS. The PEP also activates the counter and the timer to trigger rule re-evaluation by the PDP. The response to the requester will be delivered using the established session key. Additionally, and depending on the notification policy or the defined obligations, each resource access request may trigger a notification message to send activity logs to the ACS for logging, tracking, accounting and further auditing purposes. Requests will be evaluated as long as the security association has not expired in the CDS or no obligation in the CDS enforces it. Additionally, security association finalization in CDS may occur as a reaction to any revocation notification such as exceeding usage limitations or anomalous behaviour detected from tracking and accounting analyses in the ACM.

Therefore, the proposed Hidra protocol supports the ABAC authorization enforcement in two stages. On one hand, fine-grained preliminary access control is performed in the ACS, focusing on the attributes of the subject, resource and the expected action. On the other hand, local access control is performed in the CDS through the instantaneous custom policy provisioning, which behaves dynamically and avoids having to store policies permanently. This temporal policy approach covers not only the establishment of a security association but also related further resource access control during the secured session. Additionally, Hidra supports fine-grained accounting, which is a very valuable feature in the envisioned scenario.

VI. SECURITY EVALUATION

Hidra, as introduced before, is a security protocol that enables the establishment of E2E pair-wise keys for mutual authentication, dynamic multi-step authorization, tracking and accounting, while ensuring the confidentiality, integrity and non-repudiation of the messages. Nevertheless, it may also be the target of security attacks. This section presents a security analysis of the Hidra protocol under some security assumptions and threats.

First, key design aspects that make Hidra resilient against the considered security attacks are conveyed. Then, a formal security analysis of Hidra is done through a systematic method for verifying finite-state-concurrent systems. This technique is supported by validation tools so it becomes very useful for complete automatic checking to ensure the correctness of the design of Hidra at the earliest stage possible.

A. ASSUMPTIONS AND LANDSCAPE OF ATTACKS

Due to the responsibility that the management services have in the envisioned scenario, nominal access and complete traceability of the access requests, responses and access control policies are required. In the considered security model, attackers have two goals: to gain unauthorized access to resources in the CDS and to repudiate the authority. Consequently, the proposed access control model is analysed to determine its resilience against these two threats.

Regarding unauthorized access to resources and authority repudiation, the typical mechanisms involve modification of legitimate messages, fabrication of new ones or re-utilization of old ones.

In the analysed scenario, the origin of the security attacks can be third parties outside of the protocol or unauthorized subjects from the same domain. Therefore, an attacker can be aware of the supporting security configuration or possess some initial credentials shared with the ACS.

The information flow of the network is not reliable because the CDS communicates over wireless links. The attacker's profile is defined according to the Dolev-Yao threat model [49]. Consequently, the attacker can listen to all the transmitted traffic, generate new messages and replay old ones. However, perfect cryptography is assumed; therefore, an attacker cannot decipher encrypted messages without knowing the corresponding key.

Regarding the trust among the entities in the envisioned scenario, on one hand, each CDS, as well as subjects and other third parties such as the CM in the ACS, share a secret key with the ACS, which is a trusted party, as shown in Figure 10. On the other hand, none of the entities share this secret key with an attacker, so identity impersonation is not possible, because alternatives such as tampering and other physical attacks are not addressed.

1) SECURITY CONSIDERATIONS AT DESIGN TIME

Hidra security protocol specifies a message exchange protocol where some fields are included to be resilient to aforementioned attacks.

a: RESILIENCE AGAINST MESSAGE FORGERY AND MODIFICATION ATTACKS

Message forgery and modification attacks, as in Ladon, are avoided by origin authentication and the integrity of all the messages. Impersonation is not possible because only the appropriate entities own a shared key, can encrypt/decrypt the protected fields, and calculate the enclosed related MACs of the exchanged messages, as listed in Table 9.

Focusing on the messages changed in Hidra with respect to Ladon, the HID_CM_IND message encloses the authorization policy encrypted with the secret key of the CDS K_R, and the MAC to check both the integrity of the message and the proof of the claimed identity. The policy is encrypted to hide clues from potential attackers concerning the requirements of the policy to be authorized.

The HID_R_IND message encloses the details of the log encrypted with the secret key of the CDS K_R, and the response HID_R_ACK message includes an instant nonce, which is an unpredictable bit string, to correlate IND/ACK messages and a MAC to check the integrity and the origin authenticity.

b: RESILIENCE AGAINST REPLAY ATTACKS

Replay attacks are based on reusing valid but old protocol messages and tickets; therefore, message and ticket freshness must be assured.

After a TGT is issued, the freshness of the HID_CM_REQ messages is checked through the authenticator field that encloses an incremental nonce initiated randomly by the ACS during TGT retrieval. Therefore, the CM in the ACS does not accept messages with an enclosed $Nonce_{S,CM} + i$ value that does not conform to the expected incremental sequence.

After a resource ticket is issued, the freshness of the HID_S_R_REQ messages, which enclose the ticket, is assured because such tickets are valid for only one security association, and it is validated through the $Nonce_{S,R}$ value set at the reception of the HID_CM_IND message by the CDS. If the security association is finalized and a subject aims to establish another one, a new resource ticket is required.

Related to the HID_CM_IND messages, freshness is checked through the enclosed one-way key $K^i_{S,CM}$, which is based on the properties of one-way functions as explained in the Section V. The CDS then, will compute successively the one-way function (F) over the received one-way key ($K^j_{S,CM}$) and will check it against the last valid key in the previously received sequence, $F(..F(K^j_{S,CM})) = K^{i+1}_{S,CM}$, before validating it and storing it as $K^i_{S,CM}$.

Finally, related to the HID_R_IND/_ACK message exchange, freshness is achieved by means of a nonce value by which the request message can be matched with its corresponding response ($Nonce_5$), but different HID_R_IND messages among them are distinguished through another incremental $Nonce_6 + i$ value enclosed in the log fields.

c: RESILIENCE AGAINST REPUDIATION ATTACKS

A user's authority may be compromised by obfuscating information regarding the identity of the subject or perturbing the relationship between different protocol message exchanges.

User authority is achieved by enclosing the subject identifier in the nominally issued tickets. Therefore, because impersonation is not possible, the tickets and the fact that they cannot be reused assures the unequivocal identification and authority of the subject. In fact, this aspect is closely related to the origin authentication.

Regarding the tracking of resource accesses, the HID_R_IND message logs each resource request with details about who has accessed which resource to perform which action, when, and under which policy rule enforced in the ACS, including security association establishment as well as any further resource access. At the reception of the HID_R_ACK message, the CDS can flush local records, relying on the ACS to manage their storage and treatment. In fact, the central collection in the ACS, followed by normalization, correlation and analysis processes, enable both dynamic unified policy refinement and potential deviation detection. An immediate reaction to any detected deviation is enabled by transmitting the HID_CM_IND message enclosing a new custom policy instance that the ACS will enforce. Therefore, the newly introduced messages in Hidra enable not only traceability but also reactivity.

B. FORMAL VALIDATION

Designing security protocols is not straightforward and therefore, attacks conducted by intruders derived from the slightest misconception at design time can be hardly detected without the support of automated tools. There are several tools to complete automatic, robust, expressive and rigorous analysis of security protocols by either searching flaws or establishing their correctness, and AVISPA [51] has been selected in this proposal by its scalability and broad adoption.

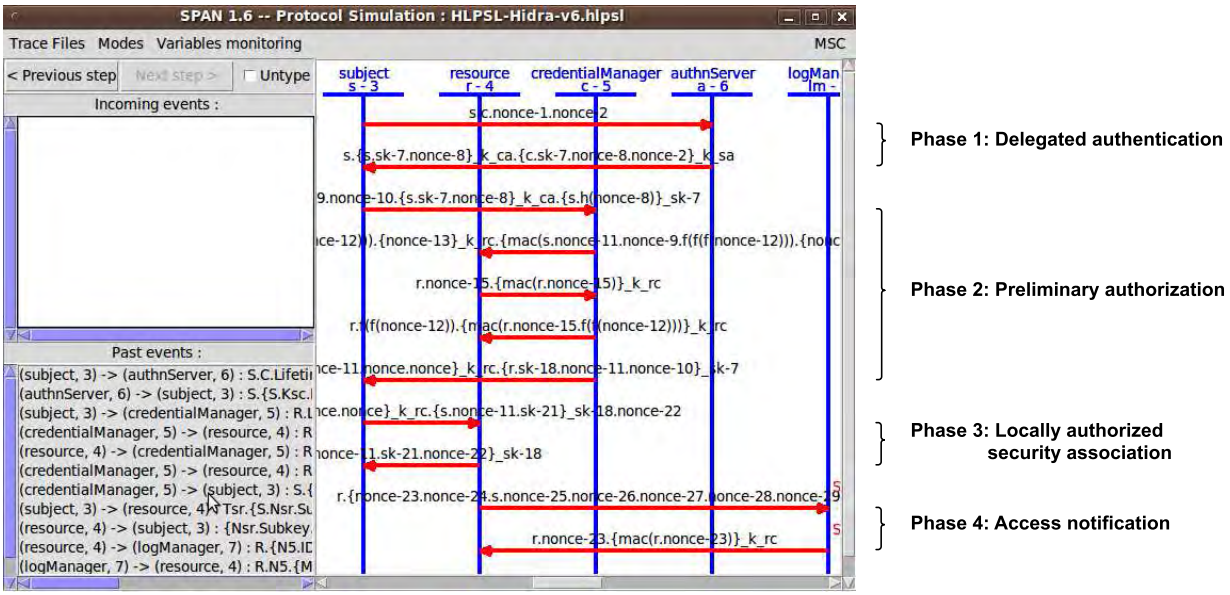


FIGURE 11. Hidra protocol simulation snapshot through SPAN tool [50] in AVISPA, including Hidra phases.

In fact, AVISPA is a software tool that supports the definition of a model of the security protocol to be validated, the verification of the correctness of the developed model and the proper formal analysis of the security protocol.

On one hand, the model of Hidra security protocol is defined through a modular and expressive formal language for specifying security protocols and properties, called High-Level Protocol Specification Language (HLPSL) [52]. HLPSL is a role-based language to define agent profiles, their states, and transitions, composing several sessions as a set of agent instances, where an intruder may try to impersonate a legitimate user. For the specification to be complete, the security goals to be achieved by the protocol must be specified within the model. In fact, the security verification checks whether these goals are met or not.

On the other hand, the rigorous specification verification and security validation checks can be performed through four different complementary modular back-ends that implement the automatic analysis techniques ranging from protocol falsification to abstraction-based verification methods for infinite number of sessions. These automatic checks are easily feasible because AVISPA provides an intermediate tool to translate human readable HLPSL model to machine processing language called Intermediate Format (IF) model. Concretely, the available checks over the IF version of the model are performed by On-The-Fly-Model-Checker (OFMC) [53], the Constraint-Logic-based Attack Searcher (CL-AtSe) [54], SAT-based Model-Checker (SATMC) [55] and the Tree Automata based Automatic Approximations for the Analysis of Security Protocols (TA4SP) [56].

The mentioned back-ends assume perfect cryptography and that the protocol messages are exchanged over a network that is under the control of a Dolev-Yao intruder [49]. In the worst case, a potential internal attacker belonging to the domain owns a valid secret key shared with the ACS.

1) HIDRA SECURITY PROTOCOL HLPSL MODEL

The Hidra security model definition is not done starting from scratch. The AVISPA tool is supported by a broad library of models of most IETF security protocols. In fact, starting from Kerberos HLPSL model (mutual strong authentication), the HLPSL model of Ladon, which avoids clock synchronization and provides basic authorization support, was defined and the formal security validation of the protocol was presented in [48]. Therefore, Hidra extends this HLPSL model including the expressive authorization and accounting just as detailed in Section V.

Being HLPSL based on roles, five Hidra related basic roles are defined: the Authentication Server (ANS) in the ACS, defined for clarity as *authnServer (A)*, the Credential Manager (CM) in the ACS, defined as *credentialManager (C)*, a communication peer acting as *resource (R)* in a CDS, a communication peer acting as *subject (S)* and a log reception server in the ACS, defined for clarity as *logManager (LM)*. Each of these roles define the proper known parameters, their initial state, the transitions and the contribution to the security goals.

Then, a *session* is defined as a composition of an instance of each of the five basic roles and the channels to send and receive the messages defined by Hidra. Additionally, the top-level role, called *environment*, defines global variables and the intruder's knowledge and combines several sessions representing attacks based on the behaviour of the intruder.

Finally, the security goals to be validated are specified in the goal section and defined through some goal facts augmenting the transitions in each involved basic

role. AVISPA provides useful and concise macros for the two most frequently used security goals, authentication and secrecy (*secrecy_of*). Authentication can be weak (*weak_authentication_on*) or strong (*authentication_on*), and the latter extends the former precluding replay attacks. Therefore, Hidra related security goals have been defined as follows.

Authentication: the *authentication_on* security goal is specified through two goal facts, namely *witness* and *request*. They are specified in every data exchange, so every recipient authenticates the sender in each message. This way mutual authentication is enforced, and since strong authentication is specified it is also verified that no replay attacks can take place.

Data confidentiality: this property is specified applying the *secrecy_of* security goal to all secret keys and secret values exchanged by peers. Data integrity: this property is specified applying the *secrecy_of* security goal to the secret keys used to calculate the MAC codes defined by Hidra over the protected data sets.

Authorization: proper enforcement is assumed based on the confidentiality and integrity of the authorization policy. Therefore, in the roles of resource and ACS both are specified, the *secrecy_of* the provisioned policy as well as the integrity guaranteed by the MAC code in the HID_CM_IND message, which is specified through the *secrecy_of* the key used to calculate such MAC code.

Freshness: Hidra specifies several nonces for this purpose and this property is specified applying the *secrecy_of* security goal to them. Additionally, previously specified strong authentication in AVISPA includes protection against replay attacks.

Accounting: the *secrecy_of* the log data sent by the resource is specified in both roles resource and logManager in the ACS, and mutual strong authentication is also specified.

It is worthy of note that by cause of HLPSL limitations not all the properties of Hidra are modelled as expected at design time. First, the increment of the *i* value on nonces included in some messages is not possible because HLPSL does not support arbitrary arithmetic operators. Alternatively, a hash function is applied as a shared well-known function over the same nonce. And second, current version of HLPSL does not support time-related events such as time-outs, so lifetime expirations are not specified. Since perfect cryptography is assumed and cryptanalysis attacks are not considered, it has been checked that this issue has no effect in the protocol behaviour and neither the fulfilment of the security goals.

Prior to security validation, Hidra HLPSL model is verified both syntactically and semantically, as well as checking that it represents properly the Hidra protocol behaviour. First, the HLPSL format file is successfully loaded and converted by AVISPA to a machine language format IF as explained before. Then, SPAN [50] security protocol simulation tool in AVISPA is executed and Figure 11 shows the resulting message sequence chart, which is very useful for visual checking of protocol semantic and behaviour.

2) SECURITY VALIDATION

The security validation consists of the assessment that the previously modelled Hidra security protocol meets the stated security goals. In fact, it is performed through the OFMC and CL-AtSe back-end modules, which verify whether the defined security goals are satisfied for a bounded number of sessions, as specified in the *environment* role. On one hand, OFMC uses the *lazy intruder* technique to represent all the possible messages an intruder could generate trying to attack the protocol. And on the other, CL-AtSe, which is based on *constraint solving technique*, implements powerful methods for validation through simplification and redundancy avoiding.

The TA4SP back-end is not used because it does not support the type of compound variable *set* used in the Hidra HLSPL model, and consequently the provided results are *INCONCLUSIVE*. Additionally, the SATMC back-end results have been discarded since it warns that the intruder can not generate fresh terms and therefore, such attacks are not included and provided results are less accurate.

For the validation, first, the Hidra protocol is checked implementing a single session and allowing all the roles to be played by legitimate agents (use-case 1). Table 10 shows that such a session is instantiated involving the five defined roles *(a,c,s,r* and *lm)* denoting authnServer, credentialManager, subject, resource and logManager respectively. Additionally, the involved shared keys between peers are specified *(K_sa, K_rc* and *K_ca)*, and finally three hash functions are declared *(h, f* and *mac)*.

Then the session is modified to test the use-cases in which the intruder owning a valid secret key shared with the ACS would try to impersonate each of the legitimate agents: the subject (use-case 2), the CDS resource (use-case 3), the ANS (use-case 4) and the CM (use-case 5). Note that the intruder does not have any knowledge of the secret keys owned by the mentioned agents.

Additionally, another use-case is tested where two parallel sessions are executed, and in one of the sessions, one of the legitimate agents (the subject) is playing a role (CM, use-case 6) for which is not intended to. Finally, the use-case of two parallel sessions are executed, and in one of the sessions, the intruder is playing the role of the subject (use-case 7).

Table 10 shows the configurations of the sessions used to execute the use-cases described above. After running all the use-cases, the results are all *SAFE*, and it can be stated that the security validation of Hidra is successful with no attacks detected by the AVISPA tool.

Therefore, it can be concluded that Hidra meets the security goals defined in AVISPA, so Hidra is a secure protocol for the mutual strong authentication, the fine-grained authorization, the confidentiality and integrity of secret data and the accounting proposed by current access control model.

VII. PERFORMANCE EVALUATION

In this section, the performance of the access control model for E2E security in CDSs is evaluated. An analytical

TABLE 10. Security validation results considering several use cases where intruder is challenging the protocol and performed by two AVISPA back-end engines, which are OFMC and CL-AtSe.

Use-case	Use cases based on impersonation trials	Session	OFMC	CL-AtSe
1	Legitimate agents with no impersonation	session(a,c,s,r,lm,k_sa,k_rc,k_ca,h,f,mac)	SAFE	SAFE
2	Intruder trying to impersonate the subject	session(a,c,i,r,lm,k_ia,k_rc,k_ca,h,f,mac)	SAFE	SAFE
3	Intruder trying to impersonate the resource in CDS	session(a,c,s,i,lm,k_sa,k_rc,k_ca,h,f,mac)	SAFE	SAFE
4	Intruder trying to impersonate the Authentication Server	session(i,c,s,r,lm,k_sa,k_rc,k_ca,h,f,mac)	SAFE	SAFE
5	Intruder trying to impersonate the Credential Manager	session(a,i,s,r,lm,k_sa,k_rc,k_ca,h,f,mac)	SAFE	SAFE
6	Parallel session and the subject trying to impersonate the Credential Manager	session(a,c,s,r,lm,k_sa,k_rc,k_ca,h,f,mac) session(a,s,s,r,lm,k_sa,k_rc,k_ca,h,f,mac)	SAFE	SAFE
7	Parallel sessions and the intruder trying to impersonate the subject in one of them	session(a,c,s,r,lm,k_sa,k_rc,k_ca,h,f,mac) session(a,c,i,r,lm,k_ia,k_rc,k_ca,h,f,mac)	SAFE	SAFE

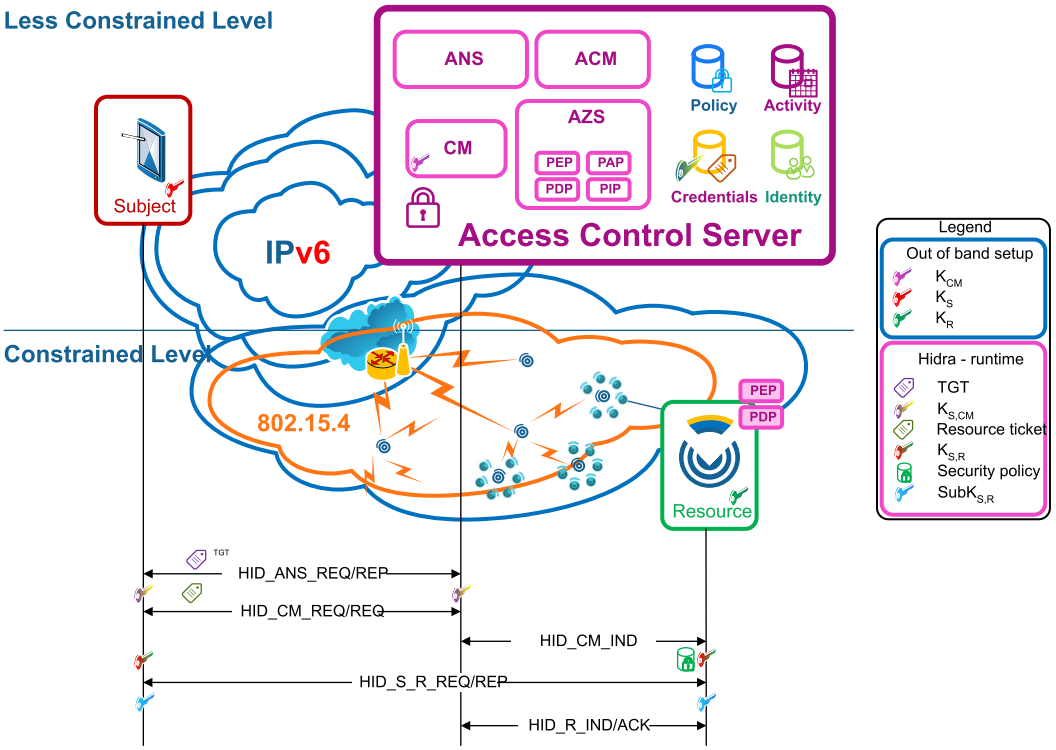


FIGURE 12. Performance evaluation scenario.

performance model covering the computation of the crucial performance parameters of such resource constrained sensing environments is described. First, the reference scenario and some assumptions are presented, and after the crucial performance parameters have been identified, their computation is described. Then, an analysis of the analytical model is presented, describing and discussing the evaluation results.

The overall goal is to demonstrate the suitability of the designed access control model for CDSs in the envisioned scenarios.

A. PERFORMANCE MODELLING

To conduct a performance evaluation of the proposed access control model for E2E security in CDSs, the analytical performance model focuses on three critical parameters: 1) the response time of the access control model to establish an authorized E2E secure session, 2) the energy cost of this model for the protected CDS running on finite batteries, and 3) the model's impact on the local storage on the CDS. The response time needs to be below an accepted value if the proposal is to be useful, and the energy consumption and local storage, due to the nature of the CDSs and their constraints on resources, cannot exceed rational and proportional limits.

1) REFERENCE SCENARIO AND ASSUMPTIONS

The reference scenario for the performance evaluation is graphically depicted in Figure 12. In this scenario, a subject is connected to the Internet and establishes an E2E connection with a resource running on a CDS in an IEEE 802.15.4 network. A 6LoWPAN router (in orange) acts as the LowPAN coordinator and connects a beacon-enabled cluster-tree structure to the Internet. The IEEE 802.15.4 network is 3-hops deep, which is considered significantly large for validation purposes. The PAN router coordinator has three child coordinators, which each have another three child coordinators, each of which controls a cluster of six leaf nodes

where the CDSs expose resources as management services. Therefore, 54 ($3x3x6$) CDSs are integrated into the network, but the details of the branches have been omitted from Figure 12 for clarity.

In this performance evaluation, only the interactions with the CDSs are considered because they experience the highest impact due to their scarce resources. Therefore, the delegated authentication and the TGT obtaining steps are omitted, and it is assumed that a requesting subject has previously obtained a TGT before trying to establish an E2E security association with a CDS.

Regarding the computation to establish the E2E secure session time, only the delays of the actors involved in the IEEE 802.15.4 segment have been considered; Internet delays are ignored because they are commonly some orders of magnitude smaller.

Regarding message transmission, queueing theory forms the foundation of the model; each transmission generates a new job in the queue. For the queueing model analysis of the average service and queue waiting times, only the jobs corresponding to the Hidra protocol exchanges have been considered.

With respect to the implementation of the Hidra protocol, a specific binary codification is used for the exchanged messages, as in Ladon, to assure its feasibility even in severely constrained devices (C0).

Finally, and related to the encryption, *cipher-text stealing* techniques are adopted, and therefore the resulting ciphertext has the same length as the corresponding plain text. The next three sections model the time, the energy cost and the storage impact of establishing an E2E security association using Hidra.

2) COMPUTATION OF THE E2E RESPONSE TIME IN THE SECURE SESSION ESTABLISHMENT

During the establishment of a security association, five messages are exchanged, as detailed in Section V. This response time includes the steps where the subject obtains the service ticket, the notification that the ticket is granted, policy provisioning in the CDS by the ACS, and the security association request and response between the subject and the CDS.

Concerning the response time computation, four contributions are considered in each message transmission: the time required to generate the message in the origin, the network transmission time, the queue wait time in the destination and the time required to process the received message. The computation of each of these delay contributions is specified below.

a: COMPUTATION OF SERVICE TIMES

The service time comprises the generation of a message to be transmitted or the processing of a received message. Executing the cryptographic operations is the most significant contributor to this performance indicator. The remaining operations, such as concatenating fields, reading the received policy, pattern matching comparisons, determining policy and enforcing the granting effect, are minuscule compared with the cryptographic operations. In fact, for the service time computation based on cryptographic functions, two different constant bit rates (in bits/s) are adopted in each entity for encryption and MAC functions, denoted by $TCRYP_{entity}$ and $TMAC_{entity}$, which depend on the hardware implementation. Specifically, in CDSs, where resources are very scarce, these two rates $TCRYP_R$ and $TMAC_R$ are crucial and differentiating characteristics, and are usually lower than those of the subject and the ACS.

Therefore, the lengths of the fields enclosed in the messages directly impact the service computation time. Table 11 summarizes the lengths of messages in the Hidra protocol, and the number of bytes that are subject to cryptographic operations in each interacting entity, denoted by $|CRYP(Message^i)|$. A (MAC) label is used to distinguish MAC computation from encryption. To compute these lengths, the assumptions involve 16 bytes for cryptographic keys, 8 bytes per nonce and two bytes for subject and resource identifiers.

Equation 1 shows the service time calculation as encryption time, related to the message i in the entity X, which is computed based on the aforementioned two terms, namely, $|CRYP(Message^i)|$ to denote the number of bytes that are subject to cryptographic operations and $TCRYP_X$ to denote the constant bit rate of encryption functions.

$$S_{Xi} = \frac{|CRYP(Message^i)|}{TCRYP_X} \tag{1}$$

b: COMPUTATION OF NETWORK TRANSMISSION TIME

When computing the transmission time, the considered network is the one depicted in Figure 12. To compute the time required to transmit a message, the main components are the backoff time (D_{BOT}) and the data transmission time over the wireless links (D_{Tx}), and it is computed for each of the three hops of the evaluation scenario (*D1, D2, D3*). Equation 2 shows the calculation of the transmission time in each of the three hops:

$$D_l = D^l_{BOT} + D^l_{Tx} \quad for \;\; l = 1, 2, 3. \tag{2}$$

Therefore, the total transmission time is calculated as follows:

$$E[t_n] = D_1 + D_2 + D_3 \tag{3}$$

Other delays that occur, such as the turnaround time from transceiver transmissions to receiving, beacon transmission time or inter-frame space have been removed because their impact is insignificant.

Regarding the back-off time (D^l_{BOT}), before the transmission, the entities first check whether the channel is free, and if it is in use, they wait for a time that increases over the iterations. To compute the back-off time, the model proposed in [57] has been followed, where both the density of the sensor network and the length of the messages directly impact the

TABLE 11. Lengths of Hidra protocol messages and number of bytes over which each entity must perform cryptographic operations. The computed lengths of four different sample instances.

Message type	Length (bytes)	Bytes Subject to cryptographic operations			
		Subject	ANS	CM	Resource
HID_ACS.ASN_REQ	15	-	-	-	-
HID_ACS.ASN_REP	62	34	60	-	-
HID_CM_REQ	47	10	-	36	-
HID_CM_IND	35 / 40 / 42 / 65	-	-	2 / 7 / 9 / 32 29 / 34 / 36 / 59 (MAC)	2 / 7 / 9 / 32 29 / 34 / 36 / 59 (MAC)
HID_CM_IND REQ	14	-	-	10 (MAC)	10 (MAC)
HID_CM_IND REP	22	-	-	26 (MAC)	26 (MAC)
HID_CM_REP	62	34	-	60	-
HID_S_R_REQ	60	26	-	-	52
HID_S_R_REP	32	32	-	-	32
HID_R_IND	26	-	-	24	24
HID_R_ACK	14	-	-	10 (MAC)	10 (MAC)

transmission time. Concerning network density, the number of hops, number of nodes, their roles as subjects or resources and the mean job generation rate have been considered as the key parameters.

In addition to the lengths of the messages defined in Hidra and listed in Table 11, the headers of the lower layers in the IEEE 802.15.4 network are computed. Specifically, UDP (8 bytes), IPv6/6LoWPAN (19 bytes), and IEEE 802.15.4 MAC/PHY (15 bytes) are considered. The same considerations of message length apply to compute the time needed to transmit each message over the air: a constant transmission bit rate for all the IEEE 802.15.4 (B) links is assumed.

c: COMPUTATION OF QUEUE WAITING TIMES

Concerning the queueing analysis, the three entities (subject (S), ACS and resource (R)) are modelled as M/G/1 queues. In fact, three of them process messages of variable length and, therefore, the service time can be represented with a general distribution. It is assumed that there are $N_S = 1$ subjects generating requests according to a Poisson distribution with a mean ratio of λ_0 to N_R resources running in the leaf nodes of the network scenario.

First, the mean arrival rates of jobs (λ) for each type of entity is calculated, considering N as the length of the one-way key chain used to assert the authenticity of HID_CM_IND messages.

Then, for each of the three entities (S, ACS, and R), the average service time ($\overline{X}$) is calculated as the average service time of the involved messages weighted by the probability that they will occur, as represented in Equations 4 to 6 respectively.

$$\overline{X_S} = \frac{S_{S1} + S_{S2} + S_{S3} + S_{S4}}{4} \tag{4}$$

where S_{Si} are the service times related to the generation or processing of HID_CM_REQ/_REP and HID_S_R_REQ/_REP messages, respectively.

$$\overline{X_{ACS}} = \frac{S_{ACS1} + S_{ACS2} + S_{ACS3} + \frac{1}{N}(S_{ACS4} + S_{ACS5})}{3 + \frac{2}{N}} \tag{5}$$

where S_{ACSi} are the service times related to the generation or processing of HID_CM_REQ/_REP, HID_CM_IND, and HID_CM_IND_REQ/_REP messages, respectively.

$$\overline{X_R} = \frac{S_{R1} + S_{R2} + S_{R3} + \frac{1}{N}(S_{R4} + S_{R5})}{3 + \frac{2}{N}} \tag{6}$$

where S_{Ri} are the service times related to the generation or processing of HID_CM_IND, HID_S_R_REQ/_REP and HID_CM_IND_REQ/_REP messages, respectively.

Finally, considering the resource utilisation as $\rho = \lambda\overline{X}$, the random variable for the service time of job i as X_i, and the second moment of the service time as $\overline{X^2} = E[X_i^2]$, the average waiting time in the queue for each entity is computed according to the Pollaczek-Khinchin mean formula [58], as shown in Equation 7.

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} \tag{7}$$

3) COMPUTATION OF THE ENERGY COST OF THE HIDRA PROTOCOL

To establish this security association five messages are exchanged. Additionally, two more messages are exchanged to send a notification of the subject's request and the policy identifier for the corresponding granting effect, as detailed in Section V.

Regarding the energy cost computation, the energy consumed by communication (either transmission or reception) of bits over the air and the computation of cryptographic operations are considered to consume the most energy [59].

Equation 8 shows the calculation of the communications energy consumption ε_{Cx} during message reception in CDS X, based on the energy consumed to receive a message i (ε_{Rx}). Here, P_{Rx} denotes a constant reception power consumption and B denotes a constant wireless link data bit rate in the LoWPAN network.

$$\varepsilon_{Cx} = \varepsilon_{Rx} = \frac{|Message^i|}{B} P_{Rx} \tag{8}$$

Equation 9 shows the calculation of the communications energy consumption during message transmission in CDS X,

which includes also the energy consumed during the back-off processes (ε_{BOT}) that is computed following the model in [57].

$$\varepsilon_{Cx} = \varepsilon_{BOT} + \varepsilon_{Tx} \tag{9}$$

Equation 10 shows the calculation of the energy consumed to transmit a message (ε_{Tx}), where P_{Tx} denotes a constant transmission power consumption

$$\varepsilon_{Tx} = \frac{|Message^i|}{B} P_{Tx} \tag{10}$$

Regarding the energy consumed to execute cryptographic operations, Equation 11 shows the calculation corresponding to the message i in the entity X. The calculation is based on the lengths of the fields object of cryptographic operations ($|CRYP(Message^i)|$), and the cryptographic operation rate $TCRYP_X$, assuming a constant instantaneous power consumption (P_C) for the computation.

$$\varepsilon_{Sxi} = \frac{|CRYP(Message^i)|}{TCRYP_X} P_C \tag{11}$$

Specifically, considering ε_{Cxi} and ε_{Sxi} as the energy consumed for the communication and the processing of the $message^i$ in the resource X, and focusing on the received messages, namely (1) HID_CM_IND, (2) HID_S_R_REQ, (5) HID_R_ACK and (7) HID_CM_IND_REP, and the transmitted ones, (3) HID_S_R_REP, (4) HID_R_IND and (6) HID_CM_IND_REQ, Equation 12 shows the calculation of the total energy consumption in the CDS X to establish a secure association.

$$\begin{aligned}\varepsilon_x = {} & (\varepsilon_{Cx1} + \varepsilon_{Sx1}) + (\varepsilon_{Cx2} + \varepsilon_{Sx2}) \\ & + (\varepsilon_{Sx3} + \varepsilon_{Cx3}) + (\varepsilon_{Sx4} + \varepsilon_{Cx4}) + (\varepsilon_{Cx5} + \varepsilon_{Sx5}) \\ & + \frac{1}{N}(\varepsilon_{Sx6} + \varepsilon_{Cx6} + \varepsilon_{Cx7} + \varepsilon_{Sx7})\end{aligned} \tag{12}$$

where N denotes the length of the one-way key chain used to assert the authenticity of HIC_CM_IND messages.

4) PERMANENT AND INSTANT STORAGE COMPUTATION

In this subsection, the increase of permanent storage and the memory footprint generated by the proposed access control model are considered. Specifically, the storage and memory footprint for the instant provisioning of a received access control policy and the code needed to parse it, as well as the data blocks related to the messages of the Hidra protocol, are considered.

On the one hand, regarding the permanent storage for the mentioned entities, the symmetric key shared with the ACS, the access control policy, the PDM specification file, the code to parse the received policy and the code to run the Hidra protocol are considered the additional minimum entities that should be permanently stored in a CDS along with some original resources and sensing applications.

When the PDP and the PEP are implemented as totally decoupled from the sensing applications they also need to be stored permanently, and must be considered, but this is not the usual case in severely constrained devices (C0) where enforcement is tightly coupled with the sensing application.

On the other hand, regarding the memory footprint, the data required for the Hidra protocol message exchange are considered: namely, the key shared with the ACS, the session keys shared with the subject, the subject identity, the different nonces, the lifetime of the security association, and the log of each access provisionally wrapped until the acknowledgement of its reception from the ACS has been received in the CDS. However, some of these values are loaded and erased during the reception, processing and transmission of relatively consecutive messages.

B. PERFORMANCE ANALYSIS

In this section, the analytical models developed in the previous sections are used to analyse the mean response time and energy consumption to establish a secure session, as well as the impact of the expressiveness of the policy on local storage.

No similar feasible approach has been validated to enforce local context-based access control in severely constrained devices (C0), but a similar validation has been conducted on 32-bit CPU devices (C2).

1) ANALYSIS SCENARIO

In the Hidra protocol analysis scenario, due to the elasticity of the policy provisioned during the establishment of a security association, four sample instances have been selected and described previously in Section IV: IS_1, which is equivalent to the authorized-if-authenticated or ACL model implemented in many severely constrained devices (C0); IS_2, which is comparable to the existing DcapBAC approaches based on PKC and has been validated in not-so-constrained devices (C2), because both check local conditions based on attributes; IS_3, which adds reactive obligations; and IS_4, which includes several rules for different resources and supports periodical re-evaluation during access. In fact, IS_2 enables richer expressions as conditions than previous solutions, and both IS_3 and IS_4 go far beyond the existing solutions in the expressiveness and granularity of access control enforcement.

In fact, considering the direct impact of the length of the messages in the computation of the response time, energy consumption and storage, it is worth noting the proportionality of the policy lengths of the four sample instances. According to Table 11, the lengths of the messages exchanged during the authentication and authorization protocol range from 15 to 63 bytes. Enclosing the policy in the HID_CM_IND message, which is one of the smallest, implies the minimum fragmentation of 6LowPAN IPv6 packets over IEEE 802.15.4 links as explained in Section V, considering the rest of the headers, and the reduced remaining available payload in bytes. This design decision makes a difference with respect to the approaches for enclosing the policy in the ticket, which is included in larger request messages. Therefore, there is a proportional and optimized impact in the length of a message from injecting the compressed policy into the shortest one.

Additionally, Table 7 shows that, comparatively, the policy length of the four exemplary instances of samples (up to 32 bytes in the most expressive sample) have much less impact in the message overload than the proposals analysed in Section II (up to 391 bytes with CBOR, and 554 bytes with JSON), that have been validated in not-so-constrained devices (C2).

In the envisioned scenario, the CDS is accessed by the subject to perform tasks such as personalization, parametrization, updating, upgrading, maintenance, and so on. These types of interactions do not occur often. As the most exigent scenario, we can consider one that requests access every hour to tune the user experience in an application where the users change hourly. Less exigent scenarios could be considered, such as the parametrization of the CDS at every manufacturing shift change or even more-relaxed scenarios related to preventive maintenance task performed at the end of every day, week, or month. Moreover, although occasional access rates are the most typical, this proposal evaluates the impact in the worst cases.

Finally, to obtain numerical results, specific values for the parametrisation of the Hidra protocol are considered as shown in Table 12. The lengths of the messages were computed after considering the changes introduced by Hidra on Ladon [47]. Additionally, both proper cryptographic operation rates in the different entities and network load profiles such as the network transmission rate, the number of resources, the simultaneously requesting subjects and the mean rate of the Poisson distribution of access requests have been considered.

The encryption and MAC computation rates of the CDSs are significantly lower than those of non-constrained devices such as the ACS and the subject; therefore, their impact in the overall performance of the protocol is considerably higher. In addition, depending on the specific hardware platform and the cryptographic algorithm implementation, the rates in the CDSs may vary notably. Therefore, a broad range of rates are considered in the impact assessment [60]. Because the ACS and the subject are not constrained devices, the encryption and MAC computation rates of these entities have little impact on the performance of the evaluated protocols. Consequently, the values based on the benchmark results gathered in [61] have been used here. Overall, the considered rates in all the devices are very conservative and affordable by most platforms even in the worst case. Additionally, because technology tends to evolve, these rates would rise, providing a positive impact.

Table 13 summarises the different instantaneous power consumption values used for the analysis. Note that these power consumption values correspond to a MEMSIC TelosB mote (TPR2420CA) powered with a 3 V power supply [62].

2) OBTAINED RESULTS AND DISCUSSION

First, the impact of the encryption and MAC computation rates of the CDSs on the performance of Hidra with four instances of samples is assessed. For this purpose, Figures 13a and 13b show the average response time for establishing a secure session using Hidra with four different sample instances.

TABLE 12. Parameters used to define the operation of the Hidra protocol, the performance of the interacting entities and the network load profiles.

Parameter	Description	Value
N	Length of the one-way key chain	100
$TCRYP_S$	subject encryption rate	50 Mbps
$TCRYP_R$	resource encryption rate	50 to 300 Kbps
$TMAC_R$	resource MAC computation rate	250 to 500 Kbps
$TCRYP_{ACS}$	ACS encryption rate	100 Mbps
$TCRYP_{ACS}$	ACS MAC computation rate	200 Mbps
λ_0	Mean job generation rate	1 request/hour
N_R	Number of resources	54
N_S	Number of subjects	1
B	Wireless link data bit rate	250Kbps

TABLE 13. Parameters used to characterize the energy consumption of sensor nodes.

Name	Description	Value
P_{RX}	Power consumption in reception mode	74.4 mW
P_{TX}	Power consumption in transmission mode (0dBm)	65.7 mW
P_C	Power consumption in cryptographic processing mode	5.4 mW

These two figures show that the maximum response time, even in the worst case, with the lowest encryption and MAC computation rate, is below 96 ms. This value is extremely low and is definitely acceptable, considering that the maximum acceptable delay in interactive data transactions specified by the ITU-T Y.1541 recommendation (Network performance objectives for IP-based services) is 400 ms [63]. It must be noted that aforementioned E2E delay thresholds consider the time elapsed from when a packet leaves the subject entity until it reaches the destination entity, while the images in Figure 13 measure the time needed to fully establish a secure connection, including all the message exchanges that establishment implies. In addition, the impact on the response time is also acceptable even according to the more restrictive value of 100 ms assessed by Stallings as a good quality for E2E response time.

Finally, one related comparable response time value has been found in the literature, although there is no mention of additional performance indicators such as energy consumption. At the C2 level, [36] reveals that the comparable measured response time for the authorization response starting when the subject sends the request is 480.96 ms. The response time of Hidra is much lower and, therefore, much better.

These two figures also show that improving the encryption rate has a greater impact on the reduction of the response time than does improving the MAC computation rate, (also shown in the comparison in Figure 14).

Attending to the impact on the energy consumption considering different cryptography and MAC computation rates, Figures 15a and 15b show the energy consumption introduced from establishing a secure session using Hidra with four different sample instances. These figures show that energy consumption remains under 1.14 mJ even in the worst case, which is both low and acceptable considering that these

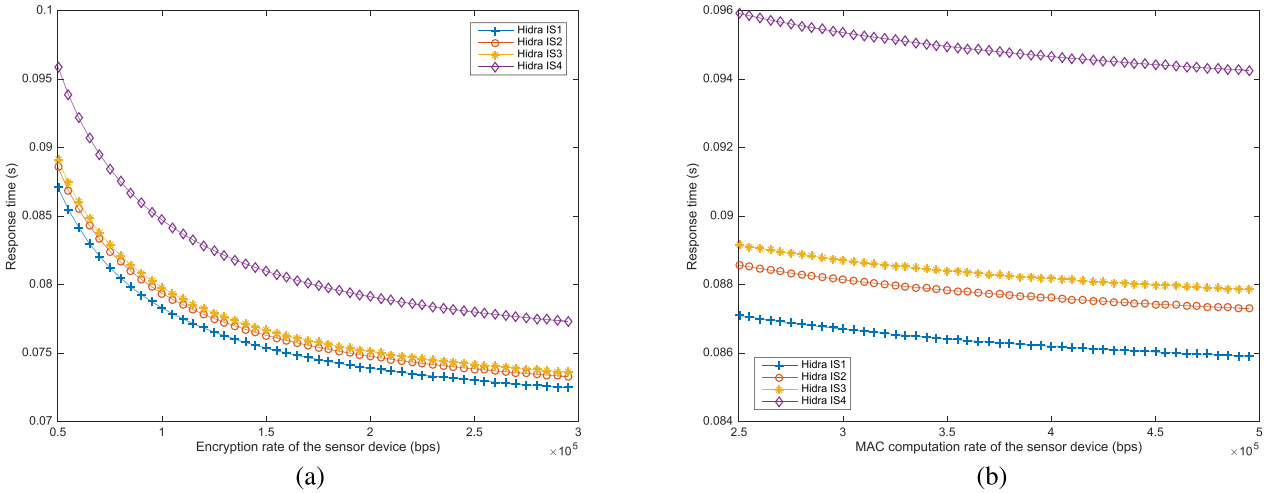


FIGURE 13. Impact of the cryptographic computation rate on the average response time from establishment of a secure session using Hidra with four different sample instances, considering the worst case of once per hour as the user request rate. (a) Impact of the encryption rate on the average response time from establishment of a secure session using Hidra with four different sample instances. (b) Impact of the MAC computation rate on the average response time from establishment of a secure session using Hidra with four different sample instances.

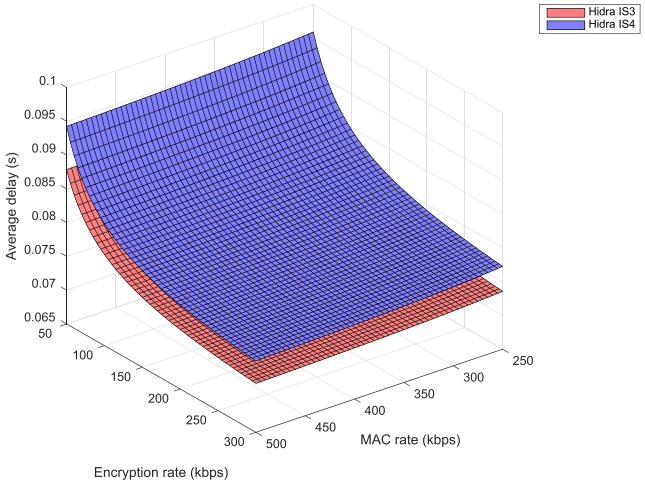


FIGURE 14. Response time comparison on Hidra with samples 3 and 4.

devices are powered by batteries with capacities of approximately 5940 J (the energy required is approximately one-millionth part of the total battery energy).

To assess the proportionality of such energy consumption values (mJ), considering a maintenance use case in which a complete system status retrieval or where a firmware file must be uploaded with an average size of about 50 KB, the energy consumption with the best networking performance approaches 150 mJ. In this use case Hidra implies a 1% additional consumption, which is a low and worthwhile impact.

Figures 15a and 15b also show that improving the encryption rate has a greater impact on the energy consumption than does improving the MAC computation rate (also shown in the comparison in Figure 16).

Comparatively, it can also be pointed out that the evaluated rates for executing cryptographic operations affect the E2E response time more noticeably than they affect energy consumption. This confirms that energy consumption in sensor devices is dominated primarily by transmission and reception operations, because the instantaneous power consumption associated with using the radio transceiver is at least one order of magnitude higher than power consumption corresponding to CPU use.

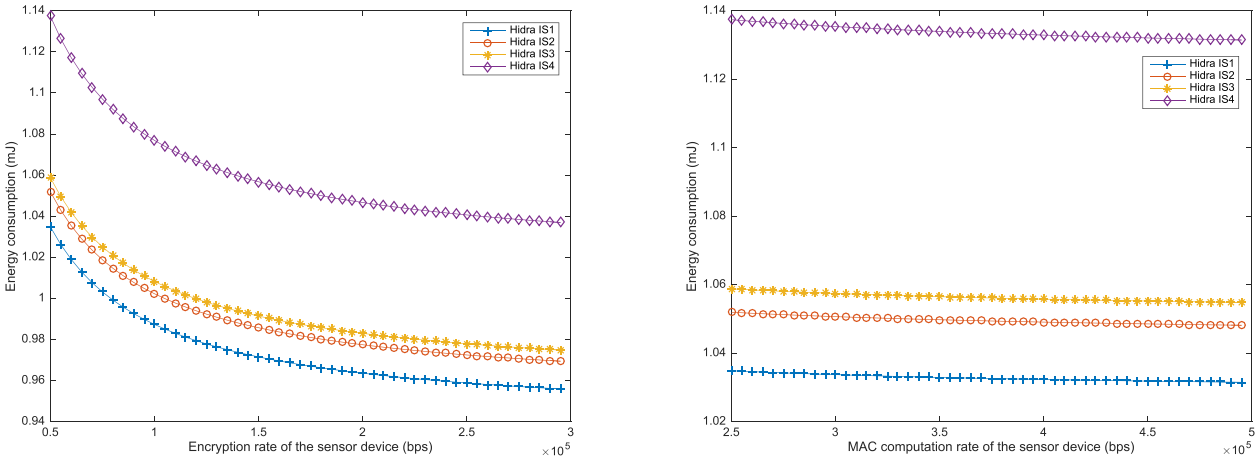
Therefore, Figures 17a and 17b show the energy consumption introduced from establishing a secure session using Hidra with four different sample instances at different session establishment rates.

On the other hand, Figure 18 shows that the access request rate has a greater impact on the energy consumption than does improving the encryption rate. This is because energy consumption in sensor devices is dominated primarily by transmission and reception operations.

Finally, from the storage point of view, at the CDS, the amount of RAM memory is the most limiting aspect, compared with permanent storage, which is usually an order of magnitude larger. Compared with Ladon, which has been validated in severely constrained devices (C0), Hidra additionally requires loading of the policy parser code, the policy fields, the logs pending acknowledgement from the ACS and the corresponding nonce for message pairing.

The policy parser code is highly dependent on the platform and implementation, involves 48 commands and decisions that are required to read and load the fields according to the policy language detailed in Section IV and codified according to Subsection IV-C.

The policy to be loaded in memory does not expand significantly compared with the serialized version detailed in Subsection IV-C. The parsed policy is a set of fields and can be complemented with some data structures aimed at making searches more agile. For example, these data structures could be hash tables that would speed up rule matching



(a) (b)

FIGURE 15. Impact of the cryptographic computation rate on energy consumption from establishment of a secure session using Hidra with four different sample instances, considering the worst case of once per hour as the user request rate. (a) Impact of the encryption rate on energy consumption from establishment of a secure session using Hidra with four different sample instances. (b) Impact of the MAC computation rate on energy consumption from establishment of a secure session using Hidra with four different sample instances.

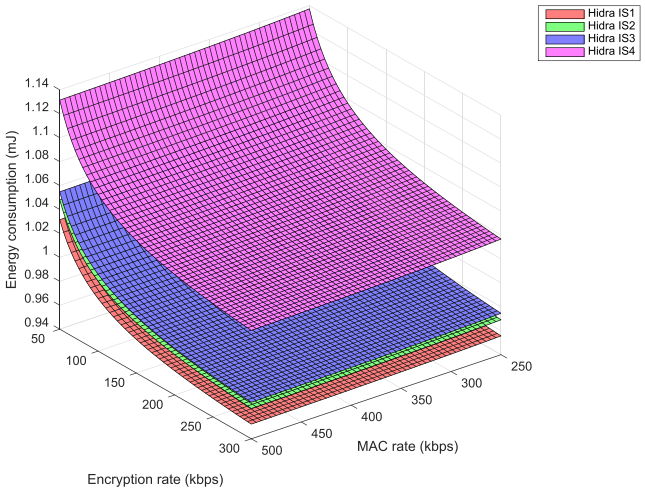


FIGURE 16. Analysis of the impact of the cryptographic computation rate on energy consumption from establishment of a secure session using Hidra with four different sample instances, considering one per hour as the user request rate.

for specific request fields. The number of fields depends on the expressiveness of the policy instance and its length. The number of fields for the four sample instances are 2, 9, 13, and 44 fields, respectively, for the compacted version (2, 7, 9, and 32 bytes, respectively). Moreover, richer policies do not exceed a hundred fields. Therefore, the impact of the policy parsing computation on the memory footprint can vary from implementation to implementation but it remains low and affordable even in severely constrained devices (C0) with only a few KB of RAM memory.

The same concepts apply to the memory required for the accounting logs and for the additional *nonce* used for message pairing. The number of fields is low (under ten); consequently, the memory impact is imperceptible.

Regarding the footprint of the code of the policy decision point and the enforcer, both are usually integrated within the sensing application code, particularly in very constrained devices (C0). Their impact may vary, depending on the platform and implementation, but an estimation based on the pseudocode versions approaches 32 bytes, with 12 required commands and decisions. On one hand, this code is required to make decisions based on policy evaluations. On the other hand, they are required to enforce the resulting granting effect, the correspondent obligations and the optional rule re-evaluation timer and counter according to the policy language detailed in Section IV. Therefore, the footprint of the PDP and PEP code is considered to be both low and acceptable.

However, with regard to the required permanent storage, the storage capacity available in C0 devices is much higher than is the RAM capacity. Typically, this permanent storage capacity exceeds the size needed to store the symmetric key shared with the ACS, the access control policy, the PDM specification file, the code to parse the received policy and the code to run the Hidra protocol. Therefore, although it is dependent on the specific implementation, the impact is considered acceptable.

Additionally, in the case of a planned reset of the CDS during one of the potential authorized maintenance actions, the active security association may be retained and saved to permanent storage by the implementer or by the access control policy editor through obligations. In these cases, additional data to be permanently stored are *lifetimes*, *nonces*, *Subkeys*, $K^i_{R,CM}$ keys and the *Logs* pending acknowledgement. Because the total length is limited to some few dozens of bytes, the storage requirements do not have a high impact on the required permanent storage volume.

VIII. CONCLUSIONS

Incoming smart scenarios enabled by IoT such as smart homes, smart cars, smart offices, smart grids and so on

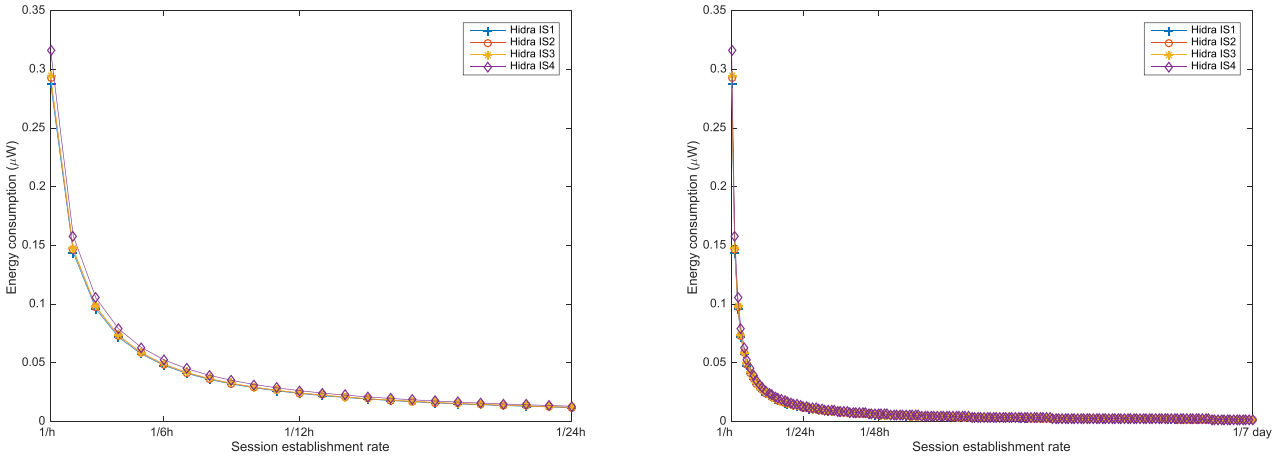


FIGURE 17. Analysis of the impact of the access request rate on energy consumption from establishing a secure session using Hidra with four different sample instances, considering the worst case of a 50 Kbps encryption rate. (a) Impact of the access request rate for a day on energy consumption from establishing a secure session. (b) Impact of the access request rate for a week on energy consumption from establishing a secure session.

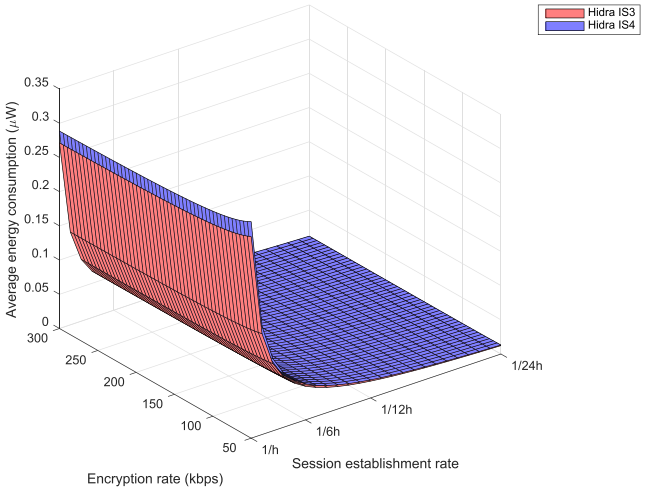


FIGURE 18. Energy comparison on Hidra with samples 3 and 4.

envision smart objects that expose services that will be adapted to user experiences or that will be managed in an effort to gain higher productivity, often in multi-stakeholder applications. In such environments, which frequently imply large scale deployments, smart devices are cheap and therefore, constrained. However, they are also critical components because of the criticality of the information they contribute to higher-level business processes. Therefore, they must implement strong security mechanisms.

However, the existing approaches do not adhere to the principle of least privilege due to the poor expressiveness of their authorization policies, nor is their behaviour flexible and scalable. Additionally, no prior proposal supports local condition checking based on powerful expressions such as comparisons, correlations, *etc*. to supplement simple static attribute matching. Moreover, no prior proposal enforces additional reactive activities launched as obligations, providing more powerful control through policy. Furthermore, the possibility of iteratively re-evaluating an access control policy during access has not been considered, nor have accounting and audit mechanisms been addressed.

Consequently, to the best of our knowledge, the presented access control model is the first proposal to implement expressive access control suitable for severely constrained devices (C0). This novel model tackles the limitations of current access control approaches for constrained devices by allowing fine-grained policy enforcement in deployed sensors based on local context conditions and corresponding obligations. Additionally, to provide control over access behaviour, policy evaluation and enforcement is performed not only during the security association process but also afterwards, while the security association is in use. To avoid overwhelming local storage resources, such a dynamic policy cycle requires an efficient message exchange protocol. For this reason, the Hidra protocol, which implements mutual authentication, expressive policy injection, tight enforcement and accounting for further tracking and auditing purposes, is adopted in this proposal.

Therefore, the proposed access control model described here is the first to bring similar expressiveness and accounting features to C0 and C1 CDSs as are available in the current Internet. The security and performance evaluation concludes that this access control model is both feasible and adequate.

APPENDIX
POLICY CODIFICATION TABLES

In this section the codification tables mentioned in the Section IV-C are conveyed to complete the detailed description of the optimization conventions. In fact, policy language related nested constructs are serialized as shown in tables conveying *expression*, *obligation*, *task*, *attribute*, *value* and remaining *semantic conventions* details.

TABLE 14. Expression codification.

	JSON	NATURE	BINARY	DESCRIPTION
Expression	{function:	Mandatory		Omitted
	byte	Mandatory	[0b00000000 - 0b11111111]	Function id [0 - 255]
		Injected	[0b0 - 0b1]	InputExistenceMask [0 - 1]
		Injected	[0b000 - 0b111]	MaxInputIndex [0 - 7]
	, inputset:[	Optional		Omitted
	attribute+	Expanded	array of attributes	At least one attribute
	]}	Mandatory		Omitted

TABLE 15. Obligation codification.

	JSON	NATURE	BINARY	DESCRIPTION
Obligation	{task:	Mandatory		Omitted
	task	Expanded	task expanded below	One task expanded below
		Injected	[0b0 - 0b1]	FulfillOnExistenceMask [0 - 1] ALWAYS by default
	, fulfillon:	Optional		Omitted
	DENY/PERMIT	Optional	[0b0 - 0b1]	FulfillOn [0 - 1]
	}	Mandatory		Omitted

TABLE 16. Task codification.

	JSON	NATURE	BINARY	DESCRIPTION
Task	{function:	Mandatory		Omitted
	byte	Mandatory	[0b00000000 - 0b11111111]	Function id. [0 - 255]
		Injected	[0b0 - 0b1]	InputExistenceMask [0 - 1]
		Injected	[0b000 - 0b111]	MaxInputIndex [0 - 7]
	,inputset:[	Optional		Omitted
	attribute+	Expanded	array of attributes	At least one attribute
	]}	Mandatory		Omitted

TABLE 17. Attribute codification.

	JSON	NATURE	BINARY	DESCRIPTION
Attribute	{type:	Mandatory		Omitted
	BOOLEAN/BYTE/INTEGER /FLOAT /STRING/ REQUEST_REFERENCE /SYSTEM_REFERENCE /LOCAL_REFERENCE	Mandatory	[0b000 - 0b111]	direct types and references [0 - 7]
	, value:	Mandatory		Omitted
	value	Detailed below	value described below	one value
	}	Mandatory		Omitted

The *expression* codification is expanded in Table 14. The first code is the function *id*, which refers to a well-known function as explained in the policy language definition. Then, the existence of related inputs is specified by an injected *InputExistenceMask* code. When inputs exist, an array containing at least one input is expected. The number of inputs is specified by the *MaxInputIndex* value. The attribute codification is expanded in Table 17.

The *obligation* codification is expanded in Table 15. Each *obligation* launches a task described in detail in Table 16. Task launching can be set to *always* (the default), or be conditional on the rule match evaluation result. When task launch is set to *not always*, the *FullfillOnExistenceMask* bit code is injected. Then, launching conditioned to the deny or permit result is specified by the *FulfillOn* code.

Each *task* is defined as a function like *expressions* with some optional inputs and codified as specified in Table 16. In fact, the first code is the function *id*, which refers to a well-known function as explained in the policy language definition. Then, the existence of related inputs is specified by an injected *InputExistenceMask* code. When inputs exist, an array containing at least one input is expected. The number of inputs is specified by the *MaxInputIndex* value. The *attribute* codification is expanded in Table 17.

Attribute arrays are instantiated in tasks and in expressions, both as inputs. Table 17 specifies the details of attribute codification. The first code specifies the attribute type, some of which are well-known (e.g., Boolean, byte,

TABLE 18. Value codification.

	Attribute type	RANGE	DESCRIPTION
Value	BOOLEAN	[0b0 - 0b1]	False/True [0 - 1]
	BYTE	[0b00000000 - 0b11111111]	[0 - 255]
	INTEGER	[0d0 - 0d65535]	[0 - 65535] (16 bits)
	FLOAT	[0d-1.E-36 - 0d1.E+36]	[-1.E-36 - 1.E+36] (32bits)
	STRING	[0 - 6] characters	String of [0 - 6] characters (3+8*6 bits)
	REQUEST_REFERENCE	[0b00000000 - 0b11111111]	Attribute id. [0 - 255]
	SYSTEM_REFERENCE	[0b00000000 - 0b11111111]	Attribute id. [0 - 255]
	LOCAL_REFERENCE	[0b000 - 0b111]	Expression index [0 - 6]

TABLE 19. Semantic conventions.

Concept	Type	Range
Target action	bit (3)	GET/POST/PUT/DELETE/ANY [0 - 4]
Granting effect	bit (1)	DENY/PERMIT [0 - 1]
Fulfill on	bit (1)	DENY/PERMIT [0 - 1]
Attribute type	bit (3)	BOOLEAN/BYTE/INTEGER/ FLOAT /STRING/ REQUEST_REFERENCE /SYSTEM_REFERENCE /LOCAL_REFERENCE [0 - 7]

integer, float, and string). The other three types are reference types and refer to attribute identifiers. These attribute identifiers refer to attributes available in the request, in the system, or in currently running expressions, as specified in the policy language definition. The *value* codification is specified in Table 18.

After an attribute is instantiated, a *value* is mandatory, with a range depending on the specified type. Note that the short string is expressive enough to manage string values, and by design, it limits the lengths of the strings and, consequently, their codification. Additionally Table 19 shows adopted semantic conventions on values. All of the agreed-upon definitions, such as the maximum number of elements in the arrays of rules, expressions, input attributes, obligations, and so on, which represent a good balance between functionality and performance, are also included in the PDM specification in Section IV-D.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.05.010

[2] C. Bormann, M. Ersue, and A. Keranen, *Terminology for Constrained-Node Networks*, RFC 7228, Internet Requests for Comments, May 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7228.txt

[3] R. R. Schaller, "Moore's law: Past, present and future," *IEEE Spectr.*, vol. 34, no. 6, pp. 52–59, Jun. 1997. [Online]. Available: http://dx.doi.org/10.1109/6.591665

[4] P. Gargini. *ITRS Past, Present and Future*.Accessed on Jan. 31, 2017. [Online]. Available: https://linxconferences.com/wp-content/uploads/2016/04/02-01-Gargini-ITRS-2.0-2.pdf

[5] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, vol. 530, no. 7589, pp. 144–147, 2016. [Online]. Available: http://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338

[6] F. Carrez, M. Bauer, M. Boussard, and N. Bui. (Jul. 2013). *Final Architectural Reference Model for the IoT v3.0*. [Online]. Available: http://www.iot-a.eu/public/public-documents/d1.5/at_download/file

[7] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jan. 2015.

[8] K. T. Nguyen, M. Laurent, and N. Oualha, "Survey on secure communication protocols for the Internet of Things," *Ad Hoc Netw.*, vol. 32, pp. 17–31, Sep. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870515000141

[9] Y. Challal, E. Natalizio, S. Sen, and A. M. Vegni, "Internet of Things security and privacy: Design methods and optimization," *Ad Hoc Netw.*, vol. 32, pp. 1–2, Sep. 2015.

[10] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed Internet of Things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013,

[11] S. Gerdes, L. Seitz, G. Selander, and C. Bormann. (Apr. 2015). *An Architecture for Authorization in Constrained Environments*, Working Draft, IETF Secretariat. [Online]. Available: http://www.ietf.org/Internet-drafts/draft-gerdes-ace-actors-05.txt

[12] (2016). *2016 DYN Cyberattack: From Wikipedia, the free encyclopedia*. Accessed on Dec. 11, 2016. [Online]. Available: https://en.wikipedia.org/wiki/2016_Dyn_cyberattack

[13] B. Moore, *Policy Core Information Model PCIM Extensions*, RFC 3460, Internet Requests for Comments, Jan. 2003.

[14] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, document RFC 7252, Internet Requests for Comments, Jun. 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7252.txt

[15] M. Kovatsch, "CoAP for the Web of Things: From tiny resource-constrained devices to the Web browser," in *Proc. ACM Conf. Pervasive Ubiquitous Comput. Adjunct Publication (UbiComp)*, New York, NY, USA, 2013, pp. 1495–1504. [Online]. Available: http://doi.acm.org/10.1145/2494091.2497583

[16] E. Kim, D. Kaspar, C. Gomez, and C. Bormann, *Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing*, RFC 6606, Internet Requests for Comments, May 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6606.txt

[17] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[18] J. Rivera, *Survey Analysis: The Internet of Things is a Revolution Waiting to Happen*. (Feb. 2015). [Online]. Available: http://www.gartner.com/document/2965320.

[19] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proc. 23rd Int. Conf. Distrib. Comput. Syst. Workshops*, May 2003, pp. 749–755.

[20] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proc. IEEE Symp. Security Privacy*, May 2003, pp. 197–213.

[21] H. Wang and Q. Li, "Distributed user access control in sensor networks," in *Distributed Computing in Sensor Systems*. Burlington, MA, USA: Springer, 2006, pp. 305–320.

[22] S. Cantor, J. Kemp, R. Philpott, and E. Maler, (Mar. 2005). Security assertions markup language SAML 2.0. OASIS. [Online]. Available: http://saml.xml.org/saml-specifications

[23] D. Hardt, *The OAuth 2.0 Authorization Framework*, RFC 6749, Internet Requests for Comments, Oct. 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6749.txt

[24] specs@openid.net. (2007). *OpenID Authentication 2.0—Final*. [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html

[25] D. R. Lindemann, D. Baghdasaryan, E. Tiffany, and F. Alliance. (Dec. 2014). *FIDO UAF Protocol Specification v1.0*. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-protocol-v1.0-ps-20141208.html

[26] S. Srinivas, D. Balfanz, E. Tiffany, A. Czeskis, and F. Alliance. (May 2015). *Universal 2nd Factor U2F Overview*. [Online]. Available: https://fidoalliance.org/specs/fido-undefined-undefined-ps-20150514/fido-u2f-overview-v1.0-undefined-ps-20150514.html

[27] M. Sloman, "Policy driven management for distributed systems," *J. Netw. Syst. Manage.*, vol. 2, no. 4, pp. 333–360, 1994. [Online]. Available: http://dx.doi.org/10.1007/BF02283186

[28] B. Parducci. (2013). Extensible access control markup language (XACML) version 3.0, standard. OASIS. [Online]. Available: http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[29] L. Kagal, T. Finin, and A. Joshi, "A policy language for a pervasive computing environment," in *Proc. IEEE 4th Int. Workshop Policies Distrib. Syst. Netw. (POLICY)*, Jun. 2003, pp. 63–74.

[30] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *Policies for Distributed Systems and Networks* (Lecture Notes in Computer Science). Berlin, Germany: Springer, Jan. 2001, pp. 18–38. [Online]. Available: http://dx.doi.org/10.1007/3-540-44569-2_2

[31] W. Han and C. Lei, "A survey on policy languages in network and security management," *Comput. Netw.*, vol. 56, no. 1, pp. 477–489, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128611003562

[32] V. Shnayder, M. Hempstead, B. R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. 2nd Int. Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2004, pp. 188–200.

[33] S. Gusmeroli, S. Piccione, and D. Rotondi, "A capability-based security approach to manage access control in the Internet of Things," *Math. Comput. Model.*, vol. 58, nos. 5–6, pp. 1189–1205, 2013,

[34] J. B. Dennis and E. C. Van Horn, "Programming semantics for multiprogrammed computations," *Commun. ACM*, vol. 9, no. 3, pp. 143–155, Mar. 1966. [Online]. Available: http://doi.acm.org/10.1145/365230.365252

[35] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. Skarmeta, "Distributed capability-based access control for the Internet of Things," *J. Internet Services Inf. Secur.*, vol. 3, nos. 3–4, pp. 1–16, 2013.

[36] A. F. Skarmeta, J. L. Hernández-Ramos, and M. V. Moreno, "A decentralized approach for security and privacy challenges in the Internet of Things," in *Proc. IEEE World Forum Internet Things (WF-IoT)*, Mar. 2014, pp. 67–72.

[37] J. L. Hernández-Ramos, M. V. Moreno, J. B. Bernabé, D. G. Carrillo, and A. F. Skarmeta, "SAFIR: Secure access framework for IoT-enabled services on smart buildings," *J. Comput. Syst. Sci.*, vol. 81, no. 8, pp. 1452–1463, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000014001858

[38] J. L. Hernández-Ramos, A. J. Jara, L. Marín, and A. F. S. Gómez, "DCapBAC: Embedding authorization logic into smart things through ECC optimizations," *Int. J. Comput. Math.*, vol. 93, no. 2, pp. 345–366, 2016.

[39] S. Gerdes, O. Bergmann, and D. C. Bormann. (Oct. 2015). Delegated CoAP authentication and authorization framework (DCAF). Internet Engineering Task Force, Internet-Draft. [Online]. Available: https://tools.ietf.org/html/draft-gerdes-ace-dcaf-authorize-04

[40] C. Bormann and P. Hoffman, *Concise Binary Object Representation (CBOR)*, RFC 7049, Internet Requests for Comments, Oct. 2013.

[41] M. Vučinic, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti, "OSCAR: Object security architecture for the Internet of Things," *Ad Hoc Netw.*, vol. 32, pp. 3–16, Sep. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514003126

[42] J. Park and R. Sandhu, "The UCON ABC usage control model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, Feb. 2004. [Online]. Available: http://doi.acm.org/10.1145/984334.984339

[43] Z. Su and F. Biennier, "On attribute-based usage control policy ratification for cooperative computing context," *CoRR*, vol. abs/1305.1727, 2013. [Online]. Available: http://arxiv.org/abs/1305.1727

[44] *Wireless Medium Access (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal AreaNetworks (WPANs)*, IEEE Standard 802.15.4, Sep. 2006.

[45] H. Tschofenig, J. Arkko, D. Thaler, and D. McPherson, *Architectural Considerations in Smart Object Networking*, RFC 7452, Internet Requests for Comments, Mar. 2015.

[46] R. E. Pattis. *EBNF: A Notation to Describe Syntax (PDF)*. Accessed on Jan. 31, 2017. [Online]. Available: http://www.cs.cmu.edu/~pattis/misc/ebnf.pdf

[47] J. Astorga, E. Jacob, N. Toledo, and M. Aguado, "Analytical evaluation of a time- and energy-efficient security protocol for IP-enabled sensors," *Comput. Elect. Eng.*, vol. 40, no. 2, pp. 539–550, 2014.

[48] J. Astorga, E. Jacob, M. Huarte, and M. Higuero, "Ladon: End-to-end authorisation support for resource-deprived environments," *IET Inf. Secur.*, vol. 6, no. 2, pp. 93–101, Jun. 2012.

[49] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[50] T. Genet, "A short SPAN+AVISPA tutorial," CELTIQUE-Softw. Certification Semantic Anal., IRISA, Rennes, France, Res. Rep. hal-01213074, version 5, Oct. 2015. [Online]. Available: https://hal.inria.fr/hal-01213074

[51] A. Armando *et al.*, "The AVISPA tool for the automated validation of Internet security protocols and applications," in *Proc. 17th Int. Conf. Comput. Aided Verification (CAV)*, 2005, pp. 281–285.

[52] Y. Chevalier *et al.*, "A high level protocol specification language for industrial security-sensitive protocols," in *Proc. Workshop Specification Autom. Process. Security Requirements (SAPS)*, 2004, p. 13.

[53] D. Basin, S. Mödersheim, and L. Viganò, "OFMC: A symbolic model checker for security protocols," *Int. J. Inf. Secur.*, vol. 4, no. 3, pp. 181–208, 2005.

[54] M. Turuani, "The CL-Atse protocol analyser," in *Term Rewriting and Applications* (Lecture Notes in Computer Science), vol. 4098, F. Pfenning, Ed. Berlin Germany: Springer, 2006, pp. 277–286.

[55] A. Armando and L. Compagna, "SATMC: A SAT-based model checker for security protocols," in *Logics in Artificial Intelligence* (Lecture Notes in Computer Science), vol. 3229, J. Alferes and J. Leite, Eds. Berlin, Germany: Springer, 2004, pp. 730–733.

[56] Y. Boichut, P.-C. Héam, O. Kouchnarenko, and F. Oehl, "Improvements on the genet and klay technique to automatically verify security protocols," in *Proc. Int. Workshop Autom. Verification Infinite-State Syst. (AVIS)*, 2004, pp. 1–11.

[57] M. Kohvakka, M. Kuorilehto, M. Hännikäinen, and T. D. Hämäläinen, "Performance analysis of IEEE 802.15.4 and ZigBee for large-scale wireless sensor network applications," in *Proc. 3rd ACM Int. Workshop Perform. Eval. Wireless Ad Hoc, Sensor Ubiquitous Netw. (PE-WASUN)*, New York, NY, USA, 2006, pp. 48–57.

[58] (2005). *The M/G/1 System, and Inside the Mentioned Theorem is Explained: The Pollaczek Khinchin (P-K) Theorem*. [Online]. Available: http://www.richardclegg.org/previous/networks2/Lecture9_06.pdf

[59] C.-C. Chang, D. J. Nagel, and S. Muftic, "Assessment of energy consumption in wireless sensor networks: A case study for security algorithms," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst. Conf.*, Oct. 2007, pp. 1–6.

[60] S. Didla, A. Ault, and S. Bagchi, "Optimizing AES for embedded devices and wireless sensor networks," in *Proc. 4th Int. Conf. Testbeds Res. Infrastruct. Develop. Netw. Commun. (TridentCom)*, 2008, pp. 4-1–4-10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390576.1390581

[61] *Cryptographic Algorithms' Speed Benchmarking in an AMD64 CPU*. Accessed on Feb. 28, 2017. [Online]. Available: http://www.cryptopp.com/benchmarks-amd64.html

[62] *MEMSIC's TelosB Mote (TPR2420CA) Datasheet*. Accessed on Feb. 14, 2017. [Online]. Available: http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html
[63] "Network performance objectives for IP-based services," Int. Commun. Union, Geneva (ITU), Geneva, Switzerland, Tech. Rep. ITU-T Y.1541, Dec. 2011.

MIKEL URIARTE received the B.Sc. and M.Sc. degrees in telecommunication engineering from the University of the Basque Country, in 1998. He spent one year in public R&D in a telecommunications enterprise (currently Tecnalia). Since 1998, he has been with Nextel S. A., a telecommunications enterprise providing ICT engineering and consulting services. From 2001 to 2006, he was the ICT Director and an Information Security Lead Auditor, subsequently becoming Head of the Research and Development Unit. His research interests include ICT interoperability, resilience, and performance and security in several areas, such as identity and access control, networking, wireless sensing, and cloud computing.

JASONE ASTORGA received the B.Sc. and M.Sc. degrees in telecommunication engineering in 2004, and the Ph.D. degree from the University of the Basque Country (UPV/EHU) in 2013. From 2004 to 2007, she was with Nextel S. A., a telecommunications enterprise. In 2007, she joined the Communications Engineering Department, UPV/EHU, as a Lecturer, and a Researcher with the I2T Research Laboratory. She is currently an Assistant Professor with the Communications Engineering Department, UPV/EHU. Her research interests include wireless networking, IP-enabled wireless sensors, security in distributed environments, and mobility management.

EDUARDO JACOB (SM'12) received the B.Sc. degree in industrial engineering and the M.Sc. degree in industrial communications and electronics from the University of the Basque Country (UPV/EHU) in 1991, and the Ph.D. degree from ICT in 2001. He spent two years in a public R&D telecommunications enterprise (currently Tecnalia). Later, he spent several years as the IT Director in the private sector before returning to the Faculty of Engineering in Bilbao (UPV/EHU), where he was Head of the Communications Engineering Department from 2012 to 2016. He is currently an Assistant Professor with the Faculty of Engineering in Bilbao. He also leads the I2T (Engineering and Research on Telematics) Research Laboratory. He has directed several Ph.D. theses and managed several research projects at the local, national, and European levels. His research interests include applying software-defined networks to industrial communications, cybersecurity in distributed systems, and IP-enabled wireless sensors.

MAIDER HUARTE received the Ph.D. degree in 2009. Her Ph.D. thesis was on the design and validation of cryptographic communication protocols for i-voting systems. She has been a Lecturer with the University of the Basque Country since 2002, where she began teaching computer science subjects, such as programming languages and computer communications. She currently teaches advanced Java EE programming and networking. She is also a member of the I2T Research Group, where she has been involved in research on cryptographic security, distributed systems, and software fault tolerance. She also conducts research into software-defined networks and openflow.

MANUEL CARNERERO received the B.Sc. degree in computer engineering from the University of the Basque Country in 2001. In 1998, before finishing his studies, he was with Panda Security as a Virus Detection Technician. In 1999, he joined Nextel S. A., where he was a System Administrator, a Developer, a Database Designer, and an Analyst. Since 2009, he has been a R&D Researcher, participating in national and European research projects. His research work has been related to the evaluation of operational security assurance, and to other initiatives, such as enforcing security policies, and applying adequate security monitoring to different application environments.