HADA: Hybrid Access Decision Architecture for Building Automation and Control Systems

Amit Soni[1], Sye Loong Keoh[2], Sandeep S. Kumar[1], and Oscar Garcia-Morchon[1]
[1]Lighting Control Systems Dept., Philips Research, High Tech Campus 34, 5656 AE, Eindhoven, The Netherlands
[2]School of Computing Science, University of Glasgow Singapore, 9 Woodlands Avenue 9, Singapore 738984
amit.soni@gmail.com, syeloong.keoh@glasgow.ac.uk, {*sandeep.kumar, oscar.garcia*} *@philips.com*

Industrial Control Systems (ICS) and Building Automation and Control Systems (BACS) are being deployed to enable monitoring and control of various intelligent systems like Heating, Ventilation and Air Conditioning (HVAC), safety, access and lighting systems. Each system is an integral part of the ICS and BACS, allowing for optimized industrial operations where devices interact with each other, with users and with other third party systems such as energy management. A key need when interacting is the controlled and trustworthy disclosure of information so that only authenticated and authorized entities can have access and control the resources of a device. However, secure authentication and authorization is not easy due to the combined distributed/centralized operation of ICS and BACS, its large scale deployment, as well as the resource-constrained nature of sensors and actuators. This paper analyzes the security requirements and constraints in ICS/BACS and proposes the Hybrid Access Decision Architecture (HADA) to allow for interoperability between centralized and distributed access control methods. While a central party is in control of policy specification, the system also allows for the deployment of lightweight and compact access control policies to the target devices so that access control decisions can take place in a distributed manner. Our prototype that is based on 6LoWPAN/CoAP IP protocols and binary JSON access control policies shows the feasibility of our approach.

Information Security, Access Control, Authorization, Building Automation and Control Systems

1. INTRODUCTION

Industrial Control Systems (ICS) and Building Automation and Control Systems (BACS) provide automated control of the various parameters of an industrial plant, factory, field experiment site, building, etc. Traditionally, BACS mostly consisted of heating, ventilation and air-conditioning (HVAC) systems and for large buildings such as schools, hospitals and offices, the construction costs constitute only one seventh of the overall operational costs Kastner et al. (2005). Therefore, optimizing BACS provides a great savings potential by reducing the operational costs. The same cost savings potential applies to the ICS as well. In recent years, the role of BACS has been broadened beyond reducing building energy and maintenance costs to providing flexibility and personalized control to the users in the building. Additionally, the number and the capabilities of devices used in modern BACS are also increasing owing to the cheap hardware costs and advances in the wireless technologies. BACS are now on par with ICS in that it is on its way to become a prime example of a distributed control system equipped with sensors and monitoring devices along with the actuators (e.g., ballasts) to be used for data acquisition, service automation and management purposes. In the future, BACS would be the major part of building information system backbone. Machine-to-machine networking would drive every connected component to be Internet addressable. Commissioning, monitoring, trending and archiving will become a norm.

This trend brings with it new concerns related to the proper flow of information and control in the BACS, and security attacks that could cause unwanted consequences. These unwanted consequences could range from simple privacy breaches (e.g., spying on the trends of a particular office room) to critical life threatening situations (e.g., a compromised lighting and ventilation system).

Current literature reveals that less work has been done to design BACS based on its safety and security, and therefore it is possible to compromise the system in numerous ways Novak et al. (2007). To overcome these concerns and provide a fully functional BACS, the intended flow of information and control cannot be governed by simple static rules but it is dependent also on the dynamic environment and the context. A simple example of such a situation would be to disallow any remote request to turn on the heating when there is no one present in the room as detected by the presence sensors. The handling of the dynamism in the access rules along with the constrained resources of the field devices makes the access control system non-trivial, requiring new solutions in this area.

The aim of this paper is to analyze the access control requirements of BACS. One of the key requirements is to allow for both centralized and distributed operations matching the operational features of existing systems. Based on these requirements and existing access control mechanisms in various fields, we present a design of an access control system suitable for BACS/ICS, namely Hybrid Access Control Decision Architecture (HADA). HADA relies on a single trusted entity in the BACS to maintain the access control policies and taking access control decisions, but it is further complemented with a

distributed access control capability in which sensors and actuators can cache access control rules and using a lightweight access control engine, take access control decisions in a distributed fashion. DTLS and cryptographic keys are used to allow for authentication and secure transfer of policies or context. A prototype of the access control system is implemented and evaluated on a wireless constrained network using 6LoWPAN/DTLS/CoAP and made up of devices with limited computational capabilities and memory resources.

The paper is organized as follows: In Section 2, we discuss related work and show that an access control solution for BACS is lacking. In Section 3, we describe the general architecture of BACS, present use cases, and outline specific access control requirements. Section 4 introduces the HADA, and this is followed by a description of its implementation and evaluation in Section 5. Finally, we conclude in Section 6.

2. RELATED WORK

In this section, we first review relevant control and communication systems in BACS, explaining their main security features. Here, we observe that there is a trend towards Internet Protocols (IP) systems that will allow for easy interaction between different systems in BACS, and interaction that needs to be controlled. For this, we further present relevant work on access control models and technologies.

2.1. Security in BACS

Building Automation and Control Networking Protocol (BACnet) ANSI/ASHRAE 135 (2008) is being developed since 1987 as a standard to enable the interoperability between different vendors and multiple types of building systems. Messages in BACnet can be transported over a variety of networks. In 1999, BACnet/IP was standardized to allow BACnet devices to communicate using IP. Additionally, a transparent bridging solution based on IEEE 802.11 has been proposed to support wireless connectivity. The BACnet security specification covers data origin, operator authentication as well as communication security. However, authorization and access control functions are not present leading to proprietary extensions to the BACnet security architecture.

LonWorks ANSI/EIA/CEA 709.1 (1999) is an event-triggered control network system. It consists of LonTalk communication protocol, a dedicated controller and a network management tool. Devices in a building are organized in domains, each domain can hold up to 255 subnets. Unicast and multicast communication can be performed in a reliable way in LonWorks if reliable transmission mode is chosen. A challenge-response authentication mechanism is defined for acknowledged message transmission. There are security concerns because the security algorithm is not public and relies on too short keys (48-bit).

Supervisory Control and Data Acquisition (SCADA) system is widely used for the control and monitoring of industrial, infrastructure and facility processes. SCADA usually implements centralized systems that manage the entire site or building. The SCADA communication infrastructure allows for remote management using Telemetry using pre-established corporate networks. The third generation of SCADA saw transition from a proprietary protocol towards standardized communication and security protocols. One of its main security vulnerabilities Tsang is the lack of authorization support, resulting in anyone who can send packets to the SCADA device can control it. The security protection on the control software can be easily bypassed, thus gaining full control of the SCADA Networks.

In the same manner, e.g., BACnet allows for IP connectivity, there is a current proposition of connecting BACS devices to the Internet via Internet Protocols (IP), thus leading to the integration of BACS and IT systems. New protocols such as 6LoWPAN or CoAP play an important role. The 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) IETF working group defines how to overlay IPv6 on IEEE 802.15.4 Kushalnagar et al. (2007), Montenegro et al. (2007). The Constrained Application Protocol (CoAP) Shelby et al. (2012), is an application layer RESTful protocol for constrained environments. It is analogous to HTTP in web and is easily translated to HTTP for mutual integration while minimizing complexitiy to adapt to constrained environments. These IP protocols require Datagram Transport Layer Security (DTLS) Rescorla and Modadugu (2006). Optionally, IPSec Kent and Atkinson (1998) can be used.

There are other relevant works beyond standards and current standardization activities. In Granzer et al. (2010), the authors identified security requirements and provided a comprehensive threat analysis in BACS. An attempt to use existing standards such as Transport Layer Security (TLS) Dierks and Rescorla (2008) and Virtual Private Networks (VPNs) to establish secure communication channels between devices was presented. In order to protect the software running on the device, a sandbox was proposed as a secure environment for BACS devices, hence protecting against software attacks. However, it still lacks of a mechanism to provide authorization. Similarly, the authors in Granzer et al. (2006) presented a survey of various security flaws in BACnet, LonWorks, and KNX/EIB (European Installation Bus). They proposed EIBSec to provide data authenticity, integrity and confidentiality to EIB, including a key management and distribution scheme for KNX/EIB.

2.2. Access Control Methods

Discretionary Access Control (DAC) Lampson (1974) has been widely used and deployed in the IT world to enable access control policies for a resource, to be specified by its owner. However, it does not adequately address the needs in BACS, as managing and storing an access control list or an access matrix on each each device is not scalable.

Role-Based Access Control (RBAC) Sandhu et al. (1996), Ferraiolo et al. (2001) is a non-discretionary access control model in which the subjects are assigned one

or many roles determined by the system. It provides the flexibility of assigning permissions to a group of users that have been assigned to a role. Furthermore, it has also been extended to support role hierarchy Ferraiolo et al. (2001). More complex properties like separation of duties can also be easily incorporated in the RBAC model. Attribute-Based Access Control (ABAC) Park and Sandhu (2004), Wang et al. (2004) can be used to overcome the shortcomings of RBAC in that attributes can be associated with any entity in the system including users, subjects, targets, and contexts, thus when unified with the existing RBAC framework Huang et al. (2012); it can provide finer granularity of authorization. In Sandhu (2012), the author advocated that by adopting ABAC, policies can be formulated by security architects to translate from attributes to rights, and dynamic elements can be built into these policies so that the outcomes of access control decisions automatically adapt to changing local and global circumstances. However, the risk is that it results to potential conflicting decisions due to the provenance of attributes obtained.

Beyond the above general techniques, many proposals have been made to address context-aware access control in smart intelligent environment, the closest research area to our application space includes Kulkarni and Tripathi (2008), Corradi et al. (2004), Sacramento et al. (2005). They described context-aware RBAC, however Kulkarni et al. described a context-aware role-based access control model in pervasive computing systems at SACMAT'09 Kulkarni and Tripathi (2008). Corradi et al. (2004) and Sacramento et al. (2005) had proposed related methods for context-aware access control. there are practical limitations such as the management of spatio-temporal access control systems Damiani et al. (2009). At SACMAT'10, Garcia-Morchon and Wehrle (2010) presented a modular access control model that allows for distributed operation and evaluation of modular policies; however, that work did not address the interaction between centralized and distributed systems.

3. BACS MODEL, USE CASES, AND GOALS

We assume a general BACS model Kastner et al. (2005), ISO Standard. 16 484-2 (2004) in which three levels of interactions are defined, namely *management* level, *automation* level and *field* level. As we will see, the first level works in a centralized fashion while the second and third level are rather distributed relying on the information of the *management* level. Sensors, actuators, and devices such as lights, sensors, actuators, windows, and blinds, etc operate at the *field* level. These *field* devices interact with each other by issuing control commands, e.g., switching, dimming, setting, positioning the devices in response to instructions from the system (i.e., *automation* level). Additionally, they collect energy data, measurements and metering to be processed.

The *automation* level is responsible for scheduling and issuing autonomous execution of commands to the *field* devices. A control loop is typically used where events generated by *field* sensors trigger automated actions on the corresponding *field* actuators. Raw data and measurements received from the *field* level are aggregated and then forwarded to the *management* level for (trending) analysis.

Information throughout the entire BACS is accessible at and handled by the *management* level. All the data collected, aggregated, processed can be used for reporting and statistics purposes. Logs and alerts are generated for exceptional situations. Furthermore, data mining techniques can be employed to aid the data analysis in order to optimize the BACS, in particular refining the schedules and automation control in the building.

3.1. Exemplary Use Cases

Our exemplary use cases are about a BACS controlling and managing three systems in an office building, namely the lighting system, the window blinds system, and the fire system, while these systems interact with each other and with users. For instance, in the Lighting case, devices and lights/luminaires are connected to a Digital Addressable Lighting Interface (DALI) network Hein (2001) that is connected to a *bridge/router*. These *routers*, *bridges* and *controllers* are responsible for managing the communication functionalities of the Lighting networks, e.g., bridging the IP network and the DALI network, routing traffic between rooms and between floors, as well as controlling the Lighting sensors and actuators. A *Domain Monitor/Automation* is usually present in the building to maintain the automation schedules and invokes services on these lighting devices. Using the above use cases, we further explain the essential interactions in BACS that we envisioned as follows. These use cases also reveals that it is of utmost importance that the interactions in BACS are secured and regulated.

- **Direct interaction between user and system:** Users of an office building are given personalized control of the lighting systems around them, thus enabling the lighting devices to be controlled by the users working in that room only. This also implies that switching and dimming commands must be regulated to prevent attackers from gaining control of the network.
- **Direct interaction between systems:** Based on the availability of sunlight, the lighting in an office room can be dimmed down if the daylight is sufficient for the room illumination. This would require controlled interaction between the lights and the window blinds in the room.
- **Context-aware operation:** When the fire sensors are activated in situation where a fire breaks out, the normal operation maybe disabled during this time, thus allowing the *Building Monitor/Automation* to have control of the building. Such context information is important and must be taken into account when performing access control.

3.2. Access Control Goals

This section outlines the security requirements based on the use cases and the related studies Grummt and Muller (2008), Weber (2010).

- **Suitable for underlying operational architecture in BACS:** Access control system should be such that it can be easily integrated into the existing automation infrastructure in BACS. Thus, the operation should fit both distributed and centralized communication patterns.
- **Fined grained access control:** A device should be able to access the services provided by another device if and only if it is authorized to do so.
- **Role-based access control:** Devices and users using the services provided by the devices should be assigned roles to help facilitate access control in a larger scale.
- **Context and content handling in the policies:** Context and content on the devices providing the services should be incorporated in the access decisions.
- **Able to handle the exchange of information between different administrative domains :** Access control system should be able to evaluate and enforce decision on requests from a different administrative domain seamlessly.
- **Efficient access decision:** Due to the limited resources on the sensor nodes, the decision evaluation and enforcement should be efficient.

4. HADA: HYBRID ACCESS CONTROL DECISION ARCHITECTURE

As we observed in the last section, the trend towards IP and the increasing intelligence of BACS devices lead to a more direct interaction between them in a fully distributed manner. Nonetheless, the overall BACS is still rather centralized and the existing related work does not provide a good solution that combines both of them. Thus, we present our Hybrid Access Decision Architecture for BACS that aims at bringing together centralized and distributed access control approaches.

4.1. Architectural Design

Figure 1 shows the architecture of HADA in which we can observe the *Access control Trust Point* (ACTP) and two devices. The Access Control Trust Point is located at the management level as defined in Section 3 and contains the centralized Policy Decision Point (PDP) and centralized access control policies for governing the interaction between devices that spans different administrative domains in the building (automation and field levels as in Section 3). The two devices are any two devices in the building, i.e., at the automation and field levels. At the ACTP, a *management interface* is available to allow the administrator to deploy, update and delete the access control policies. The PDP is responsible for interpreting and evaluating the policies before granting permissions to the devices to perform the requested actions. Every device also contains a local PDP, a cache to store access control policies relevant for its own operation, as well as the Policy Enforcement Point (PEP) where the decision of the either centralized or local PDP is enforced.

Based on this architecture, the operation is simple. A device in the BACS can take an access control decision related to a access request from another device/user in two ways. In the centralized way, it relies on the central PDP at the ACTP. In a distributed way, it relies on its local PDP and the policies in its cache. The reasoning for a hybrid access decision architecture is simple, if only the centralized operation is used, the system would not scale as discussed in Garcia-Morchon and Wehrle (2010), therefore a more distributed method is needed.

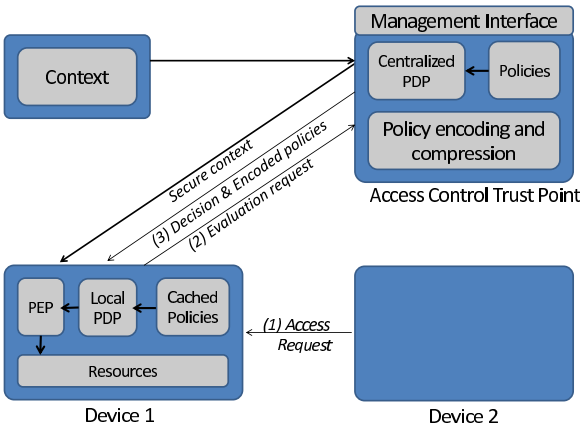

Figure 1: *The Hybrid Access Decision Architecture*

The cache of a device can be populated either in a proactive or reactive manner. It happens in a proactive manner, if it occurs during the registration of the device in the BACS. At this stage, very basic rules can be deployed because at this stage, it is still not known which other devices/users will be using the resources of a device. It happens in a reactive manner, when it is triggered by an access request from another device/user and at that stage the device cache does not contain any rule for the request. In this case, the device contacts the centralized Policy Decision Point to determine whether to grant permissions to the requestor to access the underlying resources. Since a continuous centralized access is not efficient, local caching for the corresponding rules is applied. Various caching algorithms discussed and proposed in Megiddo and Modha (2003) can be used. With this, in general, the device first checks its locally cached policies, if the corresponding policy is found, then it uses the local PDP. Otherwise, it contacts the centralized PDP.

In addition, context information is fed into the policy engine to enable context-aware policy evaluation. The context is maintained in a centralized fashion so that it can be used in a centralized access control request. To enable for distributed access control decisions, relevant context needs to be locally available, and for this, the context information needs to be securely distributed from the Access Control Trust Point.

In our architecture, the authentication between devices and the BACS is being taken care of by the BACS management level. This happens during the registration of the devices (see Sarikaya et al. (2012) for a related approach) and during this process a joining device

authenticates itself to the BACS management level in the building, and the BACS management level is responsible for distributing a shared secret-key between two communicating parties, e.g., between the ACTP and a device, a device and a sensor, context service and the ACTP, etc. In practice, the BACS management level also implements the Access Control Trust Point. In HADA, we use Datagram Transport Layer Security (DTLS) Rescorla and Modadugu (2006) to facilitate authentication and key distribution but other security protocols might also be applied Neuman and Ts'o (1994), Steiner et al. (1988), Dierks and Rescorla (2008).

4.2. Policy Specification

The policy is structured according to Figure 2 specifying the *subject*, *target*, *action* and *condition*. The *subject* is the entity that is requesting to invoke an action or access a resource; the *subject* can be specified in terms of role which applies to a group of users or devices. The *target* is the resource in which the action will be invoked, while *action* indicates the operation and command to be performed on the target resource. Lastly, the *condition* is used to encode constraints and context information which can change dynamically. It is advocated that rules in the policies are grouped together based on the *target* because in most cases, it is the target devices that are enforcing the policies. With such grouping, only the relevant policy rules that need to be enforced are deployed onto the devices, thus simplifying the caching process.

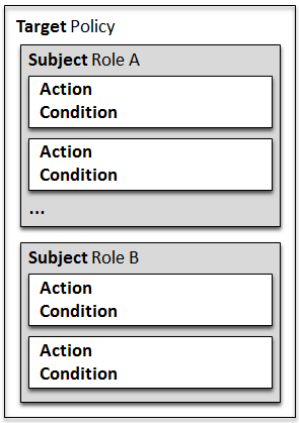

Figure 2: *Policy Structure*

The access control policies deployed in BACS are defined according to the operational requirements of BACS. As described previously, these polices are stored in a policy repository hosted in the central *Access Control Trust Point* (ACTP). If there are interactions between different domains, e.g., the lighting domain and the windows blinds domain, the policies governing their interaction are specified in accordance to the service level agreements between them. Based on the frequency of interaction, the central ACTP can push relevant policies to the field devices. Each field device has a small allocation of memory to cache the policies it needs to enforce.

4.3. Access Control Process

This section describes the access procedure for two yet mutually unknown devices in a BACS to regulate each other's behaviour, assuming that the appropriate access control policies have already been defined and reside in the central *Access Control Trust Point*. In this description we cover the device installation/commissioning, later device/resource discovery, and the local and central access control decisions. In the first two phases, the devices are provided with basic operational parameters that are used during the actual access control phase.

Devices are first installed and commissioned in the BACS and during this process, the BACS is in charge of adding them to the system and providing them with operational parameters. Next, devices *discover* each other and their resources, this happens after installation or when devices get close in a building and interaction is needed. After discovery and following the process in Figure 1, a device will send a request to access the resources of another device. If the device stores the suitable policies, then the decision will be based on the local PDP. If it is a cache-miss, the device delegates the policy decision to the central PDP in ACTP and fetches a compact representation of the policy to be stored locally.

Note that we describe how the system works if IP protocols are used as used later in our prototype, namely 6LoWPAN for networking, CoAP and DNS-Service Discovery (DNS-SD) S. Cheshire and M. Krochmal (2012) for application services, and DTLS for security.

4.3.1. Device Installation/Commissioning

When devices are installed in a building, they are authenticated and authorized to join the BACS by the central *Access Control Trust Point*. This can be done based on a number of approaches Sarikaya et al. (2012). During this process, the devices receive operational parameters such as an IP address or relevant keying materials (e.g., symmetric-keys) so that devices can establish a secure (unicast or multicast) communication channel, authenticate each other, and transfer information in a secure way in later phases. As described in Section 4.1 these keys are typically distributed when the devices register for the first time in the BACS. During this phase, the *Access Control Trust Point* can already distribute to a new device some relevant policies that apply to its operation. Thus, the policy distribution and caching is done in a proactive way.

4.3.2. Device Discovery

After installation/commissioning, devices that are installed in a building are not necessarily fully configured, depending on how the devices are programmed, they can dynamically discover other devices in proximity. For instance, a window blind might be configured to search for all the lights in the same room, it is then able to dim them whenever there is enough light outside. Also it might be that a user with a mobile phone moves into a room and wants to control the heater or air-conditioner. Searching a device of interest generally happens in two ways.

- **In field searching:** A device that wants to access resources residing in other devices has to broadcast a discovery protocol message communicating that it is looking for particular type of devices. This broadcast can either be encrypted by the domain wide discovery key or can be in plaintext. The encryption of the discovery message is to hide the search queries made by devices from outside the domain. Either way, devices of the requested type understand the broadcast and reply with their corresponding identities and addresses.
- **Directory based searching:** Relies on a central repository such as a DNS-SD S. Cheshire and M. Krochmal (2012). Devices send the discovery protocol message to the the central ACTP with optional encryption using a domain wide discovery key. This assumes that the ACTP implements a protocol such as DNS-SD S. Cheshire and M. Krochmal (2012), and it simply replies with the identities and addresses of the corresponding devices.

Upon successful discovery of each other, the communicating devices authenticate each other using DTLS Rescorla and Modadugu (2006) using this secure link to protect the access requests and the access decisions.

4.3.3. Local Access Control Decision

Once devices know about each other's identity and address, service access can be requested, e.g., the window blind can send a CoAP command to switch off the light in the room when there is sufficient daylight. The service access command is protected with the secret-key shared between the two communicating devices. When receiving such a service request, the *target* device searches for the access control policies that apply to the device requesting for the action, and evaluates the policy to determine whether the requested action(s) can be granted. The context information is also taken into account when evaluating the policies, ensuring that the *condition* clause is satisfied. Using the shared key, the *target* device can authenticate the requesting device, and if needed, maps it to a role assignment.

In case that it is a cache-miss access, the *target* device cannot make an authorization decision. It subsequently contacts the central *Access Control Trust Point* as described in Section 4.3.4. That is, this is a reactive way of populating the cache with relevant access control policies.

4.3.4. Central Access Control Decision

Access decision is delegated to the central PDP in ACTP when there is a cache-miss in the *target* device for a given request. The access request is then forwarded to the central PDP containing all deployed access control policies. This access request needs to include a signature from the requester and the target device so that the central ACTP can ensure that the information is actually required by the *target*. The corresponding access control policies are evaluated according to the access request, taking into account the context information gathered from the context service. The access decision together with the compact representation of the corresponding access control policies are securely returned to the *target* device ensuring that the message is not tampered with.

5. IMPLEMENTATION AND RESULTS

This section describes the implementation of HADA. We detail the access control policy encoding, the execution environment and demonstrate the feasibility of the implementation with evaluation results.

5.1. Policy Encoding

The access control policies specified by different stake holders are deployed into the centralized ACTP. These policies are encoded using the Javascript Object Notation (JSON) Crockford (2006), which is considered as less verbose compared to XML and that the interpretation of JSON is more efficient. A custom policy interpreter has been implemented based on the standard JSON parser, thus enabling the access control policies to be interpreted and evaluated. The JSON policy language supports the role based definition of the policies. Therefore, we are able to simplify large number of users in the BACS. When a request is sent to a *target* device from a certain user, the user must be mapped to an authorized role in order to execute the action.

Figure 3 shows an example of access control policy encoded using JSON. The policies are grouped according to the *target* object, namely the specific instance of *Light_001*. Two *subject*s are defined in this policy, namely *Switch* and *User* where each of them has a different set of permissions. In the first policy, the *Switch* is allowed to perform action *on*, and *off* on *Light_001*. Whereas a *User* is permitted to perform the same actions as the *Switch* with the condition that the time is only between 08:00 AM and 20:00 PM.

Note that the condition is specified using the *attribute* tag. This *attribute* tag is used to associate context information and constraints with the policy. The following sub-tags of *attribute* are supported:

id: denotes the context variable.

priority: denotes the priority for caching the context variables. Caching algorithm is supposed to use this tag for intelligent caching.

scope: a parameter used to decide if the context variable is evaluated by the central ACTP or by the device itself. When the tag value is *global*, the context must be evaluated centrally as the context information may depends on the network setting, while *local* means that the context variable can be evaluated by the device itself.

eval: a recursive structure of *eval* tags which depicts a series of nested binary conditions to which the context variable is subjected.

5.2. Compact Policy Representation

As the access control policies are grouped according to the *target* object, the policies can be deployed and cached

```
{
   "target": "Light_001",
   "Switch": [{
      "action": "on", "off",
      "condition": [{
         "attribute": {
            "priority": "normal",
            "scope": "local",
         }
      }],
   }],

   "User": [{
      "action": "on", "off",
      "condition": [{
         "attribute": {
            "id": "time",
            "priority": "high",
            "scope": "local",
            "eval": {
               "function": "AND",
               "eval": {
                  "function": "lt",
                  "value": "20",
               }
               "eval": {
                  "function": "gt"
                  "value": "8",
               }
            }
         }
      }],
   }],
}
```

Figure 3: *Example of access control policies encoded using JSON*

in the *target* device's local storage in order to optimize the policy evaluation process. However, storing policies in JSON format and fitting a JSON policy parser and interpreter in a constrained device pose some challenges. It is conceivable that the size of the policies to be deployed on the device can be greatly reduced, and the policy evaluation mechanism can be optimized.

An analysis of a simple policy encoded in JSON as shown in the previous example accounts to a size of approximately 200 bytes in ASCII. As a result, the number of policies that can be deployed on a device's local storage may be limited, and therefore increasing the probability of cache-miss when receiving an authorization request. Consequently, this increases the frequency to contact the ACTP and defeats the purpose of having a local policy store. Moreover, interpreting the JSON text in the constrained devices also consumes significant amount of active memory. For these reasons, the JSON policies are optimized before they are deployed to the device's local cache.

The JSON policies in the central ACTP is re-encoded in the *Type-Length-Value* (TLV) format Myers et al. (1999) in binary before they are deployed onto the target devices. The *Types* in TLV is represented as four bits denoting the JSON *tags* in the system. Different values of JSON tags are shown in Table 1:

attribute	0001
id	0010
priority	0100
scope	0101
eval	0110
function	0111
value	1000

Table 1: *Pre-defined representation of JSON* tags *as* Type

Each *type* has a *value*. The scope of the *value* in the encoded policy is de-limited by the *length*. The *length* describes number of bits in which the *value* is represented. The *length* is represented by four bits.

time	0001
people presence	0011
natural illumination	0100
temperature	0101
humidity	0110

Table 2: *Pre-defined* values *for type 'id'*

The *values* in the system are also pre-defined. For example, the *id* field that represents the context variables can be one of the following: {*time, presence, natural illumination, temperature, humidity*}. These values are pre-encoded as in Table 2. Using 4-bits to represent the values of 'id' only provides 16 possible values. This can be extended in the future to support more context variables.

Table 3 provides pre-defined encoding of some of the values for different JSON *tags*. It presents the basic *types* and their pre-defined *value*s.

The caching algorithm implemented on the *target* devices is the classical least recently used algorithm. The least recently used items are discarded first. A linked-list is used as the data-structure to store the encoded policies. The most recently used policy is automatically placed at the head of the linked-list. Therefore, when the link-list is full, the tail element of the link-list is removed, so that new policy can be introduced at the head.

5.3. Deployment Environment and Hardware Platform

5.3.1. Deployment Environment

Figure 4 shows the deployment environment of HADA. All devices or sensors run Internet Protocol (IP), in which a 6LoWPAN Kushalnagar et al. (2007) network stack is deployed. Device services have also been implemented to enable interaction with other devices and actuators. Invocation of the service is regulated by an Access Control Layer (AC Layer) which is responsible for interpreting and evaluating compact access control policies on the devices. Service requests and service responses are transported

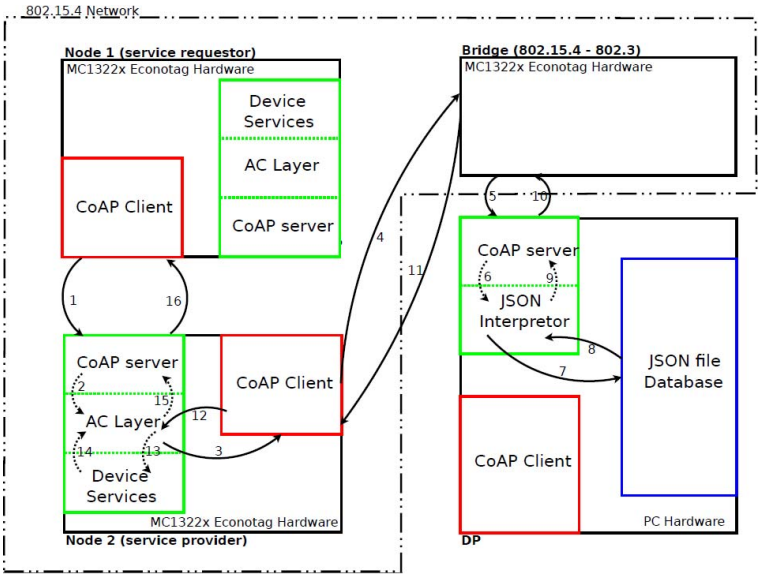

Figure 4: *Deployment environment*

using Constrained Application Protocol (CoAP) Shelby et al. (2012). All devices have an IP connectivity to a bridge that enables them to delegate access control evaluation to the ACTP and fetches the compact policies.

5.3.2. Devices and Sensors

The MC1322x Econotags MC1322x (2013) were used in this proof-of-concept. Each Econotag runs Contiki OS for constrained devices Dunkels et al. (2004). The relevant hardware specification of MC1322x are as follows: (i) IEEE 802.15.4 standard compliant on-chip transceiver; (ii) AES 128-bit co-processor; (iii) 32-bit ARM7TDMI-S CPU core with programmable performance of up to 26 MHz (24 MHz typical); and (iv) Extensive on-board memory resources with 128 Kbyte serial FLASH memory (can be mirrored into RAM), 96 Kbyte SRAM and 80 Kbyte ROM. Due to its limited memory and resources, the compact policies encoded in TLV format are deployed on these devices together with the corresponding policy interpreter.

5.3.3. Access Control Trust Point

A standard PC hardware with Linux as the operating system is assumed for the ACTP. It is more resourceful and has the capability of interpreting access control policies encoded in JSON format.

5.4. Results

In this section, we present an evaluation of our proof-of-concept with focus on the access control functionalities in terms of memory requirements on the Econotags and performance benchmark. The performance of the underlying security layer (DTLS) is not included in this paper but can be found here Garcia-Morchon et al. (2013).

5.4.1. Code Size

We present the size of the compiled binary code which provides the functionality of the access control layer on the constrained devices. This excludes the size of the code which provides the functionality of CoAP server through which the service is provided. The code size also excludes the size of the μIP stack code. In total, the size of the access control layer is 4908 bytes, which is feasible on the hardware platform we envisioned. Table 4 shows the code size of all components that are associated with the access control layer. The general purpose CoAP handler notably includes the functions of retrieving and setting the query options, header content type, and the payload. It also contains the code that is responsible for parsing the binary CoAP packet when it arrives.

In addition to the code size presented in Table 4, 2000 bytes have been allocated to store all the data structures for CoAP and the access control layer. Moreover, an extra 2000 bytes are used as the cache to store compact access control policies in a linked-list.

5.4.2. Size of JSON and Compact Policies

In general, the policy size depends on the way the policies are structured for the BACS. In our implementation, the access control policies are encoded in both JSON format and the binary encoded TLV format. The measurements reveal that on average, an access control policy specifying only one *subject* in JSON format is approximately 600 bytes, whereas the compact representation of the same policy in binary TLV format is around 15 bytes. This has shown the significant reduction in the size of policy for deployment on constrained devices. However, we would like to highlight the fact that the policies used in the proof-of-concept only defined two to three permissions on a single resource for a particular *subject* and hence the size calculation would vary greatly depending on the number of *subject*s and permissions in the policy.

5.4.3. Performance

The performance was evaluated in a setup in which: (i) the devices were in close proximity; (ii) the devices were limited in number; and (iii) there were almost no packet loss. The time required to serve a service request when the *target* device had a cache-miss and had to contact

type *length* *value*	
attribute	0001
id	0010
4	0100
time	0001
people presence	0011
natural illumination	0100
temperature	0101
humidity	0110
priority	0100
2	0010
high	00
normal	01
low	10
scope	0101
1	0001
local	0
global	1
eval	0110
function	0111
4	0100
AND	0000
OR	0001
gt	0010
lt	0011
eq	0100
value	1000

Table 3: *TLV pre-defined* types *and* values

the central ACTP to evaluate the access control policy was approximately 0.09s. On the contrary, in the case of a cache-hit, the time required to serve the service request was approximately 0.02s. These time measurements were averaged over 10 samples. The increased time efficiency of the system in the HADA is clearly perceived.

6. CONCLUSIONS

We have presented the Hybrid Access Decision Architecture for BACS, bringing together centralized and distributed access control approaches. A purely distributed model for access control provides more scalability and efficiency. However, it has a limitation in that the device can only afford to interact with a limited number of other devices as only a small number of policies can be deployed due to the constraints in its resources. A fully centralized approach for access control, although reliable and manageable, it is highly time inefficient as it has various bottlenecks and it poses a single point of failure.

HADA allows BACS devices to intelligently store the most relevant policies. Although, the efficiency is as worse as in the centralized model if there is a cache miss, the efficiency is as good as distributed model for most of the subsequent similar requests. We have also incorporated context awareness in HADA by first partially evaluating the

Size in bytes	Description
1496	General purpose CoAP handler functions
626	CoAP client to get the compact policies from the ACTP
1196	Interpreting and evaluating the policies from local cache
980	Extracting the query parameters from the CoAP request
600	Other helper functions

Table 4: *Code size of the components of the access control layer*

context parameters centrally, then sending the policy to the device to evaluate the remaining context parameters which is local to the device. The binary TLV encoding format for the policies not only reduces the memory footprint of the policies in the cache but also facilitates efficient evaluation and saves significant bandwidth during the policy transfer from the central policy repository to the devices. The design of an access control architecture is not only limited for the applications in BACS but also to general embedded systems. It is concluded that HADA provides a practical solution for efficient and scalable access control in constrained environments.

Our future work includes extending the access control language for further flexibility and analyzing strategies to resolve policy conflicts in the devices that might occur due the the caching strategy or context information. Also, the scalability and resiliency of our approach needs further analysis as a function of the network size, communication topology, and policy size and complexity. This evaluation should also lead to a better understanding of how the combined centralized/distributed nature of HADA can help coping with different types of attacks to the network or parts of it. Finally, we will further extend HADA to allow for dynamic changes of policies and/or policy/right revocation at the management level

REFERENCES

ANSI/ASHRAE 135 (2008) *BACnet - A data communication protocol for building automation and control networks*. Atlanta. GA, USA.

ANSI/EIA/CEA 709.1 (1999) *Control network protocol specification*. Washington, DC, USA.

Cheshire, S. and Krochmal, M. (2012) *DNS-based service discovery*. In IETF Internet Draft.

Corradi, A., Montanari, R., and Tibaldi, D. (2004) Context-based access control management in ubiquitous environments. In: *Proceedings of the Network Computing and Applications, Third IEEE International Symposium, NCA '04. IEEE*. 1 Sept. 2004. 253–260.

Crockford, D. (2006, July) *The application/JSON media type for JavaScript object notation (JSON)*. RFC 4627 (Informational).

Damiani, M., et al. (2009) Spatio-temporal access control: Challenges and applications. In: *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, SACMAT '09*. New York, NY, USA, 175–176.

Dierks, T. and Rescorla, E. (2008) *The transport layer security (TLS) protocol*. In IETF RFC 5246.

Dunkels, A., Gronvall, B., and Voigt, T. (2004) Contikia lightweight and flexible operating system for tiny networked sensors. In: *29th Annual IEEE International Conference on Local Computer Networks*. 455–462.

Ferraiolo, D. F., et al. (2001) Proposed NIST standard for role-based access control. *ACM Trans. Information System Secur. (TISSEC)*, 4 (3). 224–274.

Garcia-Morchon, O. and Wehrle, K. (2010) Modular context-aware access control for medical sensor networks. In: *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies, SACMAT '10*. New York, NY, USA: ACM. 129–138.

Garcia-Morchon, O., et al. (2013) Securing the IP-based internet of things with HIP and DTLS. In: *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*. New York, NY, USA: ACM. 119–124.

Granzer, W., et al. (2006) Security in networked building automation systems. Vienna University of Technology, Inst. Computer Aided Automation, Automation Systems Group.

Granzer, W., Praus, F., and Kastner, W. (2010, Nov.) Security in building automation systems. *IEEE Trans. Ind. Electron.*, 57 (11). 3622–3630.

Grummt, E. and Muller, M. (2008) Fine-grained access control for EPC information services. In: *Proceedings of the 1st International Conference on the Internet of Things*. Berlin, Germany: Springer-Verlag. 35–49.

Hein, P. F. (2001) DALI-a digital addressable lighting interface for lighting electronics. In: *Industry Applications Conference, 2001. Thirty-Sixth IAS Annual Meeting. Conference Record of the 2001 IEEE*. Chicago, IL, USA, 2. 901–905.

Huang, J., et al. (2012) A framework integrating attribute-based policies into role-based access control. In: *Proc. of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12*. June 2012. 187–196.

ISO Standard. 16 484-2 (2004) *Building automation and control systems (BACS) - Part 2: Hardware*. Geneva, Switzerland.

Kastner, W., et al. (2005) Communication systems for building automation and control. *Proc. IEEE*, 93 (6). 1178–1203.

Kent, S. and Atkinson, R. (1998, Nov.) *Security architecture for the internet protocol*. RFC 2401 (Standards).

Kulkarni, D. and Tripathi, A. (2008) Context-aware role-based access control in pervasive computing systems. In: *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies, SACMAT '08*. New York, NY, USA: ACM. 113–122.

Kushalnagar, N., Montenegro, G., and Schumacher, C. (2007, Aug.) *IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals*. RFC 4919 (Informational).

Lampson, B. (1974) Protection. *ACM Operating Syst. Rev.*, 8. 18–24.

MC1322x. (2013) *Mc13224 hardware guide*. Available from: http://mc1322x.devl.org/hardware.html. (21 July 2013).

Megiddo, N. and Modha, D. S. (2003) Arc: A self-tuning, low overhead replacement cache. In: *Proceedings of the 2nd USENIX Conference on File and Storage Technologies*. 115–130.

Montenegro, G., et al. (2007, Sept.) *Transmission of IPv6 packets over IEEE 802.15.4 networks*. RFC 4944 (Proposed Standard). Updated by RFC 6282.

Myers, M., et al. (1999) X. 509 internet public key infrastructure online certificate status protocol-OCSP. Technical report RFC 2560.

Neuman, B. C. and Ts'o, T. (1994) Kerberos: An authentication service for computer networks. *IEEE Commun. Mag.*, 32 (9). 33–38.

Novak, T., Treytl, A., and Palensky, P. (2007) Common approach to functional safety and system security in building automation and control systems. In: *IEEE Conference on Emerging Technologies and Factory Automation, 2007. ETFA*. 1141–1148.

Park, J. and Sandhu, R. (2004, Feb.) The UCONABC usage control model. *ACM Trans. Inf. Syst. Secur.*, 7 (1). 128–174.

Rescorla, E. and Modadugu, N. (2006). *Datagram transport layer security*. In IETF RFC 4347.

Sacramento, V., Endler, M., and Nascimento, F. N. (2005) A privacy service for context-aware mobile computing. In: *Proceedings of the IEEE First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SECURECOMM '05*. 182–193.

Sandhu, R. S. (2012, June) The authorization leap from rights to attributes: Maturation or chaos? In: *Proc. of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12*. 69–70.

Sandhu, R. S., et al. (1996, Feb.) Role-based access control models. *Computer*, 29 (2). 38–47.

Sarikaya, B., et al. (2012, July) Security bootstrapping solution for resource-constrained devices.

Shelby, Z., et al. (2012, Dec.) Constrained application protocol (CoAP).

Steiner, J. G., Neuman, C., and Schiller, J. I. (1988) Kerberos: An authentication service for open network systems. In: *Usenix Conference Proceedings*, 191–202.

Tsang, R. Cyberthreats, vulnerabilities and attacks on scada networks.

Wang, L., Wijesekera, D., and Jajodia, S. (2004) A logic-based framework for attribute based access control. In: *Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering, FMSE '04*. New York, NY, USA, 45–55.

Weber, R. H. (2010) Internet of things-new security and privacy challenges. *Comput. Law Secur. Rev.*, 26 (1). 23–30.