Distributed Fine-Grained Secure Control of Smart Actuators in Internet of Things

Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Hicham Lakhlef

Distributed Fine-Grained Secure Control of Smart Actuators in Internet of Things

Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, Hicham Lakhlef
HEUDIASYC UMR 7253
Sorbonne Universités, Université de Technologie de Compiègne, CNRS
CS 60319 60203 Compiègne Cedex France
{djamel-eddine.kouicem, madjid.bouabdallah, hicham.lakhlef}@hds.utc.fr

***Abstract*—Internet of Things is a new emerging technology that promises a new era of Internet through encompassing seamlessly physical and digital worlds in one single intelligent ecosystem. This goal is achieved by interconnecting a large number of smart objects from the physical word such as smartphones, sensors, robots, connected cars, etc., to Internet. Nowadays, with the advent of Internet of Things, we need efficient mechanisms to remotely control IoT smart actuators by users and controllers using smartphones and IoT devices. This arises particularly in industrial Cyber-Physical Systems to supervise industrial processes. However, the complex environment of IoT systems makes this task very difficult to achieve regarding the number of connected objects and their resource limitation. In this paper, we tackle the problem of remote secure control of IoT actuators. We propose a distributed lightweight fine-grained access control based on Attribute Based Encryption mechanism and one way hash chain. We conducted security analysis and formal verification using AVISPA. The results demonstrated that our scheme is secure against various attacks. Moreover, the simulation results demonstrated the scalability and the efficiency of our solution, which saves substantially energy consumption and computation costs.**

***Keywords*-Secure control, Internet of Things, Attribute Based Encryption, Smart Actuators**

I. Introduction

Nowadays, the development in the field of embedded systems and telecommunications technologies has participated hugely to create what we call Internet of Things (IoT), which is basically a new era of Internet that consists to connect both physical and digital worlds to the Internet. Today, IoT is implicated in several applications such as healthcare domain, smart homes, smart grids, manufacturing systems and so on. IoT promises in the future a new kind of applications where humans will be able to interact seamlessly with physical components using their smartphones. So, one can remotely control several aspects of daily life. This constitutes one of the important trends in several fields. In this paper, we focus basically on some kind of IoT applications where we need to control remotely smart actuators over the Internet using IoT devices such as mobile devices and smartphones. This arises particularly in industrial applications to control manufacturing processes and also in smart home applications to control home appliances such as the control of home alarm systems, the monitoring and the regulation of the temperature level, turning on/off lights, etc. In SCADA based manufacturing systems, there are a tremendous number of smart actuators and sensor devices to monitor and control physical infrastructures. Basically SCADA based control systems are designed in practice in such a way actuators and robots could be accessed and controlled remotely via mobile devices [6], [13].

This idea of remote control was investigated in some previous research works [3], [8], [7], [11] in order to develop efficient protocols. However, most of available solutions have only interested in implementing remote control protocols without addressing the security issues that could occur between mobile devices and remote actuators. Basically, there are some vulnerabilities related to the underlying network between mobile devices and smart actuators. As examples of those vulnerabilities, we can cite, eavesdropping, denial of service, replay attacks and node compromises among others. Therefore, an access control mechanism must be put in place in order to guarantee the execution of sensitive actions by only the uthorized users. Furthermore, existing security solutions are not scalable enough to meet the requirements of large systems such as SCADA systems. Indeed, most of those solutions don't allow to define fine-grained security policies in scalable way to control the access to remote actuators. Contrary to these solutions and in order to overcome all those important issues, we propose in this paper a solution that ensures fine-grained access control.

In this paper, we propose an efficient security protocol to secure the execution of remote actions on smart actuators. In our solution, we define fine-grained privileges to the different users based on their roles in the system. To this end, we take advantage of the Attribute Based Encryption tool to define access policies in a more scalable way. Furthermore, we use one way hash chain as a technique to authenticate IoT devices while preventing against replay attacks. To the best of our knowledge, there is no solution that tackles the problem of secure control of smart actuators in IoT based Cyber-Physical Systems using fine-grained access control mechanisms such as Attributed Based Encryption.

In this work, we evaluated the robustness of our solution in terms of security using AVISPA as a formal verification tool. Basically, the obtained results demonstrated that our protocol is secure under various kind of attacks. Beside the security of our protocol, the performance evaluation using the cryptographic primitives demonstrated its efficiency and its

scalability.

The structure of the paper is organized as follows. We present in the following section the related works. Some backgrounds about Ciphertext-Policy Attribute Based Encryption and one way hash chain techniques are presented in section III. Section IV describes the main architecture and the components of our system. We discuss our solution in section V. Security analysis and formal verification of our protocol are described in the section VI. Section VII contains the performance evaluation of our protocol. Section VIII concludes the paper and outlines the future works.

II. Related works

In this section, we review some secure control based IoT solutions which are closely related to our work.

In [10], the authors proposed a solution to remotely control and monitor home appliances via internal and external mobile devices. The communication between mobile devices and home appliances is done by GSM network and using SMS messages to send commands to the home gateway to perform remote actions. The exchanged control messages are encrypted using AES standard. As a prototype, they developped a Java application to control sensorboxes in smart home environement. The main drawback of this solution is its non scalability since the architecture is not designed to support large number of mobile devices. Therefore, the solution could not be applied in large scale systems such as SCADA systems.

Mantoro et al. [2] proposed an enhanced SSP (Secure simple pairing) for Bluetooth based comunications between smartphones and home appliances in order to control and monitor IoT devices in small areas. Basically, the enhancement consists to improve SSP protocol, designed initially for Bluetooth communications, to resist against man-in-the-middle attack during the connection between smart home device and the mobile device.

Wang et al. [12] proposed a lightweight protocol to secure remote control of IoT devices by portal controllers like smart phones or tablets. The protocol involves three parts: the IoT devices, the portal controllers (smart phones in general) and trust center. The Trust Center is responsible for transfering the control process to legitimate controllers, those later are authenticated against IoT devices based on lightweight hash function and shared keys between IoT devices and controllers. The protocol is resistant to classical attacks such replay, DoS, desynchronization and man in the middle attacks and preserves the privacy of communications like non traceability of control messages between smart devices and controllers. However, the protocol has some scalability issues, since each IoT device needs to share symmetric keys with trust center and all legitimate portal controllers controlling such device. In addition, the protocol generates an overhead regarding the number of exchanged messages.

Compared to the above schemes, our protocol ensures fine-grained access control while minimizing the overhead of message exchanges and cost computation thanks to Attribute Based Encryption and One way hash chain mechanisms.

III. Background

In this section, we provide a brief description about Ciphertext-Policy Attribute Based Encryption mechanism and one-way hash chain, which serve as techniques to design our solution.

A. Ciphertext-Policy Attribute Based Encryption

CP-ABE is a powerful encryption tool, proposed by Bethencourt et al. in [5], to ensure fine-grained access control of data shared by an owner. The idea of CP-ABE is that the encryption is done based on a policy that defines relations between a set of attributes. So, only users that hold a set of attributes that satisfy the policy are able to decrypt ciphertexts. CP-ABE consists of four algorithms [5].

- **Setup**(k): it's a randomized algorithm which is run by the authority. It takes k (security parameter) as a parameter and outputs a master key MK which is kept secret in the authority and a public key PK which is made public and used to encrypt data.
- **Encrypt**(PK, M, Γ): it's a randomized algorithm which is run by the data owner. It takes as parameters the public key PK, the message to encrypt M and the policy Γ. The algorithm outputs the ciphertext CT of the message M encrypted under the policy Γ.
- **Key-Generation**(MK, γ): It's a randomized algorithm, which is run by the authority to generate secret key D_i for each user i based on a set of attributes γ held by that user.
- **Decrypt**(CT, D, PK): it's a deterministic algorithm which is run by the user that wants to decrypt the ciphertext CT based on its secret key D. The user is able to cover the plaintext M only if its secret key D was generated based on a set of attributes γ that satisfies the policy Γ (i.e $\Gamma(\gamma) = 1$). Otherwise, the algorithm outputs $\perp$.

B. One way Hash Chain

One-Way Hash Chain is a powerful technique widely used to authenticate data sources in real-time communications. In this work, we use this technique as lightweight mechanism to authenticate IoT devices and users by IoT actuators in such a way we overcome replay attacks.

Considering a secure one way hash function $H : \{0,1\}^* \rightarrow \{0,1\}^l$, one way hash chain is defined as a sequence of n hash values: $h_1, h_2, \ldots, h_n$ for $n \in N$, where the hash values $h_i \in \{0,1\}^l$, for $1 \leq i \leq n$, are defined as follows:

$$h_k = \begin{cases} H(h_{k-1}) & \text{if} \quad 1 < k \leq n \\ H(m), \ \ m \in \{0,1\}^* & \text{if} \quad k = 1 \end{cases} \tag{1}$$

The hash chain is designed in such a way from a hash value h_i we can compute efficiently the values h_j, for $j > i$, but it is hard to compute the values h_k for $k < i$.

Each value h_i is used as a disposable token to authenticate one user in one period of time. The order of tokens' usage is $h_n, h_{n-1}, ..., h_1$.

IV. Models and security requirements

In this section, we provide our system model and the security requirements that our solution ensures.

A. System model

In our system model, we consider a set of IoT smart actuators $SA = \{SA_1, SA_2, \dots\}$ owned by some owner O_i. We note that we could have multiple owners that manage their IoT smart actuators. However for sack of simplicity, we limit in our discussion to one owner. Our system contains the following entities:

- **Network of actuators**: A set of IoT smart actuators $SA = \{SA_1, SA_2, \dots\}$ that form the control system network. These actuators are deployed in an area of interest and are internet enabled to allow outside users to control them remotely. We can give as an example an IoT based industrial control system where we have a set of IoT components that are remotely controlled by client IoT devices which are situated outside of smart actuators network. Each smart actuator SA_j allows remote execution of a set of actions $A_j = \{A_{j1}, A_{j2}, \dots\}$. Note that actuators are usually not powerful, and thereby we have to design lightweight authentication protocol that is less energy consuming.
- **Gateway**: The gateway is deployed at the edge of the actuators' network which serves as a relay between IoT actuators in the inside of the network and other IoT devices at the outside of the network. It is also charged to manage access control to IoT actuators to execute actions remotely.
- **Client IoT devices**: They consists of a set of different smart devices $D = \{D_1, D_2, \dots\}$ which are used to execute remote actions on smart actuators. We note that these devices are supposed to be numerous and each one of them could handle a set of actions on a subset of actuators $SA_{D_i} \subset SA$ defined as $AA_{D_i} = \bigcup_{j \in SA_{D_i}} DA_{ij}$, where $DA_{ij} \subset A_j$ which is a subset of actions executed by the device D_i on the actuator SA_j.
- **Owner**: The owner is a system component that holds a set of smart actuators. It is responsible to define the appropriate rules for client IoT devices in order to perform remote actions on smart actuators held by this owner. The owner evolves only in the initial steps of the secure control protocol in order to provide legitimate users the necessary tokens to execute actions on smart actuators according the predefined rules.
- **Trusted Authority**: It consists of a central entity which is trusted by all the owners, IoT devices, gateways and IoT actuators in the system. The main goal of the central authority is to bootstrap the system and manage the key materials of IoT devices, actuators, owners and gateways in the system. It operates on offline mode which means that it's not implicated in the subsequent actions after the system initialization.

B. Security model and requirements

In our security model, we consider the following assumptions:

- The central authority is completely trusted by all IoT devices, actuators and also by owners and IoT gateways.
- IoT actuators are not powerful and are not robust against internal attacks trying to comprise them. Inside threats and active attacks like DoS/ DDoS are very challenging to overcome. In this work, we will not discuss mechanisms that deal with these kind of attacks and therefore we assume that IoT actuators are available and are not compromised neither spoofed by other actuators.
- IoT Gateways are supposed to be *honest-but-curious* which means that they follow the protocol properly but they may be curious about users' behaviors and their privacy.
- As follow, we note by PK_{O_i} and D_{O_i} the owner's public and private keys respectively. Likewise, P_G and D_G are the public and the private keys of the gateway G. We assume also that the communications between the gateway G and IoT actuators and between the owner O_i and the gateway are secure (secure channels are established between the different entities).

Under the aforementioned assumptions and the system model previously described, our solution is a lightweight based token access control protocol to execute actions on remote actuators. The following security requirements should be achieved:

- **Authentication of the IoT devices**: In our protocol, IoT actuators must authenticate client IoT devices to execute actions.
- **Respect of privileges**: Each IoT device $D_i \in D$ is able to execute only the permitted actions on each actuator $SA_j \in SA$. In other words, the client D_i can only execute the set of actions DA_{ij} on the actuator SA_j.
- **Replay attacks robustness**: Our authentication protocol must deal with replay attacks. This means that an attacker or a malicious user that controls an IoT device D_j cannot use the token $T_i(t)$ elapsed by some IoT device D_i $(i \neq j)$ at the instant t to get access to the same actuator by using the same token $T_i(t)$ at the instant $t + 1$.

V. Proposed lightweight fine-grained secure control protocol

After introducing the system and security models, in this section, we present in more details our proposed protocol which allows to control remotely the execution of actions on smart actuators. In order to design an adaptative and fine-grained access control protocol, we exploit two main mechanisms which serve as the building blocks of our protocol namely: 1) Cipher-Text Attribute Based Encryption and 2) One way hash chain. Basically, our proposed protocol works in three steps that we sketch in the rest of the section.

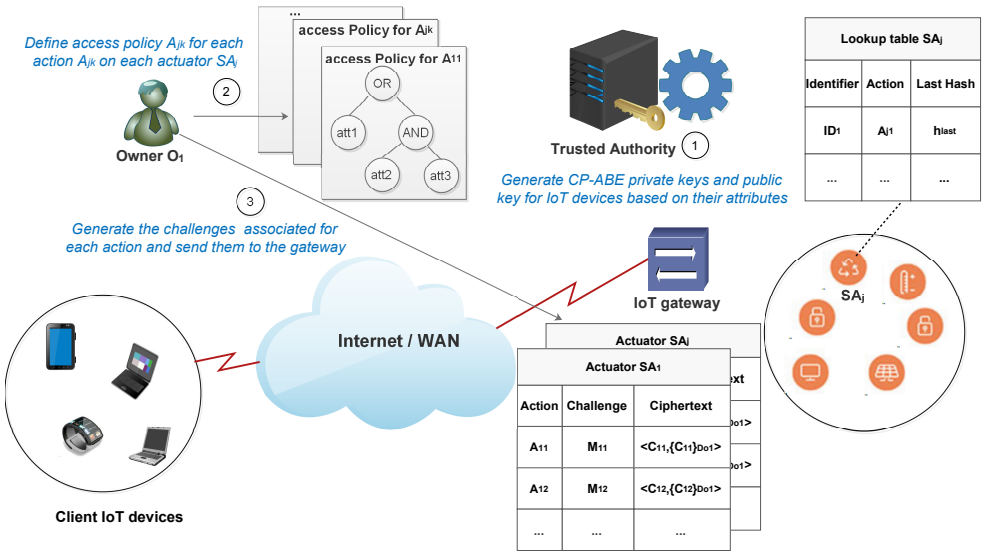

Fig. 1: Secure control protocol.

A. Initialization

In this phase, the trusted authority generates material keys (public and secret keys) for IoT devices based on the roles of the users (the attributes held by each device D_i).

Each owner O_i defines, for each smart actuator SA_j it holds, the adequate policies for all the actions $A_{jk} \in A_j$ that should be executed remotely on that actuator. In other words, we associate for each action $A_{jk} \in A_j$ a policy P_{jk} that defines fine-grained access rules based on some attributes held by the smart IoT devices. The relationships between the attributes are established based on logical **"AND"** and **"OR"** gates as discussed in CP-ABE initial scheme [5].

Subsequently, each owner O_i generates, for each action A_{jk}, a random challenge $M_{jk} \in \{0,1\}^l$. Then, it computes the ciphertext C_{jk} of the challenge M_{jk} encrypted under the policy P_{jk} using the encryption primitive of CP-ABE scheme. All the ciphertexts are also signed by the private key D_{O_i} of the owner O_i. Finally, the owner sends all the challenges and their ciphertexts under the form of a set of Fivepet $\{< SA_j, A_{jk}, M_{jk}, C_{jk}, \{C_{jk}\}_{D_{O_i}} > \text{ where } SA_j \in SA, A_{jk} \in A_j\}$, over a pre-established secure channel, to the IoT gateway G. This later constructs, for each smart actuator SA_j, three columns table that contains, for each action A_{jk}, its corresponding challenge M_{jk} and the paire: ciphertext and signature $< C_{jk}, \{C_{jk}\}_{D_{O_i}} >$ as illustrated in the figure 1.

B. Token generation

In this phase, client IoT devices negotiate tokens from the IoT gateways to execute actions on smart actuators. As depicted in figure 2 (from step 4 to step 12), many steps are carried out by three entities: the client IoT device D_i, the gateway G and the smart actuator SA_j as follows:

- The client IoT device D_i, which wants to execute the action A_{jk} on the actuator SA_j, sends, first, a request message containing the pair $< SA_j, A_{jk} >$ to the gateway G (step 4).
- By consulting the table related to the actuator SA_j, the gateway G generates randomly a nonce value $V_j \in \{0,1\}^*$ and sends back the triplet $< \{C_{jk}\}_{D_{owner}}, C_{jk}, V_j >$ to the smart object D_i, where C_{jk}, $\{C_{jk}\}_{D_{owner}}$ are the ciphertext related to the challenge M_{jk} and its signature respectively (steps 5 and 6).
- The device D_i, by receiving the response from the gateway, verifies the signature $\{C_{jk}\}_{D_{owner}}$ using the owner's public key PK_{owner}. Afterwards, if D_i holds the the CP-ABE private key SK_{D_i} generated based on a set of attributes that satisfies the policy P_{jk}, then it will be able

Natation	Description
O_i	The owner i
D_i	The IoT Device i
SA_j	The smart actuator j
G	The gateway that controls access control to smart actuators
A_{jk}	The action k that could be performed on the actuator j
$H(*)$	One way hash function
$\{M\}_K$	Encryption/Decryption of M with the key K
PK	The CP-ABE public key
MK	The CP-ABE master key. It must be kept secret in the trusted authority
SK_{D_i}	The CP-ABE secret key of the IoT device D_i
M_{jk}	The plaintext used as a challenge to execute the action A_{jk}
P_{jk}	The access policy defined by the owner to execute the action A_{jk}
C_{jk}	The ciphetext of M_{jk} obtained by CP-ABE encryption algorithm and based on the access policy P_{jk}
D_{owner}	The private key of the owner
PK_{owner}	The public key of the owner
D_G	The private key of the gateway G
P_G	The public key of the gateway G
CH_i	The hash chain used by some actuator to authenticate D_i

TABLE I: Notations

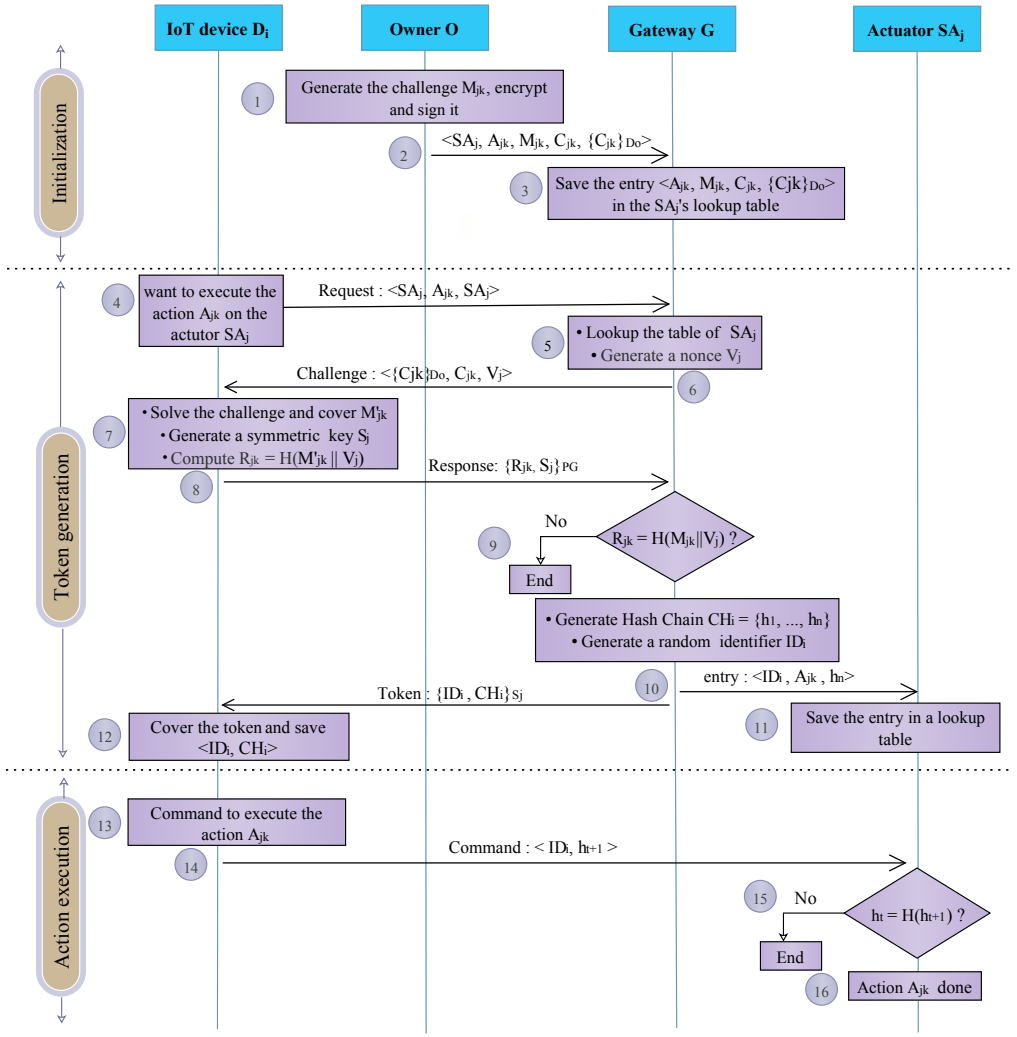

Fig. 2: Flowchart of our protocol.

to decrypt the ciphertext C_{jk} and thereby it covers the plaintext M'_{jk}. This later will be concatenated to the nonce value V_j, let $R_{jk} = H(M'_{jk}||V_j)$ be the resulting hash value. Next, the device D_i generates an AES symmetric key, which will be used as a session key to communicate the token. Finally, it sends the response $< R_{jk}, S_j >$ to the gateway G encrypted under the public key P_G (step 7 and 8).

- The gateway G, upon receiving the response $\{< R_{jk}, S_j >\}_{P_G}$, it decrypts the response message using its private key P_G and it checks out if $H(M_{jk}||V_j) = R_{jk}$. If the condition holds, then the gateway G generates an access token for the device D_i, which consists on the hash chain $CH_i = \{h_{i1}, h_{i2}, \ldots, h_{in}\}$. It generates also an identifier ID_i used to identify and track the action A_{jk} each time it's performed by D_i. Finally, the gateway sends the triplet $< ID_i, A_{jk}, h_{in} >$ to the actuator SA_j and the whole chain CH_i concatenated to the identifier ID_i to the device D_i. Note that the token $< ID_i, CH_i >$ is encrypted by the symmetric key S_j before beeing sent. As illustrated in figure 1, the actuator saves in a lookup table the triplet $< ID_i, A_{jk}, h_{in} >$ which is used to authenticate the device D_i whenever the action A_{jk} is performed (steps 9 and 10).

C. Action execution

In this phase, the actuator SA_j can authenticate remote IoT devices in real-time. The process of authentication is as follows:

- The device D_i sends a request to the actuator SA_j to execute some action A_{jk}. The request includes a pair of information $< ID_i, h_{i(n-t)} >$, where $h_{i(n-t)}$ is the hash value that is used in the t^{th} access and ID_i is the identifier associated to the action A_{jk} previously generated by the gateway G.
- The smart actuator SA_j looks for the entry $< ID_i, A_{jk}, h_{last} >$ in its lookup table. If this entry is found, then it checks out if $H(h_{i(n-t)}) = h_{i(n-t+1)} = h_{last}$. If this condition

holds, the actuator SA_j performs the action A_{jk} sent remotely as a request by D_i. Next, the actuator SA_j updates the value h_{last}, associated to the entry $< ID_i, A_{jk}, h_{last} >$ in the lookup table, by the received one $h_{i(n-t)}$ (see figure 1).

VI. Security evaluation

In this section, we evaluate the security of our scheme. For that, we give some security analysis and we provide formal verification using the AVISPA tool.

A. Security analysis

As follows, we discuss the main security properties that our proposed protocol respects.

1) Authentication: In our protocol, there are two levels of authentication. First, the gateway G authenticates the legitimate IoT devices based on a challenge-response authentication technique. Once the first authentication is done, each smart actuator SA_j authenticates IoT device D_i based on one way hash chain CH_i every time an action A_{jk} is perfomed.

2) Respect of privileges: In our scheme, the execution of each action A_{jk} could be performed only by the legitimate devices. Indeed, the challenge C_{jk} is encrypted in such a way only the legitimate devices have the required keys to decrypt the ciphertext as defined in the access policy P_{jk}. Furthermore, CP-ABE scheme does not allow users' collision , which means that two or many illegitimate IoT devices can't cooperate to construct a secret key that allows them to decrypt the ciphertext [5].

3) Replay attacks: During the token generation process, the gateway generates a random nonce value V_i and sends it to the IoT device D_i with the challenge C_{jk}. The response message contains the value $H(V_i||M_{jk})$ that reveals no information about the plaintext M_{jk}. This response value $H(V_i||M_{jk})$ cannot be used by another IoT device D_j $(i \neq j)$ to get an authorized token to execute the same action, since a fresh random nonce value V_j is generated for each token generation request.

In addition, the execution of each remote action on smart actuator elapses one hash value from the hash chain. Therefore, even thought, an intruder intercepts the hash value $h_{i(n-t)}$ at instant t, it cannot deduce the next token $h_{i(n-t-1)}$ thanks to the irreversible mathematical property of the hash function.

B. Formal verification

1) AVISPA tool: We modeled the specifications of our protocol using the AVISPA's High-Level Protocol Specification Language (***hlpsl***) [1]. The AVISPA tool allows the designers of security protocols to detect potential attacks and verify if their designed protocols meet the attended security services.

2) The protocol specifications: In our protocol, we defined four roles in **HLPSL** language. Namely: the *owner (O)*, the *gateway (P)*, the *device (D)* and the *actuator (A)* roles which correspond to the differents agents in our system. The figure 3 shows the example of the *gateway* role in which we specify the different exchanged messages as explained

```
role proxy (O,P,D,A : agent,
                Kop, Kap : symmetric_key,
                Ko : public_key,
                Hash : function,
                Pkp : public_key,
                Snd, Rcv : channel(dy)) played_by P
def=
  local State  : nat,
        Nd, Rd, Rep, H, Msg, Cipher, SigCipher, IdAction : text,
        HashChain : (text) set,
        Sk : symmetric_key

  init  State := 0 /\ IdAction := ox

  transition
   1. State   = 0 /\ Rcv({Msg'.Cipher'}_Kop.{Cipher'}_inv(Ko)) =|>
      State' := 1 /\ SigCipher' := {Cipher'}_inv(Ko)
                  /\ secret(Mo', sec_mo, {O,P,D})

   2. State   = 1 /\ Rcv({Rd'}_Pkp) /\ Rd' = IdAction =|>
      State' := 2 /\ Nd' := new() /\ Snd(Cipher.SigCipher.Nd')
                  /\ request(P,D,auth_po,Hash(Msg.Nd'))

   3. State   = 2 /\ Rcv({Rep'.Sk'}_Pkp) /\ Rep' = Hash(Mo.Rd) =|>
      State' := 3 /\ H' := Hash(new()) /\ IdAction'= new()
                  /\ HashChain' := {H', Hash(H'), Hash(Hash(H'))}
                  /\ Snd((HashChain'.IdAction')_Sk)
                  /\ Snd({Hash(Hash(H')).IdAction'}_Kap)
                  /\ secret(HashChain',sec_hc1,{O,P})
                  /\ secret(Hash(Hash(H')), sec_hc2, {O,P,A})
end role
```

Fig. 3: **Hlpsl** specifications of the role P (the gateway).

```
goal
 secrecy_of sec_mo
 secrecy_of sec_ch
 authentication_on auth_po
 authentication_on auth_oa
 authentication_on auth_pa
end goal
```

Fig. 4: The hlpsl specifications' goals in our protocol

previously. The channel (*dy*) is modeled in our specifications based on Dolev-Yao intruder model which means that all the exchanged messages between all the agents are intercepted by the intruder. This last can analyze, modify the intercepted messages or eventualy decrypts them if he knows the requiered keys.

In our protocol, we analyze some security properties, which are specified in the goal section of *hlpsl* specifications as shown in figure 4. Basically, we verify the following properties:

- P authenticates D on $H(V_i||M_{jk})$: P generates a nonce value V_i and sends the challenge C_{jk}. If D is able to construct the $H(V_i||M_{jk})$ from the challenge C_{jk} and the nonce V_i, P authenticates D.
- A authenticates D on $H(h_{(n-t)})$: The agent A disposes of the hash value $h_{(n-t+1)}$. If A receives a hash value $h_{(n-t)}$ from the agent D such that $H(h_{(n-t)}) = h_{(n-t+1)}$, A authenticates D.
- Secrecy of CH_i: P generates the hash chain CH_i. It sends the hole chain to the agent D. This information must be kept secrete between P and D.
- Secrecy of h_n: P generates the hash chain CH_i. It sends the last generated hash value h_n in the hole chain CH_i

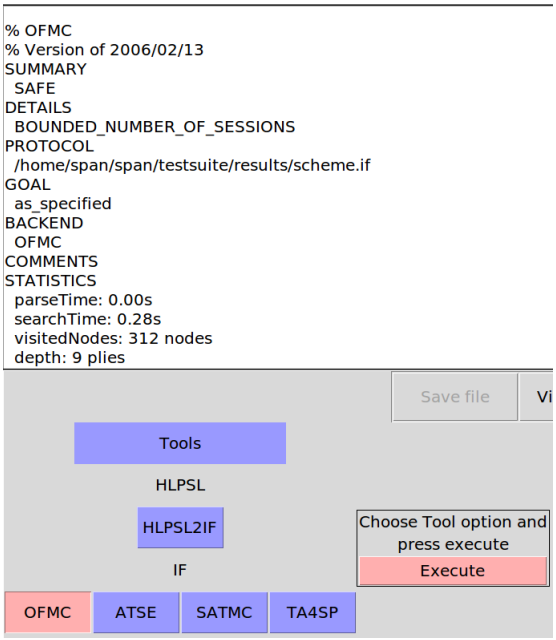

Fig. 5: Results reported by the OFMC back-end

	Computation			Storage (bytes)
	Public key enc/dec	Secret key en/dec	Hash	
gateway G	1	4	p	$1540 \sum_{j \in SA} \|A_j\|$
IoT device D_i	3	1	$p+1$	$32\|CH_i\| + 4$
Actuator SA_j	0	1	p	$36\|CH_i\|$
Owner O_k	$\sum_{j \in SA} \|A_j\|$	$\|SA\|$	-	-

TABLE II: Computation and storage analysis

to the agent A. This value must be kept secrete between S, D and A.

3) The obtained results: We can see clearly from the figure 5 that the obtained results demonstrate the security of our protcol under the test we performed using AVISPA.

VII. Performance analysis

In this section, we present the performance analysis of our scheme. The table II shows the evaluation of our proposed scheme in terms of number of execution of cryptographic operations (enrcyption/ decryption and hash) and the storage occupation with respect to the number of performed actions p by an IoT device D_i on an actuator SA_j. We consider 256 bits as the lenght of each hash value (usage of SHA-256 hash function). Furthermore, we consider the size of each challenge equale to 512 bytes. The size of the identifier of each action is set to 32 bits which allows the owners and the gateway to manage about 4.2 billions of actions on the actuators.

For IoT devices and actuators, we notice that our protocol minimises the number of encryption/decryption operations against an increase of the number of hash computations. Indeed, the number of executions of encryption/decryption algorithms does not increase with respect to p. The number of hash computations increases proportionally to p, but their cost is damn negligible compared to the high cost of encryption/decryption operations.

Challenge Lenght (bytes)	Mean execution time (seconds)
128	0.10343159533
192	0.10304590583
256	0.10351666125
320	0.10426878214
384	0.10431057986
448	0.10558491747
512	0.10519450596
1024	0.10583648784

TABLE III: Time execution of challenge response

A. Experiment settings

We performed these simulations in a virtual machine 64 bits ubuntu 16.04 with 2GB of RAM and with Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz Processor. We used the *cpabe* toolkit implemented by [4], which is based on PBC-0.5.4 pairing library [9] to implement algebraic operations. We used AES as a secret key encryption scheme, SHA256 as a Hash function, and RSA as a public key encryption scheme. We developped the different simulation scenarios based on Python language.

Subsequently, we discuss furthermore the evaluation of the computation cost of our solution, especially at the IoT device level. Basically, we performed two test scenarios with the variation of several parameters in order to evaluate their impacts on our solution.

B. Scenario 1: The evaluation of token generation cost

In this first scenario, we focus on the token generation phase of our protocol. For that, we consider one IoT device D that wants to get a token to execute a certain action on a given actuator. In the test, we compute the execution time that the device D takes to cover the plaintext associated to the challenge and produces the response message. We evaluate especially the impact of the lenght of challenges to be decrypted by D. We assume also that the challenges are encrypted based on CP-ABE encryption algorithm using special access policies that consist of a tree Γ that contains 10 attributes linked with one **"AND"** gate, ie. $\Gamma = AND(att1, ..., att10)$.

The table III shows the mean execution time of challenge response, performed by one device during the token generation phase, with respect to different lenghts of the challenge.

We notice that the execution time elapsed during the token generation phase is not largely influenced by the length of challenges. Basically, with challenges of 512 and 1024 bytes, token generation phase gets a little more time than with the challenges of 128 bytes; therefore, we recommend the usage of challenges of 256 bytes as it's still unbreakable by force brute attacks while maintaining a good efficiency.

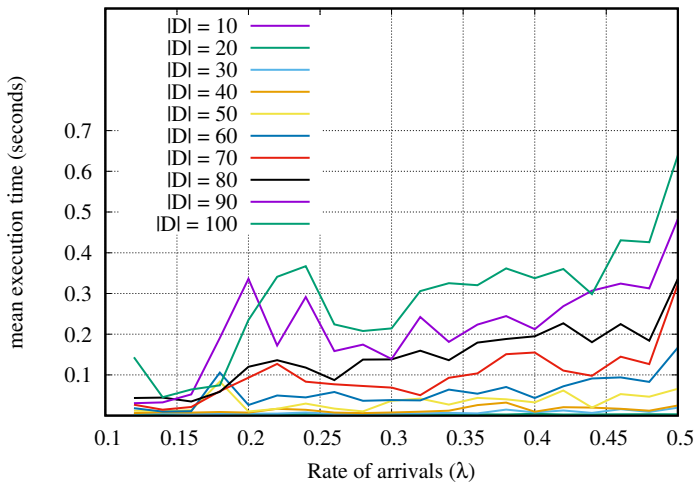

Fig. 6: The mean execution time of action execution with respect to λ and $|D|$.

C. Scenario 2: The impact of the action execution rate and the number of IoT devices

In this scenario, we vary the rate of execution of actions arrivals for several IoT devices. Each device D_i perfoms periodically one action in one actuator. To simplify, we assume that for each two devices D_i and D_j $(i \neq j)$, they execute two actions on two different actuators. In this test, we assume that the gateway can handle one request for only one IoT device at each time, which is the worse case that we can consider [1]. We simulate the rate of action execution requests arrivals according to poisson process of parameter λ. We measure the time execution from event arrival until the execution of the action. Note that, one device D_i executes the token generation algorithm only in the first time in order to get the required token to execute the same action subsequently. Figure 6 shows the average of execution time of token generation obtained by varying both the rate λ and the number of IoT devices $|D|$.

Figure 6 shows that the mean execution time of token generation and action execution phases is not hugely impacted by the number of IoT devices $|D|$ neither by the rate of the execution of actions. Indeed, with $|D| = 100$ IoT devices that execute different actions simultaneously and with a rate of one action per 2 seconds for each device, the execution is about 640 ms. Therefore, our scheme scales very well with complex systems and it could be useful for real time control systems.

VIII. Conclusion

In this paper, we have proposed a distributed protocol to control the execution of remote actions on smart actuators using smart objects and mobile devices. Our protocol has the advantage of being efficient, scalable and allows fine-grained access control. Furthermore, the security analysis, using AVISPA toolkit, showed that our protocol is robust against various attacks. Moreover, we have demonstrated in this paper that our protocol is less energy consuming and scales very well with the number of IoT devices and actuators. The proposed protocol is very suitable for many applications, particularly in resource-limited environments. As future work, it would be interesting to test our scheme in a real application such as SCADA systems to control industrial processes.

[1] In the practice, the gateway can handle several requests simultaneously thanks to multi-threading feature

IX. Acknowledgments

This work was carried out and funded in the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

References

[1] Avispa v1.1 user manual. Technical report, 2006.

[2] R. Acker and M. Massoth. Secure ubiquitous house and facility control solution. In *2010 Fifth International Conference on Internet and Web Applications and Services*, pages 262–267, May 2010.

[3] M. T. Ahammed and P. P. Banik. Home appliances control using mobile phone. In *Advances in Electrical Engineering (ICAEE), 2015 International Conference on*, pages 251–254. IEEE, 2015.

[4] J. Bethencourt, A. Sahai, and B. Waters. Advanced crypto software collection. http://acsc.cs.utexas.edu/cpabe/. Accessed: 2017-03-30.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 321–334, May 2007.

[6] A. A. Cárdenas, S. Amin, and S. Sastry. Research challenges for the security of control systems. In *HotSec*, 2008.

[7] S. R. Das, S. Chita, N. Peterson, B. A. Shirazi, and M. Bhadkamkar. Home automation and security for mobile devices. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 141–146. IEEE, 2011.

[8] O. Ghabar and J. Lu. Remote control and monitoring of smart home facilities via smartphone with wi-fly. IARIA, 2015.

[9] B. Lynn. The pairing-based cryptography library. https://crypto.stanford.edu/pbc/. Accessed: 2017-03-30.

[10] T. Mantoro, M. A. M. Adnan, and M. A. Ayu. Secured communication between mobile devices and smart home appliances. In *2013 International Conference on Advanced Computer Science Applications and Technologies*, pages 429–434, Dec 2013.

[11] F. Pellarin. Communication method and device for remote control of an actuator for mobile equipment in a building, Feb. 23 2016. US Patent 9,269,261.

[12] Z. Wang, H. Ding, J. Han, and J. Zhao. Secure and efficient control transfer for iot devices. *International Journal of Distributed Sensor Networks*, 9(11):503404, january 2013.

[13] S. Żółkiewski and K. Galuszka. Remote control of industry robots using mobile devices. In *New Contributions in Information Systems and Technologies*, pages 323–332. Springer, 2015.