2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering

Privacy-Preserving Distributed Attribute Computation for Usage Control in the Internet of Things

Gianpiero Costantino, Antonio La Marra, Fabio Martinelli, Paolo Mori, Andrea Saracino
Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche
Pisa, Italy
Email: name.surname@iit.cnr.it

***Abstract*—The distributiveness of Internet of Things (IoT) applications, introduces the requirement of a decentralized control system able to enforce security, safety and management policies. Moreover, continuous monitoring of policy conditions might require analysis on attributes whose value could be privacy sensitive. In this paper, we introduce a Usage Control framework for IoT environments which exploits secure multi-party computation and a multi-tiered architecture to collaboratively evaluate policies with privacy sensitive attributes. The paper will discuss a possible smart-grid-based application, showing the benefits of proposed framework, also by presenting performance experiments in an emulated and real environment.**

I. Introduction

Given the increasing popularity and pervasiveness of the Internet of Things (IoT), users are experiencing new services and applications brought by smart and interconnected devices, which are becoming part of daily life [9]. Being interconnected, IoT devices benefit from the possibility of performing collaborative tasks, where information are collected from different devices, shared and exploited to provide services.

An example is provided by smart-meter devices, which are used to monitor the energy consumption inside an house, disabling when necessary some devices that are not needed at the current time. By sharing these information with other devices in the house, it is possible to implement energy saving policies, monitor the overall consumption, performing estimation and avoiding that the energy provider disconnects the house due to surpassing of maximum power threshold. However, this information exchange might bring issues on the side of privacy, especially when the information is shared with entities not belonging to the house. As an example, sticking to the smart-meters use case, it is possible to envision an application in where each house might share information concerning its current energy consumption with the other houses in the neighborhood or condominium, to optimize the overall energy usage of the whole building and minimize the likelihood of power disconnection due to excessive consumption. However, this imposes privacy issues, since other entities, different from the energy provider, are reading private information related to the energy consumption, from which they can be easily inferred other information, such as the time when the tenant is at home.

This paper presents a distributed multi-tier Usage Control framework for distributed IoT systems, which exploits Secure Multi-Party Computation for measuring and testing conditions on collective attributes for Usage Control policies, without disclosing the actual value of these attributes. Hence, specific policies for Usage Control can be written using attribute values evaluated in a privacy aware fashion allowing policy verification and enforcement without data disclosure. The proposed model extends the one presented in [15], by introducing the support for a multi-layered architecture and for collaborative privacy-preserving attribute evaluation. Moreover, this paper improves the preliminary work presented in [4], by focusing on privacy-preserving distribute computations using the Secure Multi-party Computation, and presenting an extensive performance analysis of the proposed framework in an emulated and real test-bed.

A reference example based on energy consumption monitoring in a smart building and smart grid setting is discussed, to show the effectiveness of the tool and the policy writing process. Furthermore a set of performance experiments is reported, in both a simulated and real environment to evaluate under different conditions the impact on performance of the policy evaluation and of the secure multi-party attribute computation.

The rest of the paper is organized as follows: Section II reports background notions about the concepts of Usage Control and of Secure Multi-party Computation. Section III describes the proposed multi-tier architecture which exploits Secure Multi-party Computation to perform privacy preserving policy evaluation. Section IV proposes a smart-grid use case as application for the proposed framework. In Section V we report a description of the performed experiments to evaluate performance in an emulated and real test-bed. A small set of related work is discussed in Section VI. Finally, Section VII concludes the paper by also proposing some future directions.

II. Background

A. Usage Control

The Usage Control (UCON) model [21] extends traditional access control models introducing *mutable attributes* and new decision factors besides *authorizations*: *obligations* and *conditions*. Mutable attributes represent features of subjects,

2324-9013/18/31.00 ©2018 IEEE
DOI 10.1109/TrustCom/BigDataSE.2018.00279

resources, and environment that change their values as a consequence of the normal operation of the system [22]. For instance, some mutable attributes change their values because the policy includes attribute update statements that are executed before (*pre-update*), during (*on-update*), or after (*post-update*) the execution of the access. The e-wallet balance is an example of subject attribute that could be decreased by the policy every time the subject s performs a new access to a resource to charge s for that access.

Since mutable attributes change their values during the usage of a resource, the Usage Control model allows to define policies which are evaluated before *(pre-decision)* and continuously during the access to the resource (*ongoing-decision*). In particular, a Usage Control policy consists of three components: authorizations, conditions and obligations. *Authorization* are predicates which evaluated subject and object attributes, and the actions that the subject requested to perform on the object. Pre-Authorizations are evaluated when the subject requests to access the object, while ongoing-Authorizations are predicates continuously evaluated while the access is in progress. *Obligations* are predicates which define requirements that must be fulfilled before the access (Pre-Obligations), or that must be continuously fulfilled while the access is in progress (ongoing-Obligations). *Conditions* evaluate the attributes of the environment (e.g., current time or workload). Pre-Conditions are evaluated when the subject requests to access the object, while ongoing-conditions are continuously evaluated while the access is in progress.

The continuous evaluation of the policy when the access is in progress is aimed at executing proper countermeasures (such as interrupting the access) when the execution right is no more valid, in order to reduce the risk of misuse of resources.

This paper takes into account Usage Control systems based on the XACML reference architecture [20], with particular reference to the one we presented in [3], [16], which is shown in Figure 1. In the XACML reference architecture, the Policy Enforcement Points (PEPs) embedded in the controlled system intercept the execution of security relevant operations, and they invoke the Context Handler (CH), which is the front-end of the Usage Control system. The *Policy Information Points (PIPs)* are the components invoked by the CH to retrieve

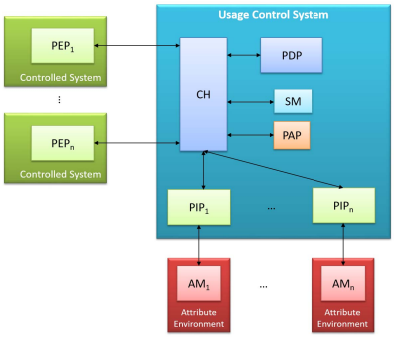

Fig. 1: Usage Control System architecture.

the attributes required by the Policy Decision Point (PDP) for the execution of the decision process, i.e., to evaluate the policy retrieved from the Policy Administration Point (PAP). Attributes are managed by Attribute Managers (AMs). Each specific scenario, where the Usage Control system is exploited, requires its own set of AMs to manage the attributes required for the policy evaluation. Hence, PIPs are properly configured in order to be able to interact with the specific AMs adopted in the scenario of interest for retrieving and updating attributes. In particular, The PIPs are also in charge to detect when the value of an attribute changes to trigger the policy re-evaluation for the involved ongoing accesses, which are managed by the Session Manager (SM). To detect attribute changes, the PIP could exploit the subscription mechanism provided by the AM, or in alternative the PIP emulates the subscription.

The phases of the Usage Control decision process are regulated by the interactions between the PEP and the Usage Control systems as follows (derived from [25]):

TryAccess: it is the *pre-decision* phase, which begins when the Tryaccess message is sent by the PEP to the Usage Control system because a subject requests to execute the access. In this phase, the Usage Control system evaluates the pre-Authorizations and pre-Conditions and, if they are satisfied, it executes the pre-updates. The TryAccess phase finishes when the Usage Control system sends the response to the PEP (PERMIT or DENY);

StartAccess: it is the first part of the *ongoing-decision* phase, which begins when the StartAccess message is sent by the PEP to the CH because the access has just started. The first evaluation of the on-Authorizations and on-Conditions is executed, and the response is sent back to the PEP;

RevokeAccess: this is the second part of the *ongoing-decision* phase. This phase is executed every time an attribute changes its value, and it concerns the re-evaluation of the on-Authorizations and on-Conditions exploiting the new values. If a violation occurred, the RevokeAccess message is sent by the Usage Control system to the PEP;

ResumeAccess: it restarts a session which has been revoked when the current conditions meet again the policy. This function is an extension to the standard UCON model (proposed in [18]), which requires the SM to not delete revoked sessions and PIPs to continue to monitor attributes of revoked sessions;

EndaAccess: it is sent by the PEP to the Usage Control system when the access (s,o,a) terminates normally, for instance because the user closes his access.

B. Secure MultiParty Computation

Secure Multi-party Computation (SMC) is a cryptographic technique to achieve privacy-preserving when running a function *(*f(x,y,z)). In the function *(*f(x,y,z)), three actors want to execute "f" by keeping their input private and not visible to the other. For instance, Alice inputs "x", Bob "y" and Charlie "z" to discover who has the highest input. So, SMC allows the three actors to execute the "maximum" function without disclosing out the input to the other participants.

Secure Multi-party Computation is an extension of Secure-Two party Computation (2PC) where only two participants are required to execute the (f(x,y)) without the involvement of a Trusted Third Party (TTP). Yao in the 1980s [1] illustrated for the time a solution for the "Millionaire Problem" through the of Secure-Two party Computation technique. In the Millionaire Problem, two actors will to discover who is richer without revealing to the other party his/her private salary. This problem is quite simple when is computed in a clear manner since it requires the evaluation of $x < y$ condition, where Alice knows only *"x"*, and Bob knows only *"y"*. However, Yao proposed a solution that allowed the actors to discover who is more reach knowing only the outcome of the condition $x < y$.

In the last fifteen years, researchers have developed new 2PC frameworks aiming at executing functions in a private fashion. FairPlay [17] is the father of recent 2PC framework that allows users to write functions using its high level language (SFDL), and to compile functions into garbled boolean circuits, which will mask the real inputs of both participants. FairPlay has strong security properties in the context of two-party computation. The framework is shown to be secure against a malicious party; in particular *i*) a malicious party cannot learn more information about the other party's input than it can learn from a TTP that computes the function; and *ii*) a malicious party cannot change the output of the computed function. Years later, the same authors of Fairplay published a new versions of their framework with multi-party support called FairplayMP [2]

In the last lustrum, the first version of Fairplay has been ported to Android Smartphones achieving the new framework MobileFairplay [6], [7]. MightBeEvil [12] is a recent 2PC framework too that allows users to write functions that can be run in a private way, in a similar fashion done by Fairplay, however, MightBeEvil outperforms better than Fairplay in terms of velocity and memory.

Another recent 2PC framework is CBMC-GC [11]. It is composed by two main blocks: one is the compiler that translates functions written in C into garbled circuits, instead the other block is formed by the interpreter able to execute compiled functions [14]. Compared with Fairplay, CBMC-GC provides a more flexible high language that allows users to write more sophisticated functions, and leveraging on its optimisation steps during the compilations phase, CBMC-GC runs STC function using less memory than Fairplay.

III. Distributed UCON Framework

The proposed framework relies on a P2P fully distributed Usage Control system, named UCIoT (Usage Control in the Internet of Things), detailed in [15], which is specifically designed to be implemented in IoT architectures, with a focus on Smart-Home environments. The UCIoT framework architecture is based on a set of nodes (e.g., smart devices such as smart Tv or smart-meters in the Smart-Home environment) each of which hosts an instance of the Usage Control System, whose logical architecture matches the UCON framework discussed in Section II. Each node has direct access to a set of attributes, named *local attributes*, and it has relayed access to the local attributes of the other nodes, named *remote attributes*, through a protocol implemented by the UCIoT framework. This allows each device to enforce policies which consider both *local* and *remote attributes*.

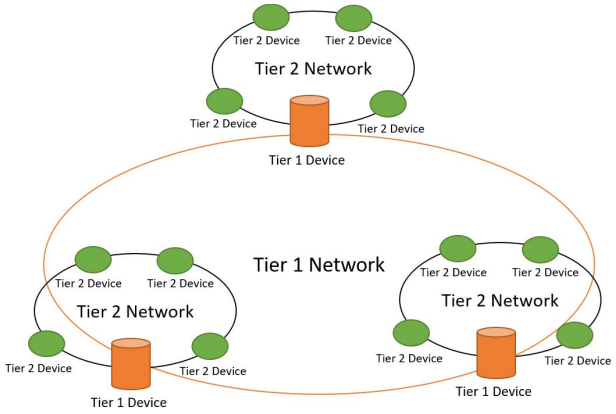

Fig. 2: Two tiers architecture of the proposed framework.

The aim of this paper is to push beyond the features of UCIoT, proposing a two tiers architecture, depicted in Figure 2. As basic component of the two tiers architecture, there is a number of Tier-2 networks. Each of these networks is a set of IoT devices connected each other, e.g., the smart devices installed in an apartment, on which is deployed a standalone UCIoT system. As a matter of fact, all the devices belonging to a Tier 2 Network have their own UCS and PEP(s) and are connected to a set of AMs, typically embedded in to the devices themselves, or physically connected to them, e.g., a power consumption meter, to retrieve the local attributes used for policy evaluation.

The Tier 2 Networks are interconnected each other through a further network, called Tier 1 Network, as shown in Figure 2. To this aim, each Tier 2 Network includes a Bridge device, which is the component that connects this Tier 2 Network with the others. For instance, recalling the previous example, the Tier 2 Networks connecting the smart devices installed in the apartments located in the same building could be connected through a Tier 1 Network, as shown in Figure 4, where the Bridge device is called Smart Home Manager.

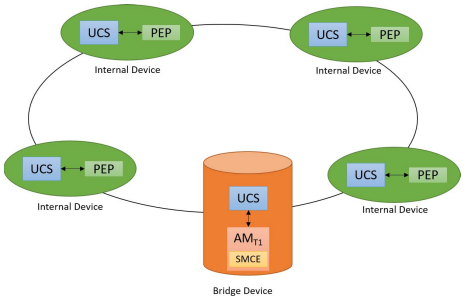

Fig. 3: Internal architecture of the Tier 2 network.

One of the main advantages of the two tiers architecture is that it defines high level attributes, i.e., attributes at the level of Tier 1 Network whose values are obtained by combining the values of the attributes belonging to the Tier 2 Networks. As an example, the energy consumption of each house of a building is a Tier 2 attribute, while the total energy consumption of the building is a Tier 1 attribute. However, to avoid the disclosure of the value of the attributes of each Tier 2 Network to the others, the calculation of the value of Tier 1 attributes is performed exploiting privacy preserving techniques.

The Bridge devices are the components embedding the AMs belonging to the Tier 1, which manage the high level attributes that are related to the system as a whole. To operate such AMs, the Bridges extract information related to the attributes of their specific Tier 2 Networks, and they cooperate to compute the value of the Tier 1 attributes performing privacy preserving collaborative calculation through Secure Multi-Party Computation. To this aim, each Bridge embeds a specific component, called Secure Multi-Party Computation Engine (SMCE), which is in charge of performing the privacy preserving attribute value computation. In other words, at Tier 1 level, the Bridge uses internal attribute values of its own Tier 2 network for computation of results that can be collaboratively exploited by the whole system, without actually disclosing their value.

The next section presents an example of how the SMC technique is exploited to compute the value of an attribute at Tier 1 level. In particular, the Smart Home Managers makes use of the values read from the appliances in their Tier 2 Networks to collaboratively compute at first the total energy consumed in the whole building, and then to verify whether the energy is over a specific threshold.

IV. Application: Energy Consumption Management

As a reference use case for the proposed framework, we consider a smart-grid-like environment, where a group of houses, belonging for example to a neighborhood or to a condominium, have a smart system to handle energy consumptions. In particular, we suppose that the electricity provider sets a maximum energy threshold for the neighborhood (T_{pn}) which, if surpassed, causes the temporary disconnection of the neighborhood from the grid. The energy provider also establishes another threshold for every single house or apartment (T_{ph}) which, if surpassed, causes the temporary disconnection of the single apartment. Since the sum of the thresholds of all the apartments is greater than the neighborhood threshold (in formula: $\sum T_{ph} > T_{pn}$), the consumption of the neighborhood could violate T_{pn} even if the energy consumption of each apartment is lower than its own threshold.

Our framework can be exploited for dynamically controlling and handling the overall energy consumption in order to avoid the temporary disconnection of one house or of the whole neighborhood. In fact, thanks to the SMC technology, it is possible for each apartment to compute the energy consumption of the neighborhood, without disclosing the consumption of each other apartment, which would be considered a privacy violation. To this end, we define for our system two additional thresholds, namely T_{sh} (where $T_{sh} < T_{ph}$) for the house, and T_{sn} (where $T_{sn} < T_{pn}$) for the neighborhood, to be used in the Usage Control policies. The choice to use an additional threshold is motivated by the fact that some energy providers already use a tolerance threshold to avoid the immediate disconnection. In fact, as it happens in Italy[1], the energy provider has a threshold that corresponds to an additional 10% of energy that can be used, allowing the a house/apartment to consume up to 3.3KW of energy for 182 minutes. In particular, we suppose that a smart-meter device is deployed in each house to monitor the current energy consumption of the house and of the neighborhood, making this information available to other smart devices in the house, such as the washing machine, the oven, the heating/cooling systems, etc. Hence, we suppose each smart device in each apartment to be part of the UCIoT framework, which implements a smart-home P2P network through our Usage Control system, as described in [15]. The Usage Control policy installed on specific devices imposes that a device can perform an operation supposed to consume energy as long as the current energy consumption in the house is lower than the threshold T_{sh} or it is lower than the threshold T_{ph} and the energy consumption of the neighborhood is lower than the threshold T_{sn}. In particular, the Usage Control policy allows each device to be activated not performing controls at activation time, to avoid delays in activation, since this would negatively affect the user experience (no predicates are present in the pre-Authorization section of the policy), and the previously described checks are included in the ongoing-Authorization section. Hence, as soon as the current energy consumption of the apartment is greater than the threshold T_{ph}, or it is greater than T_{sh} and the current energy consumption of the neighborhood is greater than the threshold T_{sn}, the device is deactivated. In the hypothesis that the majority of the device causing a considerable energy consumption in the house, are smart objects belonging to the UCIoT framework, the likelihood of disconnection due to surpassing T_{ph} is minimized. In support of this hypothesis, it is worth noting that, even for non-smart devices, is possible to enforce the energy control by means of smart plugs[2]. To ensure usability, we introduce the possibility to differentiate among devices that require a continuous energy supply and that should not be disconnected, such as a refrigerator, and those devices that are supposed to be working only for a limited amount of time per day, e.g., washing machine or vacuum cleaner, and whose usage can be delayed or interrupted and resumed. These two sets of devices will thus have different level of priority. For the sake of simplicity in the following we will consider two priority levels: `Low Priority`, to represent those devices which can be suspended or interrupted and `High Priority` to represent those devices requiring continuity. However, the example can be easily generalized to different priority levels,

[1] http://www.asmb.it/export/sites/asmb/_downloads/it/energia-elettrica/Informazioni/Il-nuovo-contatore-elettronico.pdf (Only Italian version)

[2] http://www.tp-link.it/products/details/cat-5258_HS100.html

to accommodate an higher flexibility.

The distributed architecture that describes the previous example in the Smart Home scenario is depicted in Figure 4. In the figure, each apartment is in fact represented by one cluster of Smart Devices. The orange device, named Smart-Home Manager *(SHM)* is the Bridge component that acts as interface between the apartment and the neighborhood. Each Smart-Home Manager is part of the UCIoT framework internal to the house, and it manages two Attribute Managers, one to compute the current energy consumption of the house and one to check if the current energy consumption of the neighborhood violates the threshold T_{sn}. The SHMs of the neighborhood constitute themselves a P2P network (which is the Tier 1 network according to the definition we gave in Section III), which can be used for joint computations, such as the one of total used energy. This computation becomes challenging if privacy is required; for instance, the energy consumed by each house should not be publicly visible to other. To this end, the computation of global energy consumed and the comparison with the threshold T_{sn} is performed by means of Secure Multi-Party Computation. The sum and comparison is implemented by means of the Fairplay framework (described in Section II-B) that outputs a binary decision: *TRUE* if the current overall neighborhood consumption is lower than T_{sn}, *FALSE* otherwise. This binary value will be used as attribute for the UCON policy of the various smart devices, and the function computing it is named `UnderThreshold`. Hence, the enforced UCON policy is the one shown in Listing 1.

Listing 1: Example of Usage Control Policy

```
1  policy-devices-with-low-priority:
2     target:
3        (resource.id = "c140e234f784")
4        (action.id = "TurnOn")
5     pre-Authorization:
6        TRUE
7     pre-Condition:
8        (e.dateTime ∈ DAILY_HOURS)
9     pre-Update
10       numberOfLowPriorityDeviceOn++
11    ongoing-Authorization:
12       (CurrentHomeCons) ≤ Tsh) OR
13       ( (CurrentHomeCons) ≤ Tph) AND
14         (Underthreshold = TRUE) )
15    ongoing-Condition:
16       (e.dateTime ∈ DAILY_HOURS)
17    post-Update
18       numberOfLowPriorityDeviceOn--
19    .......
```

The target section of the policy (lines 2-4 of Listing 1) specifies that this policy can be applied to the resource with ID "c140e234f784" and it regulates the execution of the action with ID "TurnOn". The pre-Authorization section (lines 5-6) always returns TRUE. This means that the user is initially always allowed to turn on a device. The ongoing-Authorization section (lines 11-14) reports the predicates that must be valid while the device is in use. In particular, the ongoing-Authorization section is a disjunction of two predicates. The first of these predicates (line 12) checks whether the current consumption of the apartment (which includes the consumption of the device to which the policy refers) is lower than T_{sh}, which is the safety threshold of the apartment. If this predicate is satisfied, the ongoing-Authorization section authorizes to prosecute the use of the device. Instead, if this predicate is not satisfied, the second predicate of the disjunction (lines 13-14) is evaluated. The second predicate is, in turn, a conjunction of predicates that require that the current consumption of the apartment (including the current consumption of the device) is lower than T_{ph} (line 13), which is the provider threshold of the apartment, and that the global consumption of the neighborhood, which includes the current consumption of the device, must be lower than the safety threshold imposed for the neighborhood (T_{sn}). We recall that the estimation of the global consumption of the neighborhood is performed through SMC. If the second predicate is satisfied, the ongoing-Authorization section authorizes the prosecution of the use of the device. Otherwise if both the predicates of the disjunction are not satisfied, the policy is violated, and the device is turned off.

Moreover, the pre-Condition section (lines 7-8) and the ongoing-Condition one (lines 15-16) state that this device can be switched on and used only during the day time, for instance because it is very noisy (such as a vacuum cleaner). Finally, the pre- and post-Update sections are required to implement priority based policies, as explained in the following. The enforcement is performed directly on the single Smart Devices.

It is possible to implement finer grained policies and consider different level of priorities for the house appliances in order to select which devices should be turned off for first in case the house threshold has been surpassed. Hence, a priority value is paired to all devices. Let us suppose to define two priority values: high and low. The policy in Listing 2 represents the policy enforced in case of devices belonging to the high priority device set, while the policy in Listing 1 can be considered as the policy enforced in case of device with low priority.

Listing 2: Example of Usage Control Policy

```
policy-devices-with-high-priority:                                    1
   target:                                                            2
      (resource.id = "e156a213b785")                                  3
      (action.id = "TurnOn")                                          4
   pre-Authorization:                                                 5
       TRUE                                                           6
   pre-Condition:                                                     7
      (e.dateTime ∈ DAILY_HOURS)                                      8
   ongoing-Authorization:                                             9
      (numberOfLowPriorityDeviceOn > 0) OR                            10
      ( (CurrentHomeCons+action.consumption) ≤ Tsh) OR                11
          ( (CurrentHomeCons+action.consumption) ≤ Tph) AND           12
            (Underthreshold(action.consumption) = TRUE) ) )           13
   ongoing-Condition:                                                 14
      (e.dateTime ∈ DAILY_HOURS)                                      15
   .......                                                            16
```

The pre-Authorization and the pre-/ongoing-Condition sections of the policy for high priority devices are the same as the ones for the policy for low priority devices. The main difference between the two policies is in the ongoing-

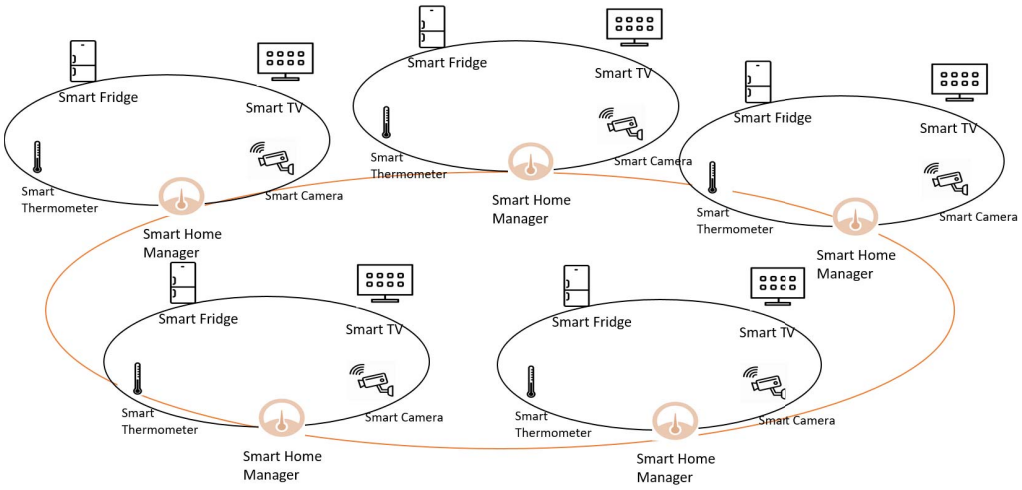

Fig. 4: Overall system architecture for the energy consumption use case.

Authorization section. As a matter of fact, in the policy for low priority devices, the ongoing-Authorization section states that, if one of the two threshold is violated, the ongoing-Authorization section is violated and the device is switched off immediately. In the policy for high priority devices, instead, the ongoing-Authorization section includes a further predicate (line 10 of Listing 2) that causes the section to be violated only when one of the threshold is violated and no low priority devices are currently switched on. This causes the devices with high priority to be switched off by the policy only after that all the low priority devices have already been switched off. To this aim, we added to the policies a new attribute, called numberOfLowPriorityDeviceOn, which counts the number of the low priority devices which are currently switched on. Moreover, the low priority device policy includes a pre-Update section which increases by 1 the number of low devices that are currently switched on when a access request is permitted (lines 9-10 of Listing 1), and a post-Update which decreases the number of devices that are currently switched on when a device is switched off because of a revocation due to a policy violation or because the user switched off the device (lines 17-18 of Listing 1).

V. Experiments

To demonstrate the feasibility of the proposed approach, we have evaluated the system performance for a simulated system composed of a Tier 2 network made of four smart-devices plus one Smart-Home Manager (Tier 1 device), e.g., smart-meter [5]. In addition, a network composed of Tier 1 devices is built to evaluate whether the sum of the energy values of all SHMs is over a specific threshold. The experiments have been performed in an emulated and in a real scenario representing the same architecture of the energy consumption use case previously described.

In the real scenario, a SHM is represented by a Raspberry PI 3 and each device runs an instance of the UCS. Tier 1 devices run the Secure Multi-Party Computation to compute the total consumption of energy and then to verify if this is over a specific threshold. Each SHM is represented by the following hardware specification: Quad Cortex A53@1.52GHz, 1GB SDRAM, a 802.11n wireless connectivity and a SHM consumes at maximum 10Watts of energy.

The smart-devices as well as the SHM are connected following a p2p wireless architecture. In this way, the Tier 2 devices belong to the same network, instead the Tier 1 devices are part of a different network. This separation, however, does not prevent that a SHM does not communicate with the smart-devices belonging to the same house/apartment.

In our experiments, the smart-devices and the smart-home manager can reach each other using the IP of the device, which is set a recipient. The wireless network is build exploiting an ad-hoc infrastructure. In this way, there is not a central device that acts as an access point and all devices are in the same layer creating in this was a p2p network.

A. SMC at Tier 1

Each SHM is able to run an instance of SMC to compute the function to sum all energy consumed by the smart-devices available in the house/apartment in a privacy-preserving fashion, and then to verify whether the sum is higher then a threshold. Executing a function in SMC is, however, a heavy task due to its cryptographic primitives and as consequence usage of memory. A SHM contains the boolean circuits needed to execute the Secure Multi-Party Computation function that represents the sum of all energy values provided as input by the SHMs. This means that inside each house/apartment, a smart-devices provides to the corresponding Smart-Home Manager, the value of energy that is “observing” and this is summed with the energy “observed” by others smart-devices belonging to the same house/apartment. The sum of these values is recorded by the SHM that in a sub-sequent phase will provide this value to establish the sum among all SHMs using the SMC funtion. In the same fashion, the other SHMs, which belong to Tier 1, provide their recorded energy values, and at the end of

the SMC function, the sum of all recorded energy values for all houses/apartments is obtained keeping the privacy of each input value provided by the SHM.

Listing 3: Calculating the total-energy consumption of SHM in SFDL and comparing with the threshold

```
1  program energy_sum {
2    const nHomes = 5;
3    type int = Int<8>;
4    type Smart_Home_Manager = struct{int input, int output};
5    function void main(Smart_Home_Manager[nHomes]
         smart_home_manager) {
6      var int totalEnergy;
7      var int threshold;
8      var int isOverThreshold; //it indicates if totalEnergy
           is higher than the threshold
9      threshold = 15; // Total_WATT for the
           neighborhood/condominium
10     isOverThreshold = 0;
11     for(i = 0 to nHomes - 1) {
12       totalEnergy = totalEnergy +
             smart_home_manager[i].input;
13     }
14     if(totalEnergy > threshold){
15       isOverThreshold = 1;
16     } else {
17       isOverThreshold = 0;
18     }
19     for(i = 0 to nHomes - 1) {
20       smart_home_manager[i].output = isOverThreshold;
21       }}}
```

In the Listing 3 it is shown the high-level program written in SFDL for FairPlay, which is the SMC framework used in our experiments. The function considers five SHMs, which provide as input their recorded energy. Line 12 indicates the inputs provided by each SHM, then these pieces of information are summed and stored in the *totalEnergy* variable. Once, all input values are collected, then the comparison with the threshold is verified at line 14. If the total "totalEnergy" variable is higher then the threshold, then the *isOverThreshold* variable is set to 1, otherwise to 0, and this information is sent for acknowledgement to the SHMs (line 20).

1) Performances: The execution of SMC functions on light-weighted devices liked the Raspberry Pi 3 is a heavy task. In our experiments, we collected the time that the function showed in Listing 3 required to be completed. In particular, we have observed that to conclude the sum plus comparison function of five SHMs, the time needed is of $\sim 25s$. This time is obtained considering variables of 8bits in the SMC function. This means that each SHM can submit at maximum a value of energy of $2^8 = 256$. Since on a real scenario we consider that each house/apartment consumes at maximum $\sim$3000Watts of energy, the $2^8 = 256$ value is not enough to represent the desired values. For this reason, in our experiment we consider that each SHM provides as input, i.e., line 12 in in the Listing 3, a value of energy that is $\lfloor(\frac{E_c}{100}) + 0.5\rfloor$, where E_c represents the input value of the SHM.

To conclude this performances analysis, to the time required by the SMC, the time of the policy evaluation of the UCIoT framework must be considered.

B. Emulated Scenario

Experiments have been replicated five times to extract minimum, maximum and average time for the main Usage Control operations introduced in Section II. The emulated experiments are aimed at measuring the evaluation timing without any delay introduced by the network, by running the experiments on a set of 25 virtual machines to replicate the environment represented in Figure 4. Each virtual machine has Ubuntu 16.04 64-bit, having a dedicated core of an Intel i7-6700HQ with 8 cores, and 1 GB of DDR4 RAM running in 2133MHz. For this set of experiments five physical machines have been used, with five VMs each. Hence, each physical machine represented a single smart-home with five devices, including the Smart Home Manager. The physical machines were interconnected via a LAN switch. For the real settings a set of 25 Raspberry (RB) Pi 3[3] has been used. Each RB presents the following technical specifications: Broadcom ARMv7 Quad Core Processor running on 1.2GHz and 1GB of LPDDR2 RAM on 900MHz, running Raspbian as operative system. Five RBs have been used as Smart Home Manager, hence they run the SMC protocol and use two network interfaces: (i) the Ethernet, which is used to interconnect the Smart Home Managers among them, (ii) the WiFi which is used to connect with the other RBs belonging to the same smart home. A WiFi router has been used as connection hot-spot for each smart home, whilst an Ethernet switch is used to connect the five Smart Home Managers.

Table I reports the average timings (in ms) for the emulated environments, extracted from five different requests issued by a PEP for a device in each network. As can be observed, the timings are comparable, in the order of 1 or 2 seconds. The policy evaluated considers a single attribute, i.e. the value of energy consumption in the smart home matched with T_{uh}. Table II shows instead the timings (again in ms) in the real settings. The experiments have been performed using the same methodology of the emulated ones.

TABLE I: Policy evaluation times on the emulated system

Action	Min	Max	Avg
`TryAccess`	1684	2137	1868
`StartAccess`	847	1190	1585
`RevokeAccess`	902	1795	1064

TABLE II: Policy evaluation times on real architecture.

Action	Min	Max	Avg
`TryAccess`	2335	3132	2424
`StartAccess`	1421	1697	1446
`RevokeAccess`	1269	1360	1277

As expected the timings are slightly longer, due to the more limited computational power of the RBs and to the presence of the WiFi delay. However, the timings are still comparable with the ones of the previous set of experiments. This delay should not have a consistent impact on the user experience.

[3]http://raspberry.org

VI. Related Work

Previous works [10], [19], [23] pointed out privacy issues that may happen when smart-meters are used to monitor the usage of energy in a house or apartment. To reduce or mitigate the privacy issues, the authors of [5] presented a solution based on Secure Two-Party Computation (2PC) to establish the energy consumption of a smart-meter from an energy provider by keeping private the energy consumed in the house/apartment. The authors of [24] showed an architecture formed by smart-meters that use the Secure Multi-Party computation technique to manage the real time energy load. The SMC allows to compute a secure functions, for instance the summation, by keeping private the input of each smart-meter. The authors designed the SMC component using homomorphic encryption, and in particular the secure summation exploits the Paillier Cryptosystem. Danezis et al. in [8] designed a protocol for smart-meter to encrypt readings adopting an efficient secret-sharing-based secure multi-party computation techniques. In [13] a bandwidth-limited protocol based on SMC that allows aggregating energy consumption of smart meters is presented.

VII. Conclusion and Future Work

Usage Control thanks to the policy language expressiveness and its enforcement flexibility, becomes an enabler technology for smart services, in particular when different parties are involved. In this paper, we have presented the application of a distributed Usage Control framework for smart services, applied to a 2-tier smart grid environment, where shared information also imply a privacy issue. In particular, we have discussed the application of secure multi-party computation for privacy preserving policy evaluation, discussing the feasibility and the performance on both an emulated and real setting. As a future work, we plan to extend the SMC analysis to more complex operations, exploring also settings in which homomorphic encryption would be a more effective approach. We also plan to extend the test-bed and the proposed model, considering an architecture with a configurable number of tiers and clustered Usage Control models.

Acknowledgement

This work has been partially supported by the H2020 EU funded NeCS (GA 675320) and C3ISP (GA 700294) projects, and by the EIT Digital #18167 Private Virtual Network Federator project.

References

[1] C. Andrew and C. Yao. Protocols for secure computations. In *23rd IEEE Symposium on FOCS*, pages 160 –164, 1982.

[2] A. Ben-David, N. Nisan, and B. Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the CCS Conference*, pages 257–266, New York, NY, USA, 2008. ACM.

[3] E. Carniani, D. D'Arenzo, A. Lazouski, F. Martinelli, and P. Mori. Usage control on cloud systems. *Future Generation Comp. Syst.*, 63:37–55, 2016.

[4] G. Costantino, A. La Marra, F. Martinelli, P. Mori, and A. Saracino. Privacy preserving distributed computation of private attributes for collaborative privacy aware usage control systems. In *Proceedings of The 4th IEEE International Conference on Smart Computing (IEEE SMARTCOMP)*, 2018. To Appear.

[5] G. Costantino and F. Martinelli. Privacy-preserving energy-reading for smart meter. In *Inclusive Smart Cities and e-Health: 13th International Conference on Smart Homes and Health Telematics, ICOST 2015, Proceedings*, pages 165–177. Springer International Publishing, 2015.

[6] G. Costantino, F. Martinelli, and P. Santi. Investigating the Privacy vs. Forwarding Accuracy Tradeoff in Opportunistic Interest-Casting. *Transactions on mobile computing*, 2013.

[7] G. Costantino, F. Martinelli, P. Santi, and D. Amoruso. An implementation of secure two-party computation for smartphones with application to privacy-preserving interest-cast. In *Proceedings of the 18th international conference Mobicom*, pages 447–450. ACM, 2012.

[8] G. Danezis, C. Fournet, M. Kohlweiss, and S. Zanella-Béguelin. Smart meter aggregation via secret-sharing. In *Proceedings of the First ACM Workshop on Smart Energy Grid Security*, SEGS '13, pages 75–80, New York, NY, USA, 2013. ACM.

[9] Gartner. Gartner says 8.4 billion connected things will be in use in 2017, 2017. https://www.gartner.com/newsroom/id/3598917.

[10] U. Greveler, B. Justus, and D. Loehr. Forensic content detection through power consumption. In *IEEE ICC*, pages 6759–6763, June.

[11] A. Holzer, Martin Franz, Stefan K., and H. Veith. Secure two-party computations in ansi c. In *Proceedings of the CCS Conference*, CCS '12, pages 772–783, NY, USA, 2012.

[12] Y. Huang, P. Chapman, and D. Evans. Privacy-preserving applications on smartphones. In *Proceedings of the 6th USENIX conference on Hot topics in security*, HotSec'11, pages 4–4, Berkeley, CA, USA, 2011. USENIX Association.

[13] M. Kirschbaum, T. Plos, and J.-M. Schmidt. On secure multi-party computation in bandwidth-limited smart-meter systems. In *Proceedings of the 2013 International Conference on Availability, Reliability and Security*, ARES '13, pages 230–235. IEEE Computer Society, 2013.

[14] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In *Automata, Languages and Programming*, volume 5126 of *LNCS*, pages 486–498. Springer Berlin Heidelberg, 2008.

[15] A. La Marra, F. Martinelli, P. Mori, and A. Saracino. Implementing usage control in internet of things: A smart home use case. In *Proceedings of The 16th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (IEEE TrustCom-17)*, pages 1056–1063. IEEE Computer Society, 2017.

[16] A. Lazouski, F. Martinelli, and P. Mori. A prototype for enforcing usage control policies based on XACML. In *Trust, Privacy and Security in Digital Business (TrustBus 2012)*, volume 7449 of *LNCS*, pages 79–92. Springer Berlin Heidelberg, 2012.

[17] D. Malkhi, N. Nisan, Benny Pinkas, and Yaron S. Fairplay—a secure two-party computation system. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[18] F. Martinelli, P. Mori, and A. Saracino. Enhancing android permission through usage control: a BYOD use-case. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 2049–2056, 2016.

[19] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the SenSys Workshop (Builsys)*, BuildSys '10, pages 61–66, New York, NY, USA, 2010. ACM.

[20] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0, January 2013.

[21] J. Park and R. Sandhu. The $UCON_{ABC}$ usage control model. *ACM Transactions on Information and System Security*, 7(1):128–174, 2004.

[22] J. Park, X. Zhang, and R. Sandhu. Attribute mutability in usage control. In *Research Directions in Data and Applications Security XVIII, IFIP TC11/WG 11.3 Eighteenth Annual Conference on Data and Applications Security*, pages 15–29, 2004.

[23] E. Leake Quinn. Privacy and the new energy infrastructure. page 43, February 2009.

[24] C. Thoma, T. Cui, and F. Franchetti. Secure multiparty computation based privacy preserving smart metering system. In *2012 North American Power Symposium (NAPS)*, pages 1–6, Sept 2012.

[25] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security*, 8(4):351–387, 2005.