Enhancing Usage Control for Performance: An Architecture for Systems of Systems

Vasileios Gkioulos[1], Athanasios Rizos[2,3], Christina Michailidou[2,3], Paolo Mori[2], Andrea Saracino[2]

[1] Norwegian University of Science and Technology,
Department of Information Security and Communication Technology, Norway
vasileios.gkioulos@ntnu.no
[2] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
name.surname@iit.cnr.it
[3] University of Pisa, Pisa, Italy

Abstract. The distributiveness and heterogeneity of today's systems of systems, such as the Internet of Things (IoT), on-line banking systems, and contemporary emergency information systems, require the integration of access and usage control mechanisms, for managing the right of access both to the corresponding services, and the plethora of information that is generated in a daily basis. Usage Control (UCON) is such a mechanism, allowing the fine-grained policy based management of system resources, based on dynamic monitoring and evaluation of object, subject, and environmental attributes. Yet, as we presented in an earlier article, *a number of improvements can be introduced to the standard model regarding its resilience on active attacks, the simplification of the policy writing, but also in terms of run-time efficiency and scalability.* In this article, we present an enhanced usage control architecture, that was developed for tackling the aforementioned issues. In order to achieve that, a dynamic role allocation system will be added to the existing architecture, alongside with a service grouping functionality which will be based on attribute aggregation. This is structured in accordance to a risk-based framework, which has been developed in order to aggregate the risk values that the individual attributes encapsulate into a unified risk value. These architectural enhancements are utilized in order to improve the resilience, scalability, and run-time efficiency of the existing model.

Keywords: Access Control · Internet of Things · Security Architecture · Systems of Systems · Usage Control.

1 Introduction

Modern interconnected systems of systems, require scalable and efficient security mechanisms, for controlling a very large number of access requests in a future with billions of heterogeneous devices connected to the Internet. The evaluation of requests for access to certain pieces of information and services commonly relies on dedicated policies [9], which incorporate object, subject, and environmental attributes. Such policies are based on predefined rules, while access control is *a process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities*

(users, programs, processes, or other systems) according to that policy [15]. A multitude of access control policies can be defined, corresponding to distinct criteria for what should be allowed and what not [13].

As presented in detail in our earlier study [3], a limitation of access control is that the access request is only checked once, at the initiation, which highlights the lack of capabilities related to checking alterations on the values of attributes during a session so as to re-evaluate the conformance to the policy. This type of continuous control is a feature that Usage Control (UCON) [6] can provide. UCON enhances Attribute-Based Access Control (ABAC) models [2] in two novel aspects [11]: continuity of control, and mutability of attributes. Continuity of control is the evaluation of access decisions not only at request time, but also when the requester executes access rights on the resource. Further, mutability of attributes means that if changes occur in attribute values while a session is in progress, and the security policy is not satisfied anymore, UCON can revoke the access, terminating the usage of the resources [16]. Yet, the examined environments carry inherent limitations in terms of both computational and communications capacity. Accordingly, corresponding optimizations must be implemented to the original UCON design, seeking to maintain operational efficiency at run-time, but also further security objectives related to resilience. Such optimizations must be initially integrated architecturally, and further enhanced within the components of the deployed policy based management systems.

In this article, we build on the results of the aforementioned articles in order to mitigate the limitations of the original UCON which have been presented earlier [3]. Namely, the current UCON architecture, requires the complete re-evaluation of access permissions per user-asset-session triplet, both at the initiation and at runtime. This, have been experimentally proven in the aforementioned articles to require excessive computational resources, especially as the users, assets, sessions and policy attributes increase. Accordingly, we describe the developed architectural optimizations to the original UCON, seeking to positively affect run time efficiency, scalability, and resilience against active attacks. In order to achieve that, a service group functionality is introduced to the existing model alongside with a dynamic role allocation sub-system, both based on risk aggregation. Thus, the right of access will be granted to a user, based on his allocated role for each group of services and not for one service at a time. The integrated optimizations improve the performance of the model, while increasing its resilience by allowing the mitigation of specific types of active attacks that are based on request flooding.

Architectures of this nature can be described in three abstraction levels, maintaining consistency and completeness. These levels are the (i) architectural model and components, (ii) protocol and interface, and (iii) implementation. In this article, we present and discuss the suggested architecture in all three levels (see section 3), highlighting the integrated optimizations to the original UCON and the corresponding affects. The rest of this paper is organized as follows: In section 2, we report related work and background information on the existing UCON mechanisms. Section 3 describes the developed architecture in the aforementioned abstraction levels. Further, section 4 presents our initial results from a small-scale test-case based validation, while Section 5 concludes by

proposing future directions which stem from our preliminary work and validation results.

2 Background and Related Work

In this section we will review the theoretical background of the most commonly used access control models, the Role-Based Access Control (RBAC), the Attribute-Based Access Control (ABAC) and the Usage Control. Furthermore, we provide a brief explanation of the risk-based aggregation process, which will be used in the upcoming sections of this study.

2.1 RBAC - ABAC

Role-Based Access Control (RBAC) is a widespread approach for regulating the access to information and resources [14]. The principal idea of this model is that a set of roles is created based on the application environment, where as an example, these roles can arise from the hierarchy of an organization or a company. Each subject is assigned to a role, depending on which, he/she also is entitled to a set of privileges. Hence, subjects that are higher in the hierarchy have the possibility to perform more actions over the resources, whilst subjects belonging in the base of the hierarchy have limited access. The RBAC model can be characterized as flexible, since subjects can be reassigned to roles if needed and also privileges can be given to roles or taken from them considering the current state of the application environment. Another positive aspect of this model is that subjects can be also organized in groups based on their role or some common characteristics, while each group has its own permissions. As an example, a group can be the IT department of a company with permissions to modify user-names/passwords, but no permissions on changing data related to the salary of the employees.

Notwithstanding the benefits in efficiency [10], RBAC also comes with a certain amount of limitations. The inability to take into account time and location constraints, and the fact that in order to change the privileges of a user the role must be also changed, are a only a few examples commonly discussed in bibliography. Thus, to overcome these limitations, a new model came to fill the gap. Attribute-based Access Control (ABAC) [4] considers many different attributes related both to the subject and the object, in order to grant or deny access to a resource. The sets of attributes that can be evaluated by ABAC include both static attributes such as the name or the role of a subject, and dynamic such as the current position of the subject, the time of the day, the age etc. The right of access is regulated by the security policy, which is defined in accordance to the attributes that need to be evaluated and their permitted ranges. Policies of this type can be expressed in formal languages such as XML [2].

The proposed enhancements in the existing UCON model, arise by the combination of the benefits provided by these two approaches, where attribute based aggregation is utilized both for subject roles and object groups. Consequently, these aggregated values are incorporated within the predefined security policies, reducing the required resources for policy evaluation and accordingly increasing the scalability potential of such deployments.

2.2 Usage Control

The original UCON model is based on the ABAC model. It introduces mutable attributes and new decision factors besides authorizations; these are obligations and conditions. Mutable attributes represent features of subjects, objects, and environment that can change their values as a consequence of the system's operation [11]. Since mutable attributes change their values during the usage of an object, the UCON model allows the definition of policies which are evaluated both at the initiation and during a session. In particular, a UCON policy consists of three components: authorizations, conditions and obligations. Authorizations are predicates which evaluate subject and object attributes, and also the actions that the subject requested to perform on the object. 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). Finally, conditions are requirements that evaluate the attributes of the environment. The continuous evaluation of the policy when the access is in progress aims at interrupting the access when the execution right is no more valid, in order to reduce the risk of misuse of resources.

Hence, in UCON it is crucial to be able to continuously retrieve the updated values of the mutable attributes, in order to perform the run-time evaluation and promptly react to the changes by revoking access when necessary. The main blocks of UCON are the Usage Control System (UCS) surrounded by the Controlled Systems and the Attribute Environment. The Controlled Systems are those components on which the UCON policy can be enforced. Each Controlled System communicates with the UCS issuing the request to access a resource by performing a specific operation on it. For more information about UCON, readers can refer to [6].

Earlier studies on UCON, highlight that in large-scale heterogeneous systems, such as an IoT application [5], the number of attributes can grow exponentially, increasing the demand for resources but also limiting scalability, and run-time efficiency. Accordingly, in this article, enhancements presented have been developed, towards mitigating these limitations and improving the operation of UCON under such constraints.

2.3 Risk aggregation

Large-scale applications create a challenging field in regard to access and usage control. The number of the attributes which need to be evaluated grows continuously and hence, the possibility of mistakes and conflicts during the policy development increases. Therefore, a model has been proposed earlier [8], which considers the risk level that each attribute encapsulates, and aggregates these values for policy decisions. For example, if a subject wants to access a classified document and the policy takes into account the role of the subject, then it is possible to assign different level of risk to different roles, e.g the administrator of the system comes with a low level of risk while a new-hired employee with a high level of risk.

The aforementioned model is a qualitative risk model for systems that make use of UCON, and its goal is to aggregate the risk values of the attributes into one single value, that will characterize the total risk of a given request. In order to achieve the aggregation, the model exploits the Analytic Hierarchy Process (AHP) [12]. Having the

total risk value the security administrator has the possibility to define policies which are based only on this value or, as it will be explained later in this section, policies of any other granularity level. In order to make the functionality of the model clearer a set of definitions must be given [8].

- *Full Policy*: A policy considering the attributes as they are extracted when acquiring the attribute values but not yet aggregated.
- *RA-Policy*: A risk aware policy is a policy which is written by considering the risk level of aggregated attributes. It has generally a smaller number of attributes with respect to the correspondent Full-Policy, hence it is easier to define and evaluate.
- *Initial Request*: A generated request enriched with the related attributes extracted.
- *Aggregated Request*: A request automatically computed by our framework, starting from an initial request, translating it to the aggregation level required by the current RA-Policy.

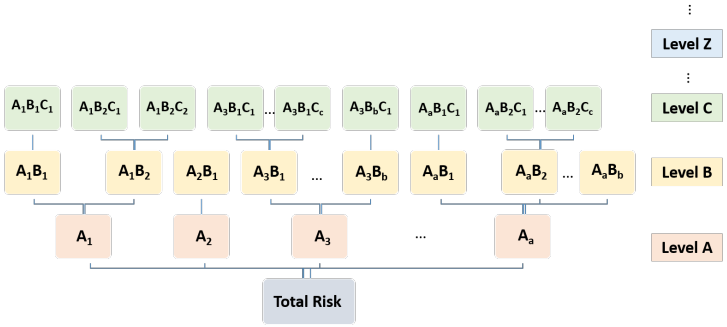

Fig. 1. Total Risk Reverse Tree [8]

The framework is based on a reverse tree structure which is depicted in Figure 1. The total risk value, which was calculated by the aggregation of the attributes' risk values, forms the root of the tree. The upper levels consist of several blocks which represent groups of attributes that are related to each other. For example, a possible group could be the attributes related to environmental factors, such as the location or the time of the request. The leaves of the tree represent the attributes that participate in the Full Policy, whilst the Total Risk value is the one being considered by the RA-Policy.

As stated above, the method used for the aggregation of the risk values of the attributes is the AHP. This method demands the definition of three elements: the *goal*, the *criteria* and the *alternatives*. Regarding the risk-aware model the *goal* is to characterize the total risk of the given request, the *criteria* are the various attributes and the *alternatives* are the possible risk levels (i.e. Low Risk, Medium Risk, High Risk). A set of comparison matrices is created, where an expert on the specific field of the usage control application environment, defines a level of preference among the criteria, stating by this way the relevance of each criterion with respect to the goal.

A comparison matrix is $N \times N$, where N is the number of the alternatives. Each element of the matrix takes a value in the interval [1,...,9] which defines the importance of an element in comparison with another one. Let us consider the previous example of accessing a classified document. Regarding the attribute of the role of the subject, it is reasonable to assume that the administrator of the system can be assigned with a lower level of risk than a new employee. The comparison matrix which represents this statement is shown in Table 1. The meaning of this matrix is that if the value of the role is the *administrator* then the value of *Low Risk* is considered to be 7 times more relevant than the *Medium* and 9 times more relevant than the *High* or *Unacceptable Risk*. On the contrary, if the value of the role is *new employee* then the *High Risk* alternative will be valued more than the others as shown in Table 2.

Table 1. Comparison Matrix of the alternatives for the administrator

Administrator	**Low**	**Medium**	**High**	**Unacceptable**
Low	1	7	9	9
Medium	1/7	1	3	5
High	1/9	1/3	1	1
Unacceptable	1/9	1/5	1	1

Table 2. Comparison Matrix of the alternatives for new employee

New Employee	**Low**	**Medium**	**High**	**Unacceptable**
Low	1	1/4	1/9	1/9
Medium	4	1	1/9	1/9
High	9	9	1	1
Unacceptable	9	9	1	1

Finally, regarding the integration of the risk-aware framework to UCON, there is no need for any modification of the original model. The only requirement is the addition of a set of PIPs, which will acquire the risk values from the AHP blocks. The proposed architecture is shown in Figure 2, where the attributes are grouped into two sets. Each one of the sets will be aggregated using AHP and the results of these aggregations will be the input to a final AHP problem which will compute the single total risk value.

Having this architecture, it is also possible to define policies of different granularity levels, although it must be noted that excessive aggregation levels can affect the expressivity of the policy, as discussed earlier [8]. For example, a policy can be defined by using only the single value of total risk, such as *"Subject can access object if the total risk of the request is at most medium"*, or combine this value with attributes either coming directly from the AMs or coming as outcome from any AHP block, such as *"Subject can access object if the total risk is at most medium and the time of the request is within the working hours"* or *"Subject can access object if the total risk is low and*

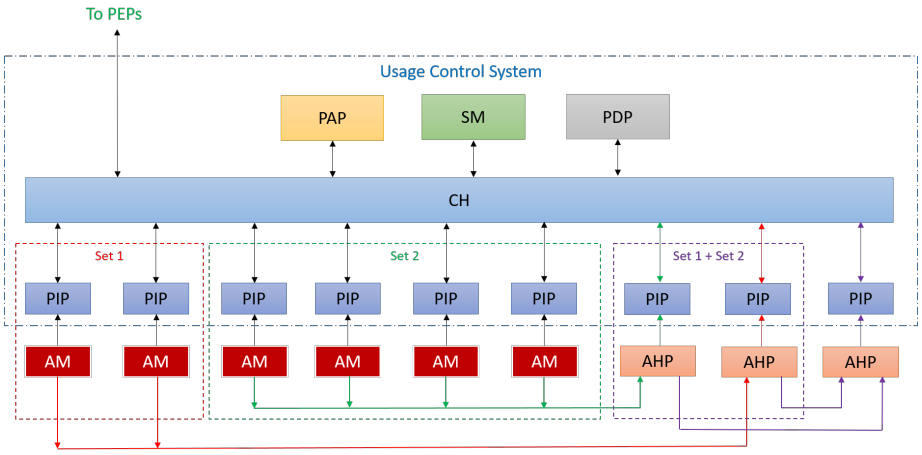

Fig. 2. Risk aware UCON architecture [8]

the risk of the environmental group of attributes is medium". Thus, this model is totally configurable and adjustable to the requirements of the application environment.

3 The Proposed Architecture

In this section, we present the architecture for enhancing the UCON model, in two abstraction levels, namely: (i) the architectural model and its components, (ii) protocol and interface. The aim of this architectural enhancement is to improve the existing UCON model in terms of performance and efficiency. To this end, a service group functionality has been introduced in the current architecture. Alongside the dynamic user role allocation, this functionality gives the possibility for a faster access evaluation and response. Policy attributes are aggregated integrating criticality and risk metrics, allowing for the mapping of service groups but also for the allocation of distinct roles across these groups to every subject. Accordingly, the extraction of the service groups and the current user role (for each group) at run-time is achieved by the Group Handler, and in accordance to the current attribute values. For example, considering that an application environment consists of ten services, the architecture for the enforcement of UCON policies proposed in [7], has to evaluate the subject's request for each one of them. On the contrary, the proposed architecture, after grouping the services, will grant access to these groups in accordance to the predefined policies, and the dynamically allocated user roles, which are independently calculated for each group. Hence, if a user has access to a group, in accordance to his role for this group, and makes a request for a service belonging in this group the evaluation will be faster, improving the run-time efficiency.

3.1 The Architectural Model and its Components

The suggested architecture remains unaltered in comparison to the one proposed in [7], which was based on U-XACML [1], with the exception of the introduction of a Group Handler (GH) as an internal sub-component of the Context Handler (CH), for the purpose of providing high-level compatibility with prior studies and implementations. The components of the architecture and their interconnections are presented in figure 3. The actions used by the PEP to interact with the UCS in order to perform an access request, a start/end of usage of resources are the same as in the UCON model described earlier. The same applies for the actions used by the UCS to interact with the PEP in order to revoke access when needed.

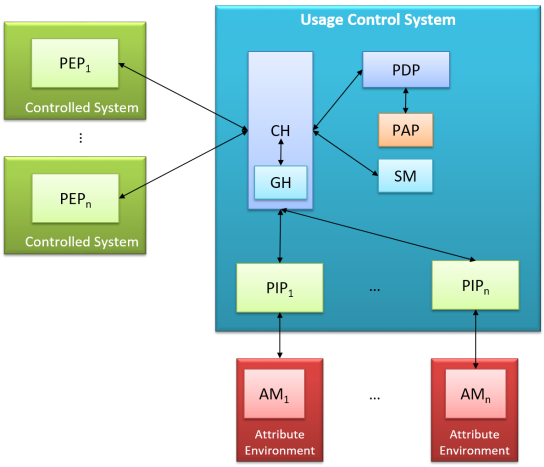

Fig. 3. The proposed architectural model.

The proposed architecture consists of six distinct components. The discrete services provided by these components are:

1. **PEP-Policy Enforcement Point:** The PEP enforces usage control policies by mediating requests from the subscribers to the UCS, and enforcing the corresponding policy decisions. The PEP incorporates functionalities which ensure that no subscriber can register (or remain registered) to a service, without the continuous enforcement of the corresponding usage control policies. Further, the PEP is responsible for the appropriate translation of subscription requests and decisions among the subscribers and the UCS. The communication between the PEP and the UCS is performed via the following actions:
   `TryAccess`: Request by the PEP to the UCS to perform an action or access on a resource. The UCS will respond with a Permit or Deny decision.
   `StartAccess`: This is the actual start of using the service requested. There is again evaluation from the PDP and after an affirmative response from the UCS the session actually starts.

`EndAccess`: This action is invoked when the usage of the resource terminates by a request of the PEP to the UCS.
`RevokeAccess`: If a mutable attribute changes its value and a violation of the policy occurs, the access has to be revoked. UCS informs the PEP that this session is revoked.
A detailed description of the previous interactions can be found in [7].

2. **SM-Session Manager:** The SM is a database of the ongoing sessions. Accordingly, this component is crucial for the (i) session initiation process, (ii) re-evaluation of active sessions process, and (iii) protection against active DoS (Denial of Service) attacks that are based on request flooding. In particular, a new entry, called Tryaccess entry, is created in the SM database every time the initiation of a new access is permitted, as a result of a successful TryAccess. As soon as a StartAccess action is received, the TryAccess entry is updated to ActiveSession entry.
3. **CH-Context Handler:** The CH operates as the controller of the other components, and is responsible for the management and supervision of the session initiation and session re-evaluation processes.
   - **GH-Group Handler:** This sub-component of the CH is responsible for the computation of both the service groups and subscriber roles that correspond to a session, in accordance with the risk aggregation model describer earlier, where the aggregated values of the corresponding attributes, are mapped into such roles and groups. In respect to the services, this computation can be done apriori and in the simplest form integrated as a Look up Table, although the GH can also incorporate the capacity for empirical environmental observation for dynamic service group management at run-time. As for the computation of the user roles, this is done at runtime in two occasions, the initiation of a session for a specific service group and the re-evaluation of access for a specific service group, but not on a per-session basis as in the original model.
4. **PIP-Policy Information Point:** The PIP is the entity which retrieves policy specific attributes from the operational environment, and provides them to the UCS upon request from the CH.
5. **PAP-Policy Administration Point:** The PAP is the entity which is utilized by the system administrators for the development and integration of policies. Moreover, the PAP is in charge of providing the proper policy when necessary.
6. **PDP-Policy Decision Point:** The PDP is the entity, which is responsible for the evaluation of the policy upon request from the CH, and the computation of pertinent decisions.

3.2 Protocol and Interface

In this subsection we provide the sequence diagrams for the session initiation and re-evaluation processes, discussing the operations and providing corresponding examples. For the rest of this Section *Consecutive steps* refer to Figures 4 and 5, which provide the sequence diagrams during the initiation and operation phases in the following scenarios.

1. **Session establishment:** *Consecutive steps: 1-3-4-5:*
   In the initial steps of every session establishment request, the PEP translates the

Fig. 4. Initiation phase-Sequence diagram.

request into a TryAcceess message towards the CH, which includes the unique identifier (Service ID) of the service that the subscriber requests access to. Consequently, the CH extracts the service group which corresponds to the given identifier, in accordance with the service grouping established during deployment, based on the risk aggregation method described earlier. Furthermore, the CH seeks to establish whether the subscriber has initiated similar request for this service, by querying the SM for active TryAccess entries. Provided that the SM replies negatively, therefore this request is not part of an active DoS attack, in step-3 the CH requests from the SM a notification about active sessions for the examined subscriber within the same service group. Given that no such sessions are identified, in step-4 the CH

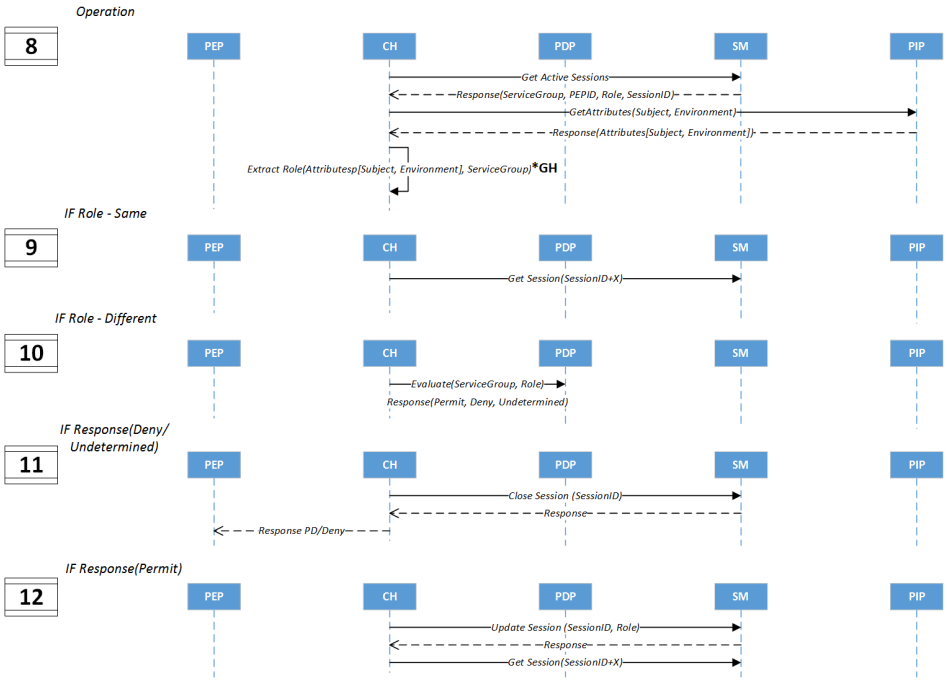

Fig. 5. Operation phase-Sequence diagram.

retrieves the required attributes from the PIP, extracts the subscriber's role that corresponds to the examined service group, and requests a policy evaluation from the PDP, based on the service group and extracted subscriber role. Further, in step-5, given that the permission is granted, the CH requests from the SM to initiate a corresponding session and send a permission notification to the dedicated PEP.

2. **Denial of Service avoidance:** *Consecutive steps: 1-2*:
In this scenario the activities executed for step-1 are identical with those described for the *session establishment* scenario. Yet, given that the SM reports that TryAccess entries are still active for the same subscriber-service pair, (i.e. the time to live has not expired) this request is recognised as part of a DoS-Request-flooding attack, and the request is immediately denied in step-2. This improves the resilience of the usage control architecture, in comparison to the original UCON [7].
3. **Initial session denial:** *Consecutive steps: 1-3-4-6:*
In this scenario the activities executed for step-1, step-3, and step-4 are identical with those described for the *session establishment* scenario. Yet, given that the request is evaluated as "Deny" by the PDP, the PEP is notified accordingly by the CH. It must be noted that in this scenario, the TryAccess entry in the SM remains active for the corresponding time to live, leading to the previously described *Denial of Service avoidance* scenario, if an identical request is delivered within this time to live.

4. **Request for the same service group:** *Consecutive steps:1-3-7:*
   In this scenario the activities executed for step-1 and step-3 are identical with those described for the *session establishment* scenario. Yet, given that the requesting subscriber has and active/permitted session for the examined service group, the CH immediately evaluates the request as ”allow” notifying the corresponding PEP in step-7. This improves both the efficiency and scalability of the usage control architecture, in comparison to the original UCON.
5. **No attribute change:** *Consecutive steps: 8-9:*
   During the session re-evaluation phase, the CH requests the ActiveSessions entry from the SM. Accordingly, the CH requests from the SM the specific information for the first-in-queue session. Based on these information, and the timely values of the corresponding attributes from the PIP, the role of the subscriber is re-evaluated. Given that the role has not been changed, no further action is taken and the CH proceeds to the next-in-queue session, as described in step-9.
6. **Attribute change with permission:** *Consecutive steps: 8-10-12*
   In this scenario the activities executed for step-8 are identical with those described for the *No attribute change* scenario. Given that a change occurred in the subscriber's role, the CH requests and new access evaluation from the PDP, in step-10, and updates the corresponding session entry of the SM in step-12, given that permission is granted by the PDP.
7. **Attribute change with denial:** *Consecutive steps: 8-10-11*
   In this scenario the activities executed for step-8 and step-10 are identical with those described for the *Attribute change with permission* scenario. Yet, given that the policy evaluation result by the PDP is *Deny*, the session in the SM is closed and the corresponding PEP is notified, as described in step-11.

4 Test Case

The test case which has been utilized for the initial evaluation of the proposed architecture, and its comparison with the original UCON, is presented in figure 6. The test case refers to the cloud service deployment of a state owned airport operator, which is distinguished between a global deployment (with three groups of services, whose instances are available across all the managed airports) and a local deployment (with three groups of services in dedicated local instances per airport). The grouping of the services is achieved utilizing the developed risk aggregation method which has been presented in section 2. A set of object, subject, and environmental attributes have been defined for the definition of the corresponding policies, while four distinct types (roles) of users have also been established in accordance to the aforementioned risk aggregation method.

In this section we present the results from one of the executed scenarios within this test case. In this, one of the operators' employees registers and seeks to obtain access for services S1, S2, and S3 of service group 1G. We executed the registration process for this scenario with the original UCON, and the Enhanced-UCON architecture presented in this article, for policies with 1, 5, 10, 15, 20, 25, 30, 35, and 40 attributes. Each test was conducted for ten repetitions, and the average times for the evaluation are presented

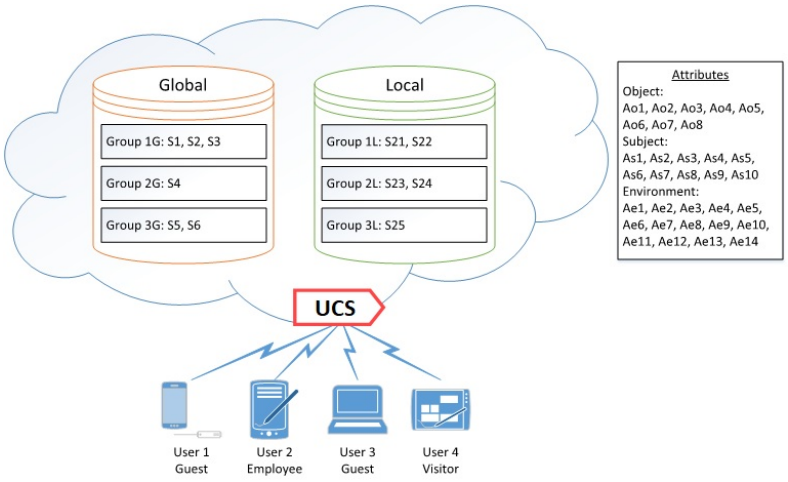

Fig. 6. Exemplified test case scenario.

in table 3 and figure 7. The table presents the elapsed time, in milliseconds, for each of the services, the total time, and the percentage of improvement. The test environment for this scenario was a virtual machine installing Ubuntu 16.04 64 bit, equipped with an Intel i7-6700HQ with 8 cores enabled, 8 GB DDR4 RAM.

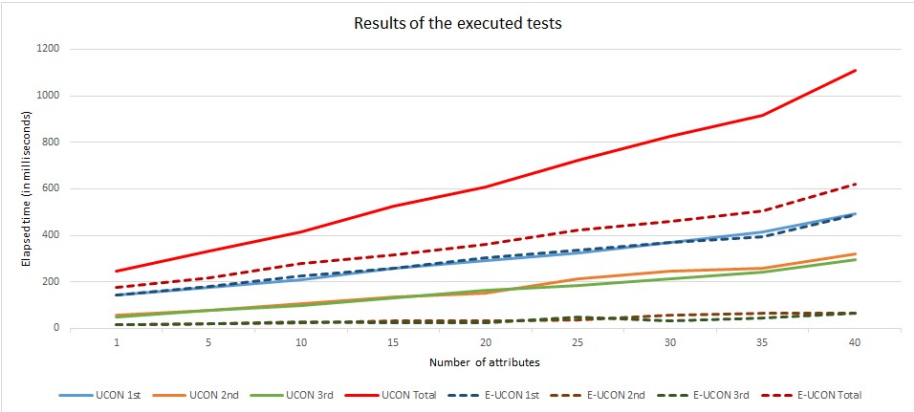

Fig. 7. Results of the executed tests

The results highlight a significant improvement in terms of run-time efficiency, as both the number of micro-services and attributes (incorporated within the security policy) increase. This improvement is not affected by the type or complexity of the service towards which the access request is directed, as the services belong to the same group,

Table 3. Results of the executed tests

Number of attributes	1	5	10	15	20	25	30	35	40
Original UCON-times in milliseconds (ms)									
1st service	141.1	175.3	210.6	256.9	291.2	322.9	367.5	415	493.9
2nd service	56.6	76.9	105.6	134.8	153	211.8	245.7	256.3	318.8
3rd service	48.1	77.6	96.1	132.1	162.4	185.6	211.3	242	294.7
Total time	247.8	331.9	414	525.5	608.9	721.9	827.3	915.7	1110.4
Enhanced UCON-times in milliseconds (ms)									
1st service	142.7	179	223.4	259	303.1	335.2	368.3	394.8	487.6
2nd service	16.7	17.8	23.4	30.1	34	37.3	57.2	63.3	64.7
3rd service	14.4	19.4	28.2	24.2	24.1	48.9	33	44.3	65.5
Total time	175.4	216.9	277.1	315.3	362.5	422.2	461.1	504.2	618.9
Optimization percentage-%									
1st service	1.134	2.111	6.078	0.817	4.087	3.809	0.218	-4.867	-1.276
2nd service	-70.495	-76.853	-77.841	-77.671	-77.778	-82.389	-76.720	-75.302	-79.705
3rd service	-70.062	-75.000	-70.656	-81.681	-85.160	-73.653	-84.382	-81.694	-77.774
Total	-29.217	-34.649	-33.068	-40.000	-40.466	-41.515	-44.264	-44.938	-44.263

for which the users role remain unaltered. A small degradation is noticeable for the initial service registration in low attribute policies, but this is quickly replaced by significant improvement of up to approximately 85%. In total the average performance, across all tests and repetitions, decreases by 1.346% for the first service, while for the second it improves by 77.195%, and for the third by 77.785%. The overall average improvement for three services, across all tests and repetitions, has been 39.154%.

5 Conclusion

In this study an Enhanced-Usage CONtrol (E-UCON) architecture is proposed, where the standard functionality of the model is extended in order to support groups of services and users. This extension aims to improve the model in terms of performance and run-time efficiency, but also to provide the scalability required from the application domain. The mentioned improvements, result from the fact that the right of access will be assigned to user roles towards groups of services and not only in one service at a time, which reduces the evaluation time and the computational requirements. Furthermore, the proposed architecture improves the standard model in terms of security, as it gives the possibility of recognizing and preventing active attacks, such as specific types of Denial of Service based on request flooding. Finally, in this paper a method of simplifying the writing of security policies through the aggregation of the risk values related to individual attributes, is also integrated in the Usage Control model.

The experiments show that the aforementioned enhancements result in significant improvements in performance and evaluation time, especially in realistic deployments with multiple micro-services governed by complex or semi-complex policies. As future work, we intent to develop an extended and heterogeneous test-bed for experimentation, which will be utilized in order to evaluate the performance of the proposed model

in different and more demanding use cases. Moreover, further enhancements will be integrated and tested within E-UCON, initially related to (i) credential management, (ii) trust, and (iii) task delegation.

Acknowledgments. This work has been partially funded by EU Funded project H2020 NeCS, GA #675320.

References

1. Maurizio Colombo, Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. A proposal on enhancing XACML with continuous usage control features. In *Grids, P2P and Services Computing*, pages 133–146. Springer US, 2010.
2. Sabrina De Capitani di Vimercati, Pierangela Samarati, and Sushil Jajodia. Policies, Models, and Languages for Access Control. In *Proceedings of the 4th International Conference on Databases in Networked Information Systems*, DNIS'05, pages 225–237, Berlin, Heidelberg, 2005. Springer-Verlag.
3. Vasileios Gkioulos, Athanasios Rizos, Christina Michailidou, Fabio Martinelli, and Paolo Mori. Enhancing Usage Control for Performance: A Proposal for Systems of Systems. To Appear. In *The International Conference on High Performance Computing and Simulation (HPCS 2018)*, 2018.
4. Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to attribute based access control (ABAC) definition and considerations. *National Institute of Standards and Technology (NIST) Special Publication*, 800(162), 2013.
5. Antonio La Marra, Fabio Martinelli, Paolo Mori, Athanasios Rizos, and Andrea Saracino. Improving MQTT by Inclusion of Usage Control. In *Proceedings of the 10th International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage (SpaCCS)*, SpaCCS '17, 2017.
6. Aliaksandr Lazouski, Fabio Martinelli, and Paolo Mori. Survey: Usage Control in Computer Security: A Survey. *Comput. Sci. Rev.*, 4(2):81–99, May 2010.
7. Aliaksandr Lazouski, Fabio Martinelli, Paolo Mori, and Andrea Saracino. Stateful Data Usage Control for Android Mobile Devices. *International Journal of Information Security*, pages 1–25, 2016.
8. Fabio Martinelli, Christina Michailidou, Paolo Mori, and Andrea Saracino. Too Long, did not Enforce: A Qualitative Hierarchical Risk-Aware Data Usage Control Model for Complex Policies in Distributed Environments. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security, CPSS@AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 27–37, 2018.
9. B. Moore, E. Ellesson, J. Strassner, and A. Westerinen. RFC 3060: Policy Core Information Model – Version 1 Specification, February 2001.
10. Alan C OConnor and Ross J Loomis. 2010 economic nalysis of role-based access control. *NIST, Gaithersburg, MD*, 20899, 2010.
11. Jaehong Park and Ravi Sandhu. The UCONabc Usage Control Model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, February 2004.
12. R.W. Saaty. The analytic hierarchy process - what it is and how it is used. *Mathematical Modelling*, 9(3):161 – 176, 1987.
13. Pierangela Samarati and Sabrina Capitani de Vimercati. *Access Control: Policies, Models, and Mechanisms*, pages 137–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

14. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, February 1996.
15. R. Shirey. RFC 4949: Internet Security Glossary – Version 2, August 2007.
16. Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal Model and Policy Specification of Usage Control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, November 2005.