Cost-Effective Enforcement of Access and Usage Control Policies under Uncertainties

Leanid Krautsevich*, Aliaksandr Lazouski†, Fabio Martinelli†, Artsiom Yautsiukhin†
*Department of Computer Science, University of Pisa, Largo B. Pontecorvo 3, Pisa, Italy
E-mail: krautsev@di.unipi.it
†Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Via G. Moruzzi 1, Pisa, Italy
E-mail: aliaksandr.lazouski@iit.cnr.it, fabio.martinelli@iit.cnr.it, artsiom.yautsiukhin@iit.cnr.it

***Abstract*—In Usage CONtrol (UCON) access decisions rely on mutable attributes. A reference monitor should re-evaluate security policies each time attributes change their values. Identifying all attribute changes in a timely manner is a challenging issue, especially if the attribute provider and the reference monitor reside in different security domains. Some attribute changes might be missed, corrupted, and delayed. As a result, the reference monitor may erroneously grant access to malicious users and forbid it for eligible ones.**

This paper proposes a set of policy enforcement models which help to mitigate the uncertainties associated with mutable attributes. In our model the reference monitor, as usual, evaluates logical predicates over attributes and, additionally, makes some estimates on how much observed attribute values differ from the real state of the world. The final access decision takes into account both factors. We assign costs for granting and revoking access to legitimate and malicious users and compare the proposed policy enforcement models in terms of cost-efficiency.

***Index Terms*—Usage Control, Mutable Attribute, Freshness, Policy Enforcement, Costs, Markov Chains**

I. Introduction

Access control aims to ensure that only trusted principals are granted access to a resource [1]. *Usage control* is responsible for guaranteeing that principals also remain trusted when the access is in progress, i.e. when these principals use the resource. The reference monitor evaluates an access decision on the basis of the principal's attributes. The attributes are issued by the attribute provider and characterize subjects and objects participating in access and usage control [25], [28].

The UCON model proposed by Sandhu et al. [31] includes access and usage control scenarios and operates with mutable attributes to specify and continuously enforce security policies. Access decisions in UCON are based on authorizations (predicates over subject and object attributes), conditions (predicates over environmental attributes), and obligations (actions that must be performed by a requesting subject). The reference monitor in UCON re-evaluates the access decision every time when an attribute changes its value. However, identifying all attribute changes in a timely manner is a challenging issue.

Some security attributes (e.g., the requester's reputation and location) are *remote*, i.e. the attributes reside outside the control of the reference monitor, and can be only observed. These attributes should be constantly *pushed* by the attribute provider (e.g., the requester) or *pulled* by the reference monitor. The system usually allows only the current attribute value to be pulled, and, as a result, some attribute changes between adjacent pull queries might be missed. Worse still, these unnoticed changes may violate security policies. For example, if a security policy grants access rights to users residing in a certain location, there is no evidence that these users did not leave the location in-between checks [8].

In addition, system failures, delays occurring during attribute delivery due to a network latency, as well as malicious activities (e.g., a man-in-the-middle, eavesdropping and impersonating of data by the attribute provider) contribute to the problem of correct policy enforcement. The impact of uncertainties associated with observed attributes should be mitigated by the reference monitor [18], [24].

This paper proposes the cost-effective enforcement models of $UCON_{AC}$ [31] security policies. Our basic idea is to take into account possible uncertainties when an access decision is made. In other words, an uncertainty-aware reference monitor should adjust its decision according to information about the relevant uncertainty. We propose to consider cost-effectiveness as the main criteria for making such a decision. We assign monetary outcomes for granting and revoking access to legitimate and malicious users and compare the proposed policy enforcement models in terms of cost-efficiency.

The main contributions of this paper are:

- to identify uncertainties associated with attributes used to produce access decisions;
- to introduce models of a correct policy enforcement and enforcement under uncertainties;
- to introduce a cost model for policy enforcement and compare the cost-efficiency of proposed enforcement models for access and usage control.

The paper is structured as follows. Section II provides basic notes on UCON. Section III describes the running example that we use throughout the paper. Section IV introduces the model of a mutable attribute, and enlists all types of uncertainties associated with mutable attributes. Section V presents models of correct policy enforcement. Sections VI and VII outline a cost model and estimate an average profit for policy enforcement under uncertainties for access and usage control. Section VIII presents the architecture of the reference monitor for enforcement of policies under uncertainties. Section IX summarizes related works. Section X concludes the paper.

Manuscript received October 01, 2011; revised March 29, 2012.

In Appendix we describe solutions for several computational problems of Markov chains.

II. USAGE CONTROL

Usage control (UCON) [31] requires continuous control over long-standing accesses to computational resources (e.g., an execution of a job in Grid, a run of a virtual machine in Cloud). Continuity of control is a specific feature of UCON intended to operate in a mutable context. The context is formed by attributes of a requesting subject, an accessed object and an execution environment.

An attribute is denoted as $h.r$ where h identifies a subject requesting an object, the object itself, or environment, and r refers to the attribute name. An assignment of an attribute maps its name to a value in its domain V_r, i.e., $h.r = v$, where $v \in V_r$. Without loss of generality, we assume that there is only one attribute in the system denoted as r and that this attribute has a finite domain of values.

Attribute mutability is an important feature of UCON, which means that an attribute can change its value as a result of an access request or another uncontrollable factor. We define the *behaviour of an attribute* as a sequence of values assigned to an attribute with time passage: $\{v_0, v_1, ..., v_i, ...\}$, where v_0 refers to the attribute value when a subject sends an access request, and index $i \in \mathbb{N}$ refers to a time point at which the attribute changes its value. We define a strictly increasing function cl which assigns a real time value to any index, $cl : \mathbb{N} \to \mathbb{R}$.

Access decisions in UCON are based on authorizations (predicates over subject and object attributes), conditions (predicates over environmental attributes), and obligations (actions that must be performed by a requesting subject). We consider security policies consisting of authorization and condition predicates only, that is, the $UCON_{AC}$ model [31]. We define a predicate p to be a boolean-valued computable function mapping an attribute value to either true or false, $p : V_r \to \{true, false\}$.

Another important feature of UCON is that it specifies when access decisions are evaluated and enforced. There are two phases: pre-authorization, or *access control*, and continuous policy enforcement, or *usage control*.

Access control starts at time t_{try} when a user sends a request to the reference monitor. The reference monitor acquires an attribute value v_0, evaluates authorization predicates only once and grants the access at time t_{perm} if $p(v_0) = true$, and $t_{try} = cl(0)$, $t_{try} \leq t_{perm}$.

Usage control begins at time t_{perm} when the attribute takes value v_i, and $cl(i) \leq t_{perm} < cl(i+1)$. The reference monitor re-evaluates authorization predicates each time the attribute changes its value. The access should be continued by time $t_{now} = cl(j)$ only if $p(v_i) \wedge p(v_{i+1}) \wedge ... \wedge p(v_j) = true$. Although usage control ends as a result of the access revocation or at the subject's discretion, w.l.o.g. we consider only the first scenario. When a new value v_k violates the policy, i.e., $p(v_k) = false$, the reference monitor revokes the access. Usage control is over at time $t_{rev} = cl(k)$.

III. RUNNING EXAMPLE

We consider a reputation of a user in Grid as an example of a mutable attribute. The attribute changes its value based on "bad", "good" and "neutral" feedback received from other parties. The attribute domain is $V_r = \{$"general", "normal", "suspicious", "malicious"$\}$. There is a reputation management system (RMS) which measures the reputation value for all users in Grid. Every manager of a resource has an access and usage control system (AUCS) which allows usage of the controlled Grid resources only if the reputation of a user is other than "malicious". Each request for resources is intercepted by the corresponding AUCS. The AUCS (reference monitor) pulls the reputation value from the RMS (attribute provider). If the value is "malicious" the AUCS denies access, otherwise the AUCS grants access to the user. During the usage session the AUCS periodically pulls the reputation from the RMS.

The problem is that, at the time of the access request, a user may be involved in several other jobs for which the RMS has no feedback. In other words, the reputation that a user has at the time of the access request could differ from the real one. The AUCS, which uses only the current version of the reputation, is opened to the following attack. A new user with a good or neutral reputation gets involved in many jobs in a short period of time. The user abuses his rights but, since feedback about his behaviour is only provided at the end of a job, his reputation remains good for some time. During this time the malicious user is able to leverage resources of Grid. The AUCS should take into account the uncertainty which is in the system in order to make a right access decision.

After granting access to some resources the AUCS should monitor the current reputation of the user. Now, during the usage of the resource, the AUCS has another problem which is also rooted in uncertainty. The AUCS has to define how often the reputation of the user has to be requested from the RMS. Checks after every change in the reputation value imply the use of resources and are expensive to perform. Therefore, the AUCS has to define the amount of changes after which the check should be performed. In other words, there is a need for a balance between security and benefits of usage of the resource.

IV. ATTRIBUTE MODEL

Our main concern in this paper is the enforcement of a UCON policy based on a *remote* attribute with *observable mutability*. *Remote* means that an attribute is managed by the attribute provider which is not under the control of the reference monitor. *Observable mutability* means that the reference monitor observes only how the attribute behaves in time. Thus, for the same attribute we distinguish *real attribute values* which truly describe the attribute behaviour in the system and *observed attribute values* which are obtained by the reference monitor and used to evaluate authorization predicates.

A. Real Attribute Values

We assume that a change of the attribute's value can be modelled as a *random event*. Let $\omega : r = v$ denote this event

which happens when the attribute r takes the value v. We define Ω_r to represent a set of all possible events ω. Since the attribute can take *any* value from its domain, there is a one-to-one correspondence between elements of Ω_r and V_r. Each change of an attribute is paired with the value the attribute takes as the result of this change.

In probability theory, it is often easier to deal with a value associated with the random variable rather than with the event itself. Therefore, we introduce a random variable A which gives a numerical description of the event ω. A is a real valued function on Ω_r, that is $A : \Omega_r \rightarrow \mathbb{R}$. The event $A = a$ represents the fact that the attribute r takes the value v, s.t., $A(\omega) = a_\omega$. Let probability of the event to happen be $\mathbf{Pr}[A = a]$. The function $\mathbf{Pr}$ has all properties of a probability function, e.g., for any event E, $0 \leq \mathbf{Pr}[E] \leq 1$. We write $E_1 \cap E_2$ for occurrence of both E_1 and E_2 and write $E_1 \cup E_2$ for the occurrence of either E_1 or E_2 (or both). Let the event $\mathcal{P}(A)$ denote the fact that an attribute takes *any* value which satisfies a policy, i.e., $\mathcal{P}(A) = \bigcup_{\omega \in \Omega_G}(A = a_\omega)$, and $\Omega_G = \{r = v | p(v) = true, v \in V_r\}$. The event $\overline{\mathcal{P}}(A)$ specifies the fact that the attribute takes *any* value which violates the policy. Further, we use A to refer to the attribute value.

Let the behaviour of a *real attribute* be specified by a scheme $\langle \mathbf{A}, \mathbf{CL}_{AP} \rangle$, where:

- $\mathbf{A} = \{A_i : i \in \mathbb{N}\}$ is a discrete-time stochastic process modelling a behaviour of a mutable attribute. We call A_i the state of the process at i, and $A_i = a_i$ denotes that after i changes the attribute value equals a_i;
- $\mathbf{CL}_{AP} = \{cl_{AP}(j) | j \in \mathbb{N}\}$ is an ordered set of timestamps assigned to each attribute change by the attribute provider when it happens. We assume that $cl_{AP}(0) = t_{try}$ and for all $j \geq 1$, $cl_{AP}(j) = cl_{AP}(j-1) + \mathrm{T_j}$, where $\mathrm{T_j} > 0$ and it specifies a time interval between adjacent attribute changes.

Example 1: A reputation attribute may be modelled as a random variable A with values $A(r = \text{“general”}) = 1$, $A(r = \text{“normal”}) = 2$, $A(r = \text{“suspicious”}) = 3$, and $A(r = \text{“malicious”}) = 4$. The mutability of the reputation attribute could be modelled as a discrete-time Markov chain [16], [15] uniquely defined by the one-step transition matrix. Thus, the entry in the i-th row and j-th column is the transition probability $\mathbf{Pr}(A_i = a \mid A_{i-1} = b)$ giving the probability that the attribute changes value to a if its current value is b.

Figure 1 shows the Markov model for our running example with the transition probabilities collected in a transition matrix. These probabilities could be used in order to find whether reputation has a certain value (e.g., $\mathbf{Pr}(A = 2)$). The transition probabilities are taken from the history of changes stored by the RMS and shared with the AUCS:

$$\mathbf{Prob} = \begin{pmatrix} 0.6 & 0.4 & 0.0 & 0.0 \\ 0.5 & 0.3 & 0.2 & 0.0 \\ 0.0 & 0.2 & 0.3 & 0.5 \\ 0.0 & 0.0 & 0.1 & 0.9 \end{pmatrix} \quad (1)$$

B. Observed Attribute Values

Only the attribute provider knows how the attribute behaves in time, but the reference monitor can also observe this

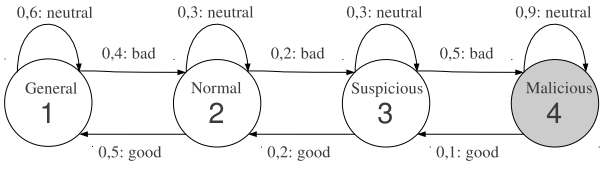

Fig. 1: A Reputation Attribute Model

process. There are two basic models how attribute changes are delivered to the reference monitor: *push* and *pull*. The push model defines a scenario where every new attribute value is timestamped and pushed by the attribute provider to the reference monitor. The pull model defines a scenario where the reference monitor queries the attribute provider to give the current attribute value. The attribute provider replies with the value, its timestamp and some additional information.

By analogy with real attribute values, let *observed attributes* be specified by a scheme $\langle \tilde{\mathbf{A}}, \mathbf{CL}_{RM} \rangle$, where:

- $\tilde{\mathbf{A}} = \{\tilde{A}_i : i \in \mathbb{N}\}$ is a discrete-time stochastic process modelling an observation of attribute changes over time. $\tilde{A}_i = a_i$ denotes that an attribute value after i observations equals a_i;
- $\mathbf{CL}_{RM} = \{cl_{RM}(j) | j \in \mathbb{N}\}$ is an ordered set of timestamps assigned by the *reference monitor*. A timestamp j denotes when the j-th observation of an attribute value was processed and the appropriate access decision was enforced by the reference monitor. We assume that $cl_{RM}(0) = t_{perm}$.

Real and observed attribute values form a bipartite directed graph $\mathrm{W} = (\mathbf{A}, \tilde{\mathbf{A}}, \mathbf{E})$, where edges $\mathbf{E}$ connect real and observed attributes via push/pull queries. If there exists an edge e which connects A_c and $\tilde{A}_{c'}$, we say that A_c corresponds to $\tilde{A}_{c'}$ and denote this as $A_c \approx \tilde{A}_{c'}$. To evaluate authorization predicates, the reference monitor can exploit observed attribute values and timestamps of the corresponding real counterparts.

Example 2: Figure 2 describes the exchange of attributes between the RMS and the AUCS from our running example. The left part of the figure is devoted to access control. The attribute value A_0 sent by the RMS at $t_{try} = cl_{AP}(0)$ is observed by the AUCS at $t_{perm} = cl_{RM}(0)$ as $\tilde{A}_0$, i.e., $A_0 \approx \tilde{A}_0$.

For the right part of the figure, i.e., usage control, RMS sends the fourth change of the attribute A_4 at $cl_{AP}(4)$, which is observed by AUCS as $\tilde{A}_2$ at time $cl_{RM}(2)$, i.e., $A_4 \approx \tilde{A}_2$.

C. Intentional and Unintentional Uncertainties

Observed attributes differ from their real counterparts due to attacks, noise, delays during delivery, missed attributes, etc. We call *uncertainty* a property on real and observed attributes which specifies how these values vary. The closer observed values are to the real ones the more reliable the enforcement of the policy. In this paper, we consider two types of uncertainties: *unintentional* (*freshness* and *correctness*), and *intentional* (*trustworthiness*).

1) Freshness of Attributes: is an unintentional uncertainty occurring due to the mutability of attributes. Generally, this

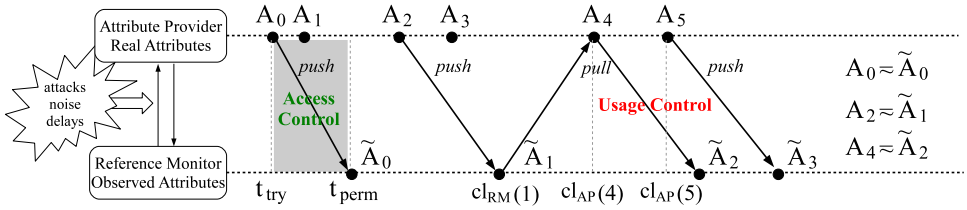

Fig. 2: Real and Observed Attribute Values

property means that the latest observed value of an attribute is out-of-date, while the current real value of the attribute is unknown. We introduce three types of freshness uncertainties.

Freshness I (non-continuous checks) corresponds to scenarios where only part of attribute changes is detected because the checks are carried out through some time interval:

$$\exists c \geq 0,\ m > 0,\ \ c, m \in \mathbb{N}\ :\ A_{c+m} \approx \tilde{A}_c$$

In Figure 2 the attribute provider sends A_2 and A_4 values, while A_3 is not sent. Thus, the reference monitor making a decision after getting A_2 value ($\tilde{A}_1$) uses the wrong input for the decision.

Example 3: After granting access to the user the AUCS has to monitor the reputation value during the usage. The RMS sends the current reputation value only once per hour in order to save resources. If the reputation of the user became "malicious" between the checks he will still use the system because the AUCS is not aware of this change.

Freshness II (delays in processing) implies that there are inevitable time delays in delivery of an attribute value (due to a network latency) and decision making (evaluation of authorization predicates). That is:

$$\exists c' \geq 0,\ c'' \geq 0,\ c', c'' \in \mathbb{N}:\ A_{c''} \approx \tilde{A}_{c'}$$
$$cl_{RM}(c') >\ cl_{AP}(c'')$$

When the attribute provider gets a request for the attribute it sends value A_0 to the reference monitor (see Figure 2 again). Since the delivery takes some time ($t_{perm} - t_{try}$) the attribute changes to A_1 and the access control system uses the wrong value for the analysis.

Example 4: A user asks AUCS for an access. AUCS asks RMS for the current reputation value of the user and gets "suspicious". The problem is that because of the delay in the delivery when AUCS makes the decision the value becomes obsolete, since a new feedback comes to RMS and the reputation value changes to "malicious".

Freshness III (pending updates) corresponds to scenarios where the current attribute value is uncertain since some update queries are pending at the time of the access re-evaluation. In this case, the attribute provider sends two values: (i) the last certain attribute value, (ii) additional information on how the real value differs from the last certain value.

The presence of the uncertainty freshness III implies:

$$\exists c' \geq 0,\ c'' \geq 0, m > 0,\ c', c'', m \in \mathbb{N}:\ A_{c''} \approx \tilde{A}_{c'}$$
$$cl_{AP}(c'' + m) \leq cl_{RM}(c')$$

In Figure 2, the reference monitor which is going to make a decision after getting value A_4 may already know, that this value is not certain. The attribute provider sending A_4 also sends additional information that there should be one more change ($m = 1$) in the attribute between $cl_{AP}(5) - cl_{AP}(4)$.

Example 5: The RMS updates the reputation only when an execution is ended and the RMS receives feedback from a resource provider. Applications run concurrently and each single execution may be long-lived and last for days. The access decision to use the resource (made by the AUCS) is based on the reputation value dated by the last registered feedback and on the number of applications currently running on the user's behalf. Indeed, the ongoing applications can be malicious but this fact will only be discovered afterwards. The only way to make the certain decision is to block the access until all running applications terminate. Instead, the AUCS has to be set up to make an access decision with some uncertainty regarding the current reputation of the user. This uncertainty is contained in the amount of jobs still active (m value).

2) Correctness: is affected by additive noises that usually exist in case of non-accurate measurements. For example, the location attribute can be sensed only with the given precision. Thus, observed attribute values differ from the real ones:

$$\exists c' \geq 0,\ c'' \geq 0\ , c', c'' \in \mathbb{N}:\ A_{c''} \approx \tilde{A}_{c'}$$
$$\tilde{A}_{c'} = A_{c''} + N$$

and N is a random variable that models additive noises presented in observed attribute values. The reference monitor may know that the attribute value measured by the attribute provider is not precise. Thus, on getting a value (e.g., A_2) the reference monitor makes the decision taking the mistake N into account. This case cannot be shown in Figure 2 directly.

Example 6: It is known that the RMS reputation values may differ from the real ones by a maximum of 1 for various reasons (e.g., some feedback could be lost). The AUCS should be aware of the possibility of such mistakes.

3) Trustworthiness: is an intentional uncertainty. It appears as a result of the attribute provider altering attributes or as a result of attacks during attribute delivery, storage, etc. Current approaches guarantee only the integrity of an attribute by validating a signature of the entity signing the attribute, but this does not guarantee trustworthiness. This uncertainty assumes that either an attribute value, or a time of issuance, or both can be modified. It implies that the reference monitor does not trust the attribute provider and assigns a confidence value for each observed attribute. This value represents the reliability of the attribute provider in the assertions it makes.

Approaches which consider trust as a probability that an interaction will succeed or fail can be used for analysis the probability for *static* attributes, i.e., the fact which can be either true or false (e.g., [29], [10]). For the computation of trustworthiness value a feedback collection mechanism is required, which is powerful enough to detect whether the received value was modified. Naturally, if such check could be performed timely for all received values there is no intentional uncertainty in the system. However, such check may require significant amount of resources and time (e.g., checking the logs of service provider) and the information about trustworthiness of a user may be collected only from time to time just to compute the reputation value. The problem of trustworthiness for *mutable* attributes, i.e., the attributes which have a wider domain of possible values, is an open issue which is to be investigated. But our method only uses the probability of policy violation and does not depend on its way of computation.

The presence of the trustworthiness uncertainty states:

$$\exists c' \geq 0,\ c'' \geq 0\,, c', c'' \in \mathbb{N}:\ A_{c''} \approx \tilde{A}_{c'}$$
$$\mathbf{Pr}[\tilde{A}_{c'} = A_{c''}] = \eta,\ 0 \leq \eta < 1$$

i.e., the probability that the observed attribute is equal to the real counterpart is below 1 and we assume that the reference monitor has the power to compute η. Similarly to Correctness this uncertainty also cannot be shown in Figure 2.

Example 7: The RMS sends to the AUCS the reputation attribute is equal to "normal". The AUCS does not trust the RMS entirely and based on its internal estimates the AUCS considers that the observed attribute has the "normal" value but with a probability of 0.8.

In the following sections we continue to use our running example taking into consideration only the uncertainties of Freshness III type for access control (Section VI) and Freshness I for usage control (Section VII).

V. Correct Policy Enforcement

The correct policy enforcement implies that having observed attributes the reference monitor enforces the policy exactly in the same fashion as with real attributes, and both observed and real attributes satisfy authorization predicates.

A. Correct Enforcement of Access Control

Access control starts at time $t_{try} = cl_{AP}(0)$ when the user sends the access request and the initial attribute value. The reference monitor evaluates a policy only once and grants an access to a resource at time $t_{perm} = cl_{RM}(0)$ if the policy holds. We say that the *policy holds for access control* if:

1) $\mathcal{P}(\tilde{A}_0)$ happens, i.e., the initial observed attribute value $\tilde{A}_0$ satisfies the policy;
2) $\mathcal{P}(A_m)$ happens, i.e., the real attribute value A_m at the time the decision is made also satisfies the policy and $cl_{AP}(m) \leq t_{perm} < cl_{AP}(m+1)$ where $m \geq 0$.

Note, that some attribute changes may happen between t_{try} and t_{perm}, but attribute values must satisfy security policy exactly when the request is issued and later when the access decision is evaluated.

Let H be an event specifying that the policy holds and $\overline{H}$ specifies the opposite. Clearly, the policy satisfaction and violation can be defined as:

$$H = \mathcal{P}(\tilde{A}_0) \cap \mathcal{P}(A_m) \qquad (2)$$
$$\overline{H} = \overline{\mathcal{P}}(\tilde{A}_0) \cup (\mathcal{P}(\tilde{A}_0) \cap \overline{\mathcal{P}}(A_m))$$

Definition 1: ***(Correct Enforcement of Access Control)*** The reference monitor grants the access at t_{perm} if the policy holds and denies it otherwise.

Let G be an event specifying that the reference monitor grants the access and $\overline{G}$ specifies the opposite (i.e., denies the access). Thus, the correct enforcement of access control is

$$G = H,\ \ \overline{G} = \overline{H} \qquad (3)$$

B. Correct Enforcement of Usage Control

We say that a *policy holds for usage control* on a time interval $[t_b : t_e]$ if:

1) $\mathcal{P}(\tilde{A}_k) \cap \mathcal{P}(\tilde{A}_{k+1}) \cap ... \cap \mathcal{P}(\tilde{A}_l)$ happens and $cl_{RM}(k) \leq t_b < cl_{RM}(k+1), cl_{RM}(l) \leq t_e < cl_{RM}(l+1)$;
2) $\mathcal{P}(A_i) \cap \mathcal{P}(A_{i+1}) \cap ... \cap \mathcal{P}(A_j)$ happens and $cl_{AP}(i) \leq t_b < cl_{AP}(i+1), cl_{AP}(j) \leq t_e < cl_{AP}(j+1)$,

i.e., all real and observed attribute changes occurring within this interval do satisfy authorization predicates.

If there is at least one attribute value (either real or observed) which does not satisfy authorization predicates, we call this a *policy violation of usage control*.

Definition 2: ***(Correct Enforcement of Usage Control)*** The reference monitor correctly continues the usage session at t_{now} if a policy holds on interval $[t_{perm} : t_{now}]$. The reference monitor revokes the access immediately when the policy violation occurs.

VI. Enforcement of Access Control under Uncertainties

Correct enforcement is not feasible in the presence of uncertainties since the reference monitor is unable to show that real attribute values satisfy a policy. The basic idea of the policy enforcement of *access control* under uncertainties is:

1) The reference monitor evaluates the policy with respect to observed attribute values.
2) If the observed values satisfy the policy, the reference monitor runs an experiment which estimates to what extent the observed attributes vary from the real ones. If this difference is negligible, the experiment succeeds and the reference monitor allows the access.

A. Models for Access Control Enforcement

We suppose that the reference monitor is powerful to get some probabilistic knowledge about a real attribute value based on the observed attribute $\tilde{A}_0 = a$:

$$\mathbf{Pr}_{RM} = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = a]$$

specifies a conditional probability that a value of real attribute A_m satisfies authorization predicates at time t_{perm} if the

observed attribute value at time t_{perm} is equal to a. The reference monitor computes $\mathbf{Pr}_{RM}$ using the following data:

1) observed values of the attribute;
2) parameters of a stochastic process that models a real behaviour of an attribute;
3) a list of uncertainties presented in the system.

Possible combinations of the last two factors produce a variety of techniques on *how* to compute $\mathbf{Pr}_{RM}$. As an example, we refer the reader to [16], [15] where the behaviour of an attribute is modelled as a Markov chain and freshness uncertainties exist in the system. Another example given in [5] studies a static attribute (i.e. the attribute does not change its value over time) in the presence of the trustworthiness uncertainty. In our running example for access control we compute $\mathbf{Pr}_{RM}$ considering only Freshness III uncertainty and model the attribute behaviour as a discrete-time Markov chain.

Let Y be a random variable such that

$$Y = \begin{cases} 1 & \text{if uncertainties are acceptable} \\ 0 & \text{otherwise} \end{cases}$$

Let $\delta(x)$ be a function, that is

$$\delta(x) = \begin{cases} 1 & \text{if } x \geq th \\ 0 & \text{otherwise} \end{cases}$$

where th is a real-value threshold.

We propose two models of enforcement for access control under uncertainties: a *threshold* enforcement and a *flip coin* enforcement. The reference monitor chooses one of these models.

Definition 3: ***(Threshold Enforcement of Access Control)*** The reference monitor computes $\mathbf{Pr}_{RM}$ and grants access at t_{perm} if:

1) $\mathcal{P}(\tilde{A}_0)$ happens;
2) $Y = 1$, where $\mathbf{Pr}[Y = 1] = \delta(\mathbf{Pr}_{RM})$,

otherwise, the access is denied.

That is, if the initial observed attribute value satisfies authorization predicates, the reference monitor grants the access if the probability that the real attribute value A_m also satisfies authorization predicates is above a specified threshold th.

Definition 4: ***(Flip Coin Enforcement of Access Control)*** The reference monitor behaves exactly as in the threshold enforcement but uses $\mathbf{Pr}[Y = 1] = \mathbf{Pr}_{RM}$ instead.

Hence, if the initial observed attribute value satisfies authorization predicates, the reference monitor runs the random experiment that succeeds (returns grant) with probability $\mathbf{Pr}_{RM}$ and fails (returns deny) with probability $1 - \mathbf{Pr}_{RM}$.

In notation of events, we get for the enforcement of access control under uncertainties (either threshold or flip coin):

$$\begin{aligned} G &= \mathcal{P}(\tilde{A}_0) \cap [Y = 1] \\ \overline{G} &= \overline{\mathcal{P}}(\tilde{A}_0) \cup (\mathcal{P}(\tilde{A}_0) \cap [Y = 0]) \end{aligned} \tag{4}$$

Example 8: Consider the access control part of our running example (see Figure 2). The AUCS gets value $\tilde{A}_0 = 3$ ("suspicious") at time t_{perm} and it knows that there was one attribute change between t_{try} and t_{perm}. Now the AUCS should evaluate whether the current reputation value is still a good one, e.g., $\mathbf{Pr}_{RM} = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = 3]$.

The transition matrix (see Example 1) shows that if the initial attribute value is $A_0 = 3$, then there are three possibilities for the value to evolve in one step: (i) $(A_1 = 4)$ with $\mathbf{Pr}_{34} = 0.5$; (ii) $(A_1 = 3)$ with $\mathbf{Pr}_{33} = 0.3$; (iii) $(A_1 = 2)$ with $\mathbf{Pr}_{32} = 0.2$. Since, the good states are 1, 2, and 3 then $\mathbf{Pr}_{RM} = 0.3 + 0.2 = 0.5$.

B. Cost Matrix

We would now like to estimate the cost-effectiveness of the proposed enforcement methods. Our goal is to find the *expected profit* $\langle C \rangle$ for enforcement of access control.

We assign monetary outcomes for granting and revoking access. Correct enforcement is impossible in the presence of uncertainties and mistakes in the decisions made by the reference monitor are unavoidable. We have four scenarios (events) of how the reference monitor acts under uncertainties:

- $G \cap H$ *true positive*: grant access when policy holds;
- $G \cap \overline{H}$ *false negative*: grant access when policy is violated;
- $\overline{G} \cap H$ *false positive*: deny access when policy holds;
- $\overline{G} \cap \overline{H}$ *true negative*: deny access when policy is violated.

True positive and *true negative* are well-chosen scenarios, while *false negative* and *false positive* are erroneous.

Each scenario has a monetary outcome, i.e. *cost*, the reference monitor loses/gains if a scenario happens. Let C_{tp} denote the cost of the true positive scenario, when the reference monitor grants the access and the policy really holds. C_{fn}, C_{fp}, C_{tn} are the costs of the remaining scenarios, respectively. The semantics of costs for access control corresponds to "pay-per-access" attributes, and specifies exact benefits and losses for a given access request. Naturally, well-chosen scenarios have positive values, i.e. $C_{tp} \geq 0, C_{tn} \geq 0$, while the erroneous ones have negative costs, i.e., $C_{fp} < 0, C_{fn} < 0$. Finally, let C_a be the cost to push/pull (observe) an attribute value.

Finding correct costs is not an easy task and usually requires a considerable amount of statistical data. Thus, we make the usual assumption for risk-based methods that the reference monitor has enough historical data to compute costs.

C. Cost of Access Control Enforcement

The expected profit received by the reference monitor processing a single access request is the sum of the costs of all 4 scenarios weighted on corresponding probabilities.

$$\begin{aligned} \langle C \rangle = C_{tp} \cdot \mathbf{Pr}[G \cap H] + C_{fn} \cdot \mathbf{Pr}[G \cap \overline{H}] \\ + C_{fp} \cdot \mathbf{Pr}[\overline{G} \cap H] + C_{tn} \cdot \mathbf{Pr}[\overline{G} \cap \overline{H}] + C_a \end{aligned} \tag{5}$$

1) Correct Enforcement: Since H and $\overline{H}$ are disjoint events, i.e. $\mathbf{Pr}[H \cap \overline{H}] = 0$ and $\mathbf{Pr}[H] + \mathbf{Pr}[\overline{H}] = 1$, from Equations 2, 3 and 5 we receive

$$\langle C \rangle_{cor} = C_{tp} \cdot \mathbf{Pr}[H] + C_{tn} \cdot \mathbf{Pr}[\overline{H}] + C_a \tag{6}$$

$$\begin{aligned} \mathbf{Pr}[H] = \mathbf{Pr}[\mathcal{P}(\tilde{A}_0) \cap \mathcal{P}(A_m)] = \\ \mathbf{Pr}[\mathcal{P}(\tilde{A}_0)] \cdot \mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)] \end{aligned} \tag{7}$$

In what follows, we use $\mathbf{Pr}[\mathcal{P}(\tilde{A}_0)]$ interchangeably with α, and $\mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)]$ with β. Note, that for the correct access control $\beta = 1$. Finally,

$$\langle C \rangle_{cor} = C_{tp} \cdot \alpha \cdot \beta + C_{tn} \cdot (1 - \alpha \cdot \beta) + C_a \tag{8}$$

2) Threshold Enforcement: We point out that the probability of a policy satisfaction for real attributes is conditionally independent of the estimates made by the reference monitor given that observed attribute values satisfy the policy. Using this observation and Equations 2 and 4 we receive

$$\begin{aligned}
&\mathbf{Pr}[G \cap H] = \alpha \cdot \beta \cdot \mathbf{Pr}[Y = 1|\mathcal{P}(\tilde{A}_0)] \\
&\mathbf{Pr}[G \cap \overline{H}] = \alpha \cdot (1 - \beta) \cdot \mathbf{Pr}[Y = 1|\mathcal{P}(\tilde{A}_0)] \\
&\mathbf{Pr}[\overline{G} \cap H] = \alpha \cdot \beta \cdot (1 - \mathbf{Pr}[Y = 1|\mathcal{P}(\tilde{A}_0)]) \\
&\mathbf{Pr}[\overline{G} \cap \overline{H}] = \alpha \cdot (1 - \beta) \cdot (1 - \mathbf{Pr}[Y = 1|\mathcal{P}(\tilde{A}_0)])
\end{aligned} \tag{9}$$

We assume that all access requests come with the same initial attribute value a which satisfies authorization predicates. Such a situation is modelled with an assumption $\alpha = 1$. With this assumption, we get that $\mathbf{Pr}[Y = 1|\mathcal{P}(\tilde{A}_0)] = \mathbf{Pr}[Y = 1|\tilde{A}_0 = a]$ and $\beta = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = a] = \mathbf{Pr}_{RM}$.

We denote $C_g = \beta \cdot (C_{tp} - C_{fn}) + C_{fn}$ and $C_d = \beta \cdot C_{fp} + C_{tn} \cdot (1 - \beta)$. From Definition 3 and Equations 9 and 5 we get the average profit for a threshold enforcement:

$$\langle C \rangle_{th} = C_a + \begin{cases} C_g & \text{if } \beta \geq th \\ C_d & \text{otherwise} \end{cases} \tag{10}$$

Cost-effective enforcement implies that we should pick a threshold which gives the maximal profit for all possible average costs. Since the cost is a function of β which takes any value from 0 to 1, we should maximize the sum of costs for all β. The argument, for which this sum attains its maximum, constitutes the *optimal threshold value*:

$$\arg\max_{th} \int_0^1 \langle C \rangle_{th} \, d\beta$$

To obtain it, we solve the equation in which the derivative of the integral takes zero:

$$\left(\int_0^{th} C_d \, d\beta + \int_{th}^1 C_g \, d\beta \right)'_{th} = 0$$

Hence, the optimal threshold value is given by

$$th = \frac{C_{fn} - C_{tn}}{C_{fp} + C_{fn} - C_{tn} - C_{tp}} \tag{11}$$

3) Flip Coin Enforcement: All equations of a threshold enforcement are also valid for a flip coin enforcement. Taking the assumptions made in the threshold enforcement and Definition 4, we obtain the average profit for a flip-coin enforcement per access request:

$$\begin{aligned}
\langle C \rangle_{flip} = C_{tp} \cdot \beta^2 + (C_{fp} + C_{fn}) \cdot \beta \cdot (1 - \beta) + \\
C_{tn} \cdot (1 - \beta)^2 + C_a
\end{aligned} \tag{12}$$

Proposition 1: Threshold strategy is more cost-effective than flip-coin, except the points $\beta = 0$, $\beta = 1$, and $\beta = th$, where the strategies are equal: $\langle C \rangle_{th} \geq \langle C \rangle_{flip}$.

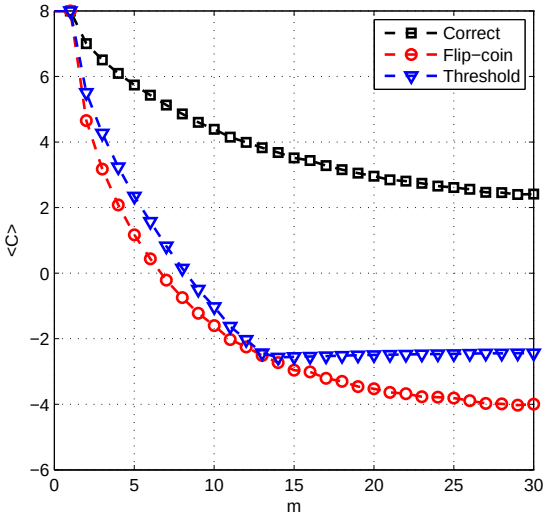

Fig. 3: Cost-Effective Enforcement of Access Control

Proof: Consider the first case when $1 > \mathbf{Pr}_{RM} = \beta \geq th$. We do not consider the case when $\mathbf{Pr}_{RM} = \beta = 1$ since it is easy to see that the two strategies are equal at this point. Now let us derive the conditions where the flip-coin strategy is better than the threshold one:

$$\begin{aligned}
&C_{tp} \cdot \beta^2 + (C_{fp} + C_{fn}) \cdot \beta \cdot (1 - \beta) + \\
&C_{tn} \cdot (1 - \beta)^2 + C_a > \beta \cdot (C_{tp} - C_{fn}) + C_{fn} + C_a
\end{aligned}$$

After algebraic transformations that gives:

$$\beta < \frac{C_{fn} - C_{tn}}{C_{fp} + C_{fn} - C_{tn} - C_{tp}} = th$$

We see that the condition for the flip-coin strategy is better than that of the threshold violates the initial preposition $\beta > th$. Thus, if $1 > \beta > th$ the threshold strategy is more profitable.

In the same way we can compare the strategies with the conditions $th > \beta > 0$. We exclude $\beta = 0$ point where the strategies are equal. In this case we get that $\beta > th$, that proving once again that the threshold strategy is better, except the points where the strategies are equal. ■

Example 9: We show how different strategies cope with Freshness III uncertainty in our running example.

The AUCS gets the attribute value $\tilde{A}_0 = 2$ at t_{perm} and can compute that there were exactly m attribute changes between t_{try} and t_{perm}. The AUCS must then compute the probability β that the policy holds at t_{perm} and choose the model of the policy enforcement. The probability matrix of the Markov chain was given in Example 1 and the probability β can be found as (see also [16], [15], [13] and Appendix A1 and A2):

$$\beta = \mathbf{Pr}[\mathcal{P}(A_m)|\tilde{A}_0 = 2] = \sum_{j \in \{1,2,3\}} (\mathrm{S} \cdot \mathbf{Prob}^m)[j]$$

the vector $\mathrm{S} = [0;\ 1;\ 0;\ 0]$ specifies the initial attribute value.

The AUCS makes monetary estimations and determines the following costs: $C_{tp} = 10$, $C_{fn} = -15$, $C_{fp} = -1$, $C_{tn} = 0$ and to query an attribute we pay $C_a = -2$.

We performed a set of simulations in order to illustrate our theory. We computed the average profit per access request for the *correct enforcement* $\langle C \rangle_{cor}$, for the *threshold enforcement* $\langle C \rangle_{th}$, and for the *flip-coin* enforcement $\langle C \rangle_{flip}$. We varied the uncertainties between real and observed attributes by increasing the number m of attribute changes that occur between t_{try} and t_{perm}. We start from $m = 0$ and go up to 30 unobserved attribute changes.

Figure 3 shows the results obtained. The average profit per access request for the *correct enforcement* is always higher. The decline of the correct curve occurs because while the delay increases the probability that the received value would fail the policy also increases (because of $\mathbf{Pr}[\mathcal{P}(A_m)|\mathcal{P}(\tilde{A}_0)] = \beta$). Since the attribute cannot get a bad value in $m = 0$ or $m = 1$ steps (starting from state 2) all three curves have the same maximal value in these cases. The flip coin enforcement shows the worse results with respect to the threshold enforcement which tallies with our theoretical findings.

VII. Enforcement of Usage Control under Uncertainties

Our model of usage control enforcement under uncertainties imposes that the reference monitor iteratively performs three main activities.

1) Evaluates a policy and makes the decision based on the observed attribute values. If the access decision is "deny", the reference monitor terminates the usage session and halts.
2) Computes when the next attribute query should be performed.
3) Waits until the next check and when time elapses pulls a fresh attribute value.

The reference monitor executes these actions on each *check*. A check is a time interval $[t_b : t_e]$ between two adjacent observations of the attribute $\tilde{A}_{k-1}$ and $\tilde{A}_k$, where $cl_{RM}(k-1) = t_b$, $cl_{RM}(k) = t_e$. The time of the first check is t_{perm} when there is the observed attribute $\tilde{A}_0$. The *usage session* contains a sequence of n checks and $n \in \mathbb{N}$.

A. Models for Usage Control Enforcement

1) Decision Making: The basic idea of a decision making for usage control under uncertainties is the same as for access control (see Section VI). The only difference is that the reference monitor should take into account all possible changes occurred on a check. We assume that the reference monitor has the power to compute the probability that all real attributes satisfy a policy on the k-th check

$$\mathbf{Pr}^k_{RM} = \mathbf{Pr}[\mathcal{P}(A_i) \cap ... \cap \mathcal{P}(A_j)|\tilde{A}_{k-1} = a_{k-1} \cap \tilde{A}_k = a_k]$$

where $cl_{AP}(i) \leq cl_{RM}(k-1) \leq cl_{AP}(i+1)$, $cl_{AP}(j-1) \leq cl_{RM}(k) \leq cl_{AP}(j)$.

We propose two models of a decision making for usage control under uncertainties: a threshold and a flip coin.

*Definition 5: **(Usage Control Based on Threshold)*** The reference monitor continues the access after n policy checks at $t_{now} = cl_{RM}(\tilde{A}_n)$ if:

1) $\mathcal{P}(\tilde{A}_0) \cap \mathcal{P}(\tilde{A}_1) \cap ... \cap \mathcal{P}(\tilde{A}_n)$ occurs, i.e., all attribute changes observed within n checks do satisfy the policy,
2) $\forall k = 1, .., n : Y_k = 1$, where $\mathbf{Pr}[Y_k = 1] = \delta(\mathbf{Pr}^k_{RM})$, i.e., for each check the probability that a policy holds on this check should be above a specified threshold,

otherwise access is revoked.

*Definition 6: **(Usage Control Based on Coin Flip)*** The reference monitor behaves as in the threshold enforcement but uses $\mathbf{Pr}[Y_k = 1] = \mathbf{Pr}^k_{RM}$.

2) Attribute Retrieval: Fresh attribute values could be pushed or pulled. Without loss of generality we assume that the reference monitor is responsible for pulling attribute values. Since frequent attribute queries are not always possible, expensive and lead to a performance slowdown, we assume that several attribute changes may occur on a single check. Such scenario brings the inevitable Freshness I uncertainty since the reference monitor will observe only a part of attribute changes. The reference monitor should be aware that unnoticed attribute changes may violate a policy and result in a loss.

Our main concern is to find such intervals between queries that give the maximal profit for the enforcement of a usage session. We propose two models of attributes retrieval. The first one is *periodic pull of attributes* when the interval between attribute quires is constant. The second model is *aperiodic pull of attributes*. We assume that the reference monitor may increase the profit if it selects the interval between quires according to the history of observed attributes during the current session. Thus, there is a specific value of interval for each specific check.

B. Costs of Usage Control Enforcement

Possible combinations of decision making and attribute retrieval launch a variety of enforcement models. Due to space limitations, we discuss only the models relevant for usage control and do not consider models discussed previously for access control. We examine the cost-effectiveness of models when attributes are pulled periodically and aperiodically while the decision making is based on a threshold. In both models, we set the threshold value to 0 and assume that no uncertainties exist in the system except inevitable Freshness I. Such assumptions allows the reference monitor to skip the execution of the random experiment and just continue access if the observed attribute value satisfies a policy and revoke otherwise.

1) Cost of Usage Session in case of Periodic Checks: We start with a cost gained from the enforcement of a particular usage session. The semantics of costs for usage control corresponds to "pay-per-time-of-usage" attributes, and specifies the benefits and losses the system gains in a unit of time. The system receives profit if a policy holds on a time interval and this revenue is proportional to the duration of the interval. In opposite, the system suffers losses during the policy violation time. There are three costs for usage control: (i) c_{tp} - the gain per atomic interval of time when all changes of real attributes satisfy the policy; (ii) c_{fn} - the cost per atomic interval of time when the policy fails; and (iii) C_a - the cost paid for the attribute retrieval and the re-evaluation of access decision.

The usage session is associated with a sample sequence s of a stochastic process which models the behaviour of a real attribute. That is:

$$s : (A_0 = a_0) \cap (A_1 = a_1) \cap ... \cap (A_l = a_l)$$

Let n state a total number of checks in the session before revocation. This means that after the last check the reference monitor revokes the session, i.e., $\overline{\mathcal{P}}(\tilde{A}_n)$ happens and $A_l \approx \tilde{A}_n$. Let q be a number of attribute changes on a check. Since checks are periodic, q is a constant for any check and $l = n \cdot q$. A cost C_s of a particular usage session depends on the time τ_g when an attribute satisfies a policy, on the time τ_b when the attribute violates the policy, and a number of checks n:

$$C_s = c_{tp} \cdot \tau_g + c_{fn} \cdot \tau_b + C_a \cdot (n + 1) \tag{13}$$

Let $\theta(x)$ be a function such that

$$\theta(x) = \begin{cases} 1 & \text{if } \mathcal{P}(A_x) \text{ happens} \\ 0 & \text{otherwise, i.e., a policy violation happens} \end{cases}$$

Then, τ_g and τ_b are given by

$$\tau_g = \sum_{j=0}^{l-1} (cl_{AP}(j+1) - cl_{AP}(j)) \cdot \theta(j)$$
$$\tau_b = cl_{RM}(n) - cl_{RM}(0) - \tau_g$$

In fact, s is a random event and let $\mathbf{Pr}[s]$ denote a probability that s occurs. Thus, the *average* cost of usage control enforcement will be a sum over every possible cost weighted by the probability of s:

$$\langle C \rangle_q = \sum_{s \in S} \mathbf{Pr}[s] \cdot C_s \tag{14}$$

where S contains all possible sample sequences associated with usage sessions enforced under uncertainties.

Cost-effective enforcement implies that the reference monitor should choose such q that maximizes profit: $\arg\max_q \langle C \rangle_q$.

2) Cost of Usage Session in case of Aperiodic Checks: In case of aperiodic checks, a number of attribute changes occurred on each check is different. There is a set $Q = \{q_1, q_2, ..., q_n\}$ and each q_i tells how many attribute changes happened on the i-th check. All formulas given for periodic checks are valid for aperiodic. Only a number of attribute changes is different, and for aperiodic checks we have that $l = \sum_{q \in Q} q$. We also use $\langle C \rangle_Q$ to denote the average cost of the usage control enforcement under aperiodic checks.

Cost-effective enforcement implies that the reference monitor should choose such Q that gives the maximal profit, i.e., $\arg\max_Q \langle C \rangle_Q$. The simplex method can be used to find Q for which $\langle C \rangle_Q$ attains the maximum. The application of such methods is left behind the scope of this paper but initial ideas can be found in [2], [27].

Proposition 2: Aperiodic checks are at least as good as periodic checks in terms of cost-effectiveness: $\langle C \rangle_Q \geq \langle C \rangle_q$.

Proof: The proof follows from the fact, that the method selects the set Q with the best average cost within all possible Q's. Note, periodic checks may be considered as a particular case of aperiodic checks when all intervals are equal. ■

Example 10: We continue our running example comparing periodic and aperiodic checks.

The AUCS selects the following costs $c_{tp} = 3$, $c_{fn} = -5$, and $C_a = -2$ on the basis of previous behaviour of the reputation attribute. The AUCS exploits discrete-time Markov chain (Equation 1) to model the behaviour of the reputation and find the best strategy for querying this attribute.

For the periodic checks, the probability $\mathbf{Pr}[s]$ is:

$$\mathbf{Pr}[s] = \mathbf{Pr}^*_{j_0} \cdot (\prod_{y=1}^{n-1} \mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)) \cdot \mathbf{Pr}^{k_n}_{y_{n-1} y_n}(q)$$

Where $\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)$ is a probability of the reputation change from the value j_y to the value j_{y+1} taking the set of values k_y on the interval between changes, $\mathbf{Pr}^*_{j_0}$ is a probability that the attribute will have the certain good value at the first check.

$$\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q) = \prod_{z=1}^{q-1} \mathbf{Pr}_{f_z f_{z+1}}$$

$\mathbf{Pr}_{f_z f_{z+1}}$ is an element of the matrix **Prob** of one-time transition probabilities (Equation 1). Clearly $f_1 = j_y$, $f_q = j_{y+1}$, and k_y determines concrete values of $\{f_2, \ldots, f_{q-1}\}$. There are m^{q-2} possible $\mathbf{Pr}^{k_y}_{j_y j_{y+1}}(q)$ if j_y and j_{y+1} are fixed.

For aperiodic checks the computations are similar to ones above. However, since the reputation is modelled as a Markov chain, the probabilistic behaviour of the reputation significantly depends on the current state of the random process. Thus, q now depends on the current value of the reputation and the AUCS selects a specific interval q_i on the basis of the last observed value $\tilde{A}_{i-1} = a_{i-1}$. Markov process quickly converges to a steady state. Therefore, the AUCS considers $q_i < q_{max}$, where q_{max} is the number of changes when the distribution of probabilities differs from the steady state distribution by some small value ϵ. For a more detailed description see Appendix B.

We performed several simulations to check the values provided by our theoretical equation. To evaluate the aperiodic checks we carried out an exhaustive search of the optimal lengths of intervals between checks and found the values $q_1 = 7$, $q_2 = 4$, and $q_3 = 1$ if the current observed value is "general", "normal", and "suspicious" respectively. The computations of q_i are only required ones the policy is deployed in the system.

The results of the simulations are shown in Figure 4. Since there is no single interval for aperiodic checks, we display aperiodic checks as a straight line.

First, both periodic and aperiodic checks are close enough to the theoretical curves. Second, the simulations illustrate our proposition regarding the fact that aperiodic checks are at least as cost-effective as periodic ones. In our example, aperiodic checks are about 15% more cost-effective then periodic checks. Third, the analysis of the periodic checks shows that the average cost of the session has the maximum value when the interval between checks is 4. The smaller interval is ineffective because we pay more for requesting an attribute. The bigger intervals are ineffective, because the system misses more policy violations.

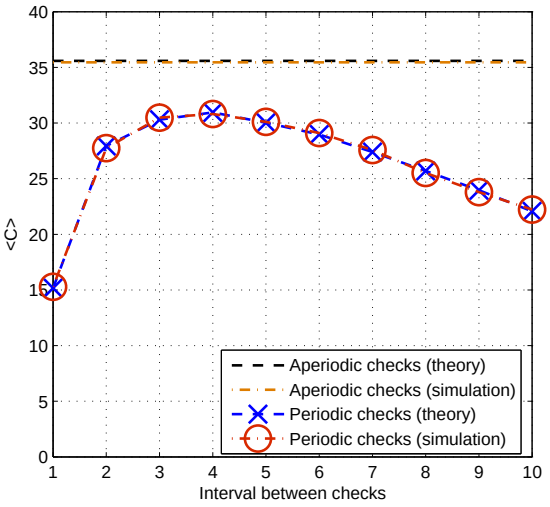

Fig. 4: Cost-Effective Enforcement of Usage Control

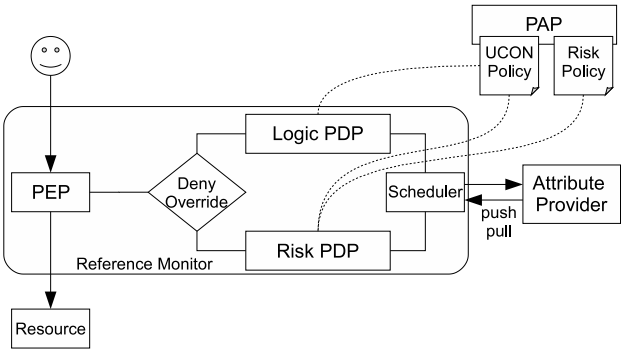

Fig. 5: Architecture of Reference Monitor

VIII. Architecture of Policy Enforcement under Uncertainties

The architecture of the reference monitor should be tuned to capture the presence of uncertainties. Figure 5 shows the overall architecture consisting of the following components:

- Policy Enforcement Point (PEP) is a component which intercepts invocations of security-relevant access requests, suspends them before starting, queries the PDP for access decisions, enforces obtained decisions by resuming suspended requests, and interrupts ongoing accesses when the policy violation occurs;
- Policy Decision Point (PDP) is a component which evaluates security policies and produces the access decision;
- Attribute Provider (AP) is a component which manages attributes and knows their real values;
- Policy Administrative Point (PAP) is a component which provides and governs security policies.

Note that in the settings of a distributed environment each component can run on a different host.

The main novelty of the policy enforcement under uncertainties is that the PDP also consists of several components: the *logic PDP*, the *risk PDP*, and the *scheduler*.

The logic PDP behaves as a usual PDP [19] and evaluates logical predicates over observed attributes. The risk PDP computes all uncertainties associated with observed attributes and runs the random experiment to get a value of a random variable Y. If $Y = 1$, the risk PDP outputs "grant" and "deny" otherwise. Decisions of both PDPs are combined as "deny-override", i.e. the PDP sends "grant" to the PEP only if both the logical and risk PDPs grant the access.

Policies used by the logical PDP can be written in any appropriate language to formalize the UCON model, e.g. a POLPA language [4]. The risk PDP additionally uses risk policies, i.e. a cost matrix, specifications of stochastic processes which model the behaviour of attributes, and values of thresholds. In fact, security and risk policies can be provided by different parties (security administrators).

The scheduler is managed by the risk PDP and is responsible to collect and process attribute observations. When a new attribute value is pushed to the reference monitor, the scheduler transforms it into the proper format and triggers both PDPs to re-evaluate the access decision. During usage control, the scheduler usually pulls new attributes from the AP and then again processes them and forwards these observations to the PDPs. The risk PDP is responsible for informing the scheduler about how and when attribute queries should be initiated: either periodically or aperiodically.

IX. Related Work

This paper is an extended and revised version of our previous works. An initial description of uncertainties impacting access and usage control is given in [16], [15]. In addition, the papers describe algorithms for the computation of probabilities on the basis of discrete [16] and [15] continuous-time Markov chains. This material is briefly presented in the Appendix. Our work [17] focuses on the cost-effective enforcement of access control under uncertainties. The current paper extends the part on access control and adds the part on the cost-effective enforcement of usage control. Moreover, we provide an architecture of reference monitor for the enforcement of access and usage control policies.

Data freshness is an important property of many computer systems (e.g., data caching, replication systems, data warehousing, etc). The property was widely studied by the computer science community during past years [6], [26]. Recently, the importance of authorisation information to be up to date during the access decisions was stated by Krishnan et al. [18] and Niu et al. [24]. Authors formally define two security properties: *weak stale safety* and *strong stale safety*. Authors design enforcement and decision points for group-based Secure Information Sharing (g-SIS) system as State Machines and use model checking to show that the points satisfy defined properties. Instead, we empower the decision making procedure with a probabilistic model, which takes into account the possibility of unnoticed change of attribute. Also, we show, that even if unnoticed changes occur, a system owner can still obtain profit from the exploitation of the system. Finally, we extend our approach for a more complicated case of decision making in usage control.

Cost-effectiveness of access and usage control is frequently analysed on the bases of a risk notion. Some authors use risk

as a static parameter which simply helps to assign correct privileges taking into account possible losses [20], [12], [29]. For example, Skalka et al. [29] discuss an approach for risk evaluation of authorisations. The formal approach is used to assess and combine the risks of assertions that are used in the authorisation decision. Other authors use risk as a dynamically changing value which depends on the current value of possible losses and benefits as well as on the probability of abusing granting privileges by a concrete subject [9], [21], [7]. Deip et al. [9] propose to compute the risk of granting the access to the resource and to grant the access if the risk is less than a threshold. Ni et al. [23] consider a risk-based access control system which assumes that the access to a resource can be granted to a risky subject if mitigation actions (post-obligations) will be applied in the future.

Several authors paid more attention to incorporating risk semantics in access policies rather than to the computation of risk. For example, the policy language, proposed by Aziz et al. [3], contains three types of risks: operational, combinatorial, and conflict of interest. Dimmock et al. [11] show how OASIS access control system and its role-based policy language can be extended with trust and risk analysis.

X. Conclusions and Future Work

In this work we investigated how access and usage control could work in presence of uncertainties. We have identified several types of uncertainties which can affect the access decision made by the reference monitor and defined threshold and flip-coin policy enforcement models which are able to make a decision under uncertainties if the required probabilities are available. We have shown that the threshold strategy is more profitable and showed how to select the threshold to maximise the profit.

Another important contribution of this work is that we have discussed periodic and aperiodic models for usage control. The simulation results conform the theory: aperiodic checks are more cost-effective than the periodic ones. On the other hand, periodic checks is a simpler model and the complexity of the aperiodic model may outweigh the benefits.

In our future work we would like to consider computation of probabilities of policy failure under intentional uncertainty when mutable attributes are considered. Current state of the art can be applied only to static attributes and we would like to extend the applicability of our theory for more general cases.

References

[1] M. Abadi. Logic in access control. In *LICS '03: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, page 228, Washington, DC, USA, 2003. IEEE Computer Society.

[2] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Publishing, 2003.

[3] A. B. Aziz, A. S. Foley, A. J. Herbert, and A. G. Swart. Reconfiguring role based access control policies using risk semantics. *Journal of High Speed Networks*, 15(3):261–273, 2006.

[4] F. Baiardi, F. Martinelli, P. Mori, and A. Vaccarelli. Improving grid service security with fine grain policies. In *Proceedings of On the Move to Meaningful Internet System 2004*. Springer, 2004.

[5] S. Bistarelli, F. Martinelli, and F. Santini. A semantic foundation for trust management languages with weights: An application to the rt family. In *Proceedings of the 5th international conference on Autonomic and Trusted Computing*, ATC '08, pages 481–495. Springer, 2008.

[6] M. Bouzeghoub and V. Peralta. A framework for analysis of data freshness. In *Proceedings of the 2004 International Workshop on Information Quality in Information Systems*, pages 59–67. ACM, 2004.

[7] P.-C. Cheng, P. Rohatgi, C. Keser, P. A. Karger, G. M. Wagner, and A. S. Reninger. Fuzzy multi-level security: An experiment on quantified risk-adaptive access control. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 222–230. IEEE Computer Society, 2007.

[8] M. L. Damiani, E. Bertino, and C. Silvestri. Approach to supporting continuity of usage in location-based access control. In *Proceedings of the 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*. IEEE Computer Society, 2008.

[9] N. N. Diep, L. X. Hung, Y. Zhung, S. Lee, Y.-K. Lee, and H. Lee. Enforcing access control using risk assessment. In *Proceedings of the Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, pages 419–424. IEEE Computer Society, 2007.

[10] N. Dimmock. How much is"enough"? risk in trust-based access control. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 281–282. IEEE Computer Society, 2003.

[11] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*, pages 156–162, New York, NY, USA, 2004. ACM.

[12] Y. Han, Y. Hori, and K. Sakurai. Security policy pre-evaluation towards risk analysis. In *Proceedings of the 2008 International Conference on Information Security and Assurance*, pages 415–420. IEEE Computer Society, 2008.

[13] O. C. Ibe. *Fundamentals of Applied Probability and Random Processes*. Elsevier Academic Press, 2005.

[14] O. C. Ibe. *Markov processes for stochastic modeling*. Elsevier Academic Press, 2009.

[15] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Influence of attribute freshness on decision making in usage control. In *Proceedings of the 6th International Workshop on Security and Trust Management*, 2010.

[16] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Risk-aware usage decision making in highly dynamic systems. In *Proceedings of The Fifth International Conference on Internet Monitoring and Protection (ICIMP'10)*, 2010.

[17] L. Krautsevich, A. Lazouski, F. Martinelli, and A. Yautsiukhin. Cost-effective enforcement of ucona policies. In *Proceedings of the 6th International Conference on Risks and Security of Internet and Systems*, 2011.

[18] R. Krishnan, J. Niu, R. Sandhu, and W. H. Winsborough. Stale-safe security properties for group-based secure information sharing. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering*. ACM, 2008.

[19] A. Lazouski, F. Martinelli, and P. Mori. Usage control in computer security: A survey. *Computer Science Review*, 4(2):81–99, 2010.

[20] Y. Li, H. Sun, Z. Chen, J. Ren, and H. Luo. Using trust and risk in access control for grid environment. In *Proceedings of the 2008 International Conference on Security Technology*. IEEE Computer Society, 2008.

[21] R. W. McGraw. Risk-adaptable access control (radac). available via http://csrc.nist.gov/news_events/privilege-management-workshop/radac-Paper0001.pdf on 16/08/09.

[22] M. Mitzenmacher and E. Upfal. *Probability and Computing Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[23] Q. Ni, E. Bertino, and J. Lobo. Risk-based access control systems built on fuzzy inferences. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010.

[24] J. Niu, R. Krishnan, J. F. Bennat, R. Sandhu, and W. H. Winsborough. Enforceable and verifable stale-safe security properties in distributed systems. Technical Report CS-TR-2011-02, University of Texas at San Antonio, 2011.

[25] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, 3(2):85–106, 2000.

[26] V. Peralta. Data freshness and data accuracy: A state of the art. Technical report, Universidad de la Repblica Uruguay, 2006.

[27] R. L. Rardin. *Optimization in operations research*. Prentice Hall, 1997.

[28] F. B. Schneider, K. Walsh, and E. G. Sirer. Nexus authorization logic (nal): Design rationale and applications. Technical report, Cornell Computing and Information Science Technical Report, 2009.

[29] C. Skalka, X. S. Wang, and P. Chapin. Risk management for distributed authorization. *J. Comput. Secur.*, 15(4):447–489, 2007.

[30] H. C. Tijms. *A First Course in Stochastic Models*. Wiley, 2003.

[31] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005.

APPENDIX

In the appendix we present a brief solutions of several computational problems related to Markov chains theory.

A. Computation of Transition Probabilities

We discuss in details a method for computation of transition probabilities based on discrete-time (DTMC) and continuous-time (CTMC) Markov chains. We define the following variables: $r \in \Omega_{attr}$ is a value of an attribute. By x_i we denote the value of the attribute in the state i; t_0 is the time (step) when we know the exact value of the attribute; t' is the time (step) when we make an access decision about the usage session.

1) Discrete-time Markov Model: First, we consider a random process represented as a DTMC. Our goal is to compute the probability $\mathbf{Pr}_{ij}(q)$ of the process to be in the state j if the process started from the state i and exactly q transitions occurred. There is a vector of such probabilities: $S_i(q) = [\mathbf{Pr}_{i1}(q), \mathbf{Pr}_{i2}(q), \ldots, \mathbf{Pr}_{i|\Omega_{attr}|}(q)]$, where $|\Omega_{attr}|$ is the number of elements in the domain Ω_{attr}. $S_i(q)$ can be found using Kolmogorov-Chapman's equation [30]. Assume that we know the initial value x_i of the process of the attribute at t_0. Thus, only $\mathbf{Pr}_{ii}(0) = 1$ and others are 0, i.e., $S_i(0) = [0, 0, \ldots, 1, \ldots, 0]$. The value of the vectors at t' is

$$S(t') = S(t_0) \cdot \mathbf{Prob}^q \tag{15}$$

where **Prob** is a transition matrix composed by probabilities of transitions from a state i (row) to a state j (column), $\mathbf{Prob}^q$ shows the matrix in power q.

2) Continuous-time Markov Model: Now, we consider a slightly different situation when we know only the time passed from the last check of an attribute. We assume that the average time between changes of the attribute value is exponentially distributed with the rate parameter ν. This assumption allows modelling the behaviour of attribute values using CTMC. We define the rate parameter ν_i of an exponential distribution for the time of jumping from the state i to another state and the average life-time $\frac{1}{\nu_i}$ of the attribute in state x_i. Also p_{ij} is the one-step transition probability (the probability that the process makes a direct jump from a state i to a state j without visiting any intermediate state). Using ν_i and p_{ij} we can evaluate the probability $\mathbf{Pr}_{ij}(\Delta t)$ of the attribute transition from the state i to the state j during time interval Δt.

The transitions between the states are described with the infinitesimal transition rates ($q_{ij} \in Q$). The infinitesimal transition rates are defined as

$$q_{ij} = \nu_i \cdot p_{ij},\ \forall i, j \in I \text{ and } i \neq j \tag{16}$$

The infinitesimal transition rates uniquely determine the rates ν_i and one-step transition probabilities p_{ij}:

$$\nu_i = \sum_{\forall j \neq i} q_{ij},\ p_{ij} = \frac{q_{ij}}{\nu_i} \tag{17}$$

We apply a *uniformization* method to compute transient state probabilities $\mathbf{Pr}_{ij}(\Delta t)$ [14], [30]. The method replaces a continuous-time Markov chain by a discrete-time analogue, which is more suitable for numerical computations. The uniformisation starts with replacing the transition rates of Markov chain ν_i with a sole transition rate ν, such as $\nu \geq \nu_i,\ \forall i \in I$, where I is the set of nodes of the continuous-time Markov chain. If $\nu = \max_{\forall \nu_i \in I} \nu_i$, then the one-step transition probabilities of the discrete-time Markov chain are defined as

$$\overline{p}_{ij} = \begin{cases} \frac{\nu_i}{\nu} p_{ij} = \frac{q_{ij}}{\nu}, & \forall i \neq j; \\ 1 - \frac{\nu_i}{\nu}, & \forall i = j \end{cases} \tag{18}$$

Now we have all required parameters for the computation of $\mathbf{Pr}_{ij}(\Delta t)$ and we skip the mathematical proofs, which can be found here [30, pages 167-168].

$$\mathbf{Prob}(\Delta t) = \sum_{n=0}^{\infty} e^{-\nu \cdot (t' - t_0)} \cdot \frac{(\nu \cdot (t' - t_0))^n}{n!} \cdot \overline{p}_{ij}^{(n)} \tag{19}$$

for $\forall i, j \in I$, $t' > t_0$, and $\overline{p}_{ij}^{(n)}$ can be recursively computed starting with $\overline{p}_{ii}^{(0)} = 1$ and $\overline{p}_{ij}^{(0)} = 0$ for $i \neq j$ from

$$\overline{p}_{ij}^{(n)} = \sum_{x_k \in I} \overline{p}_{ik}^{(n-1)} \cdot \overline{p}_{kj},\ n = 1, 2... \tag{20}$$

For fixed $t' > t_0$ the infinite series can be truncated because of the negligible impact of the residue. The truncation number U (upper limit of summation) in Formula 19 is chosen as

$$U = \nu \cdot t' + c \cdot \sqrt{\nu \cdot t'} \tag{21}$$

for c with $0 < c \leq c_0(\varepsilon)$, ε is a tolerance number [30, p. 169].

$\mathbf{Prob}(\Delta t)$ is a matrix of all possible transition probabilities probabilities after time Δt passed, $\mathbf{Pr}_{ij}(\Delta t)$ is the element on the crossing of the i^{th} row and the j^{th} column.

B. Convergence of a Markov Chain to the Steady State

If the attribute behaviour follows the Markov property, the probabilities of the attribute to be in a certain state converges to a stationary (steady state) steady state distribution. We can find the number n_{st} of transitions when the distribution of probabilities differs from the steady state distribution by any small value [22].

$$n_{st} = \tau(\epsilon) \leq \lceil \frac{\ln \epsilon}{\ln(2c)} \rceil T,\ \tau(c) \leq T,\ c < 1/2$$

$$\tau_{a_s}(t) = \min\{t : \Delta_{a_s}(t) \leq \epsilon\},\ \tau(\epsilon) = \max_{a_s \in \Omega_{attr}} \tau_{a_s}(\epsilon)$$

$$\Delta_{a_s}(t) = \|p^t_{a_s} - \pi\| = \frac{1}{2} \sum_{a_s \in \Omega_{attr}} |p^t_{a_s}(a) - \pi(a)|$$

Where $\Delta_{a_s}(t)$ is a distance between distribution $p^t_{a_s}$ and steady state distribution π, when process starts from the initial state a_s and t transitions occurred.