KAPUER: A Decision Support System for Privacy Policies Specification

Arnaud Oglaza, Pascale Zaraté, Romain Laborde

DOI 10.1007/s40745-014-0027-3

KAPUER: A Decision Support System for Privacy Policies Specification

Arnaud Oglaza · Pascale Zarate · Romain Laborde

Abstract We are using more and more devices connected to the Internet. Our smartphones, tablets and now everyday items can share data to make our life easier. Sharing data may harm our privacy and there is a need to control them. However, this task is complex especially for non technical users. To facilitate this task, we present a decision support system, named KAPUER, that proposes high level authorization policies by learning users' privacy preferences. KAPUER has been integrated into XACML and three learning algorithms have been evaluated.

Keywords Decision support · Access control · Privacy

1 Introduction

Nowadays, our relation with computers is no more limited to the use of a personal computer that can access the Internet with a wire connection. A study realised by GFK/Mediametrie published in November 2013 [1] shows that the number of houses equipped with more than one device (personal computer + smartphone + tablet) has more than doubled and reached 4.7 millions houses in France. In addition, smartphones and tablets have now enough processing and storage capabilities to host many applications. For example, french people have an average of 32 applications installed

A. Oglaza (✉) · P. Zarate · R. Laborde
Institut de Recherche en Informatique de Toulouse (IRIT), Universite Paul Sabatier,
118 Route de Narbonne, 31062 Toulouse Cedex 9, France
e-mail: oglaza@irit.fr

P. Zarate
e-mail: zarate@irit.fr

R. Laborde
e-mail: laborde@irit.fr

in their Android smartphones according to a survey made by Google in 2013.[1] This number grows to 40 in countries like Korea or Switzerland. Furthermore, the number of devices connected to networks is going to increase with the Internet of Things. Various studies show that there are between 15 and 20 billions "things" connected to the Internet and this number is expected to reach between 50 and 80 billions in 2020 [2,3]. All these connected things and applications can process and share data related to their owners. Thus, every owner of these things will have to control them to protect their privacy.

It is now a priority to provide people with tools allowing them to understand issues of privacy and the complexity of protecting their personal data. Various initiatives have emerged from this perspective [4]. Some works have proposed to help people to understand the risks attached to the disclosure of data through serious games like 2025 ex-machina [5]. Project Platform for Privacy Preferences [6] has standardized websites privacy policies to allow people understand how websites process their data. These policies are then evaluated with users preferences by an ad hoc mecanism. The same objective is pursued by Kelley et al. [7]. They noticed that people understand nutrition labels found on food package. So they proposed a similar solution to display privacy policies. Inglesant et al. [8] have presented a constrained natural language to ease the understanding of authorization policies. Stiepen et al. [9] worked on a non technical notation to facilitate the understanding of XACML authorization policies. All these works are important to help people to understand the risks they face and to let technical documents like privacy policies or authorization policies understandable to everyone. However, few works focus on helping people to design and write authorization policies to protect their privacy.

A first approach to assist in the design and the writing of authorization policies consists in a graphical interface where users can modify their authorization rules. An example of this approach is Privacy Guard Manager, which is a component of an alternative Android distribution called CyanogenMod [10]. This interface provides a dashboard with all information about permissions given to each application. It allows users to set granted and denied permissions to each applications. The benefits of this approach are (1) the use of the graphical interface doesn't require any technical skills to define the authorization policies and (2) the possibility to manage permissions at a fine grained level. But this approach is grabbling with the issue of scalability. Indeed, Privacy Guard Manager can only express low level rules. To quantify this problem, we have analyzed the average number of permissions to handle on an Android smartphone. Given that there are an average of 32 applications on an Android smartphone owned by french people and an application requests in average of 11.4 permissions where 5.72 have an impact for privacy (we have obtained these values by analyzing the permissions of the 50 most downloaded free applications in the Android market), a user has to manage 364 permissions where 183 have an impact on his privacy.

This problem of scalability has already been studied in various research works on access control models. Indeed, administrators have already had this problem. For example, the RBAC model [11] ease the management of permissions by grouping them

[1] http://think.withgoogle.com/mobileplanet/fr/.

depending on the role that users have in an organization. The abstraction of roles limits the number of rules. Other notions and abstractions have been introduced in access control models to facilitate the definition and management of authorization policies especially for privacy like the concept of purpose [12], sensitivity of a resource [13], trust [14], accuracy or consent [15]. These access control models offer the possibility to write high level rules that are suited to complex environment. However, manipulating these abstractions is a complex task that requires an analysis step before writing authorization rules. As consequence, it isn't possible for non technical users to write policies according to these models. Furthermore, defining a generic user interface for easily writing policies with abstract notions is a difficult task [16]. How to avoid beginner mode (simple but limited) versus expert mode (complete but complex)?

Based on this observation, we present a new approach that allows a non technical user to write high level policies while limiting the required cognitive load (i.e. design phase and interface specification). Our proposition is a system named KAPUER (KAPUER is an Assistant for Protection of Users pErsonnal infoRmation) who uses techniques from decision support to help users to write abstract authorization rules. KAPUER analyzes low level permissions granted by a user to learn his privacy preferences and proposes to him high level rules corresponding to these preferences. The user can (1) accept a proposed rule that will be implemented by the authorization system, (2) deny or modify it if the rule doesn't fit his behavior. We have integrated KAPUER in the architecture of the XACML authorization management system that offers a high degree of flexibility both in terms of integration and expression of policies. This work extends the proposition made in Oglaza et al. [17] that was limited to the integration of a desicion support system for protecting privacy in an Android environment.

The rest of the article is structured as follows. In Sect. 2, we introduce decision support systems. In Sect. 3, we defined our problem solving model to learn users preferences in terms of privacy. In Sect. 4, we present the integration of KAPUER in XACML. We conclude and discuss future improvements in Sect. 5.

2 Introduction to Decision Support System

Decision support systems (DSS) have been introduced in the seventies. Gorry and Morton were the first researchers to employ this term in 1971 [18].This approach combines mathematical models to analyze the behavior of decision makers and computers for their interactivity and visualization techniques. With the actual power of hardwares and softwares, decision support systems can help users to make more and more complex decisions based on more information than a person can do.

In order to help users, a DSS has to understand him. Thus the DSS and the user interact with each other. Through these interactions, the DSS can analyze decisions made by the user with learning algorithms. The main goal of a DSS is to assist users in their decisions making. It does never make decisions in their stead. The system is here to aid users, it does not replace them.

There are many ways to help a user. This can be done by explaining to him the problem he faces, giving him the causes of the problem or by decomposing a com-

plex problem into multiple subproblems easier to deal with. For example, there exist dashboards in stock market that aggregate financial parameters into indicators. These indicators ease the decision making process. Another way to help a decision maker is to propose different solutions or items to him. This kind of system is called a recommender system. For instance, Amazon uses a recommender system to suggest their clients items that might interests them based on their previous shopping, navigation history and also products bought by other clients.

Building a efficient recommender system requires to consider several aspects. It needs to present satisfying solutions as fast as possible. If the system takes too much time to return a proposition, or propositions are ludicrous, users will forsake it. To understand the user and propose him solutions, it needs to interact with him. These interactions are mandatory but need to be minimised. The difficulty in building a good recommender system is finding the good balance between accurately learning users preferences and limiting as much as possible interactions.

Three different approaches exist to build a recommender system [19]. Collaborative filtering uses information about other users to find the best propositions to make. The main advantage of this approach is that it can propose solutions quickly to users with few information on their preferences. Since proposed solutions are based on other users' preferences, they aren't very accurate. Similarities and differences between users are calculated using algorithms like the k-nearest neighbors [20].

The second approach, called content based, is strictly about preferences of each user and describes items as a set of attributes. For example, a movie can be described by its title, its release date, its producer and its actors and actresses. Each movie can be represented by these attributes and knowing preferences of the users allows the DSS to make recommendations based on these attributes [21].

The last approach is a combination of the first two. This hybrid approach uses at the same time collaborative filtering to find habits of other users and the content based technique to identify items appreciated by the target user.

We think that the content based filtering is the most appropriate approach for our system because we strongly believe in the personal meaning of privacy. Similarities of privacy preferences between two different users for a specific personal information doesn't imply same behavior for another one. As consequences, propositions made for one won't be necessarily relevant for another. Furthermore, keeping preferences locally was a strong requirement. Several works emphasize confidentiality problems following the use of a collaborative recommender system [22,23]. In Calandrino et al. [24], the authors explain that it is possible, using information acquired by someone's actions, to infer others information about his behavior. Content based recommandations allow to avoid these confidentiality problems since users preferences aren't shared. In addition, privacy decisions are complex. We propose to assess authorization policies through multiple criteria. Indeed, a mono-criterion approach isn't possible since users privacy preferences are always based on several criteria.

A decision support system has practical goals. No strong hypotheses on the environment are made. Simon proposed a base for decision making called bounded rationality [25]. Rational decisions are often impossible to make because sometimes some criteria used to find the optimal choice are contradictory. For instance, someone wants to go from Toulouse to Paris as fast as possible but also wants to minimize the pollution of

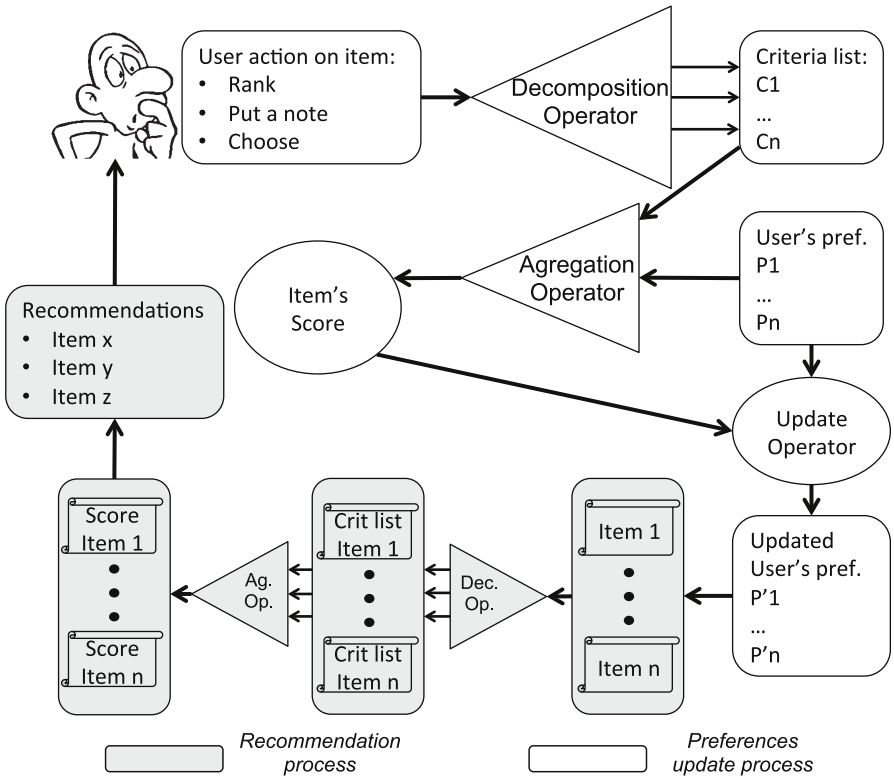


Fig. 1 Generic process of a recommender system

his trip. Although flying is the fastest way to travel, it is also one of the most pollutant (600–900 km/h for 145 g/CO_2/km). Train is not as fast as plane but is more ecological (90–250 km/h for 13 g/CO_2/km).[2] There is not one solution where speed is maximal and pollution minimal. Thus, the most satisfying solution has to be chosen instead of the optimal one.

A recommender system is a cyclic process (Fig. 1). It starts with an action of the user who can rank or mark items or choose one item directly. This action allows the system to decompose the item into criteria. A criterion is the combination of an identifier and a value. Criteria gathered by the system are the attributes of the item and the value will depend on the action of the user. This step is done using a decomposition operator, part of a method used in DSS based on multiple-criteria decision making approach (MCDM). Then the criteria list of the item is associated to the user's preferences. The user's preferences are represented by a list of criteria defining his behavior. The whole user's preferences compose the profile of the user. The user's preferences are used by the system to ponder each criterion of the item. Then, the system is able to calculate the score of the item by aggregating all the criteria. An aggregation operator is used to

[2] http://www.consoglobe.com/les-14-modes-de-transport-les-moins-polluants-cg.

obtain the item's score. The last step of the learning part is to capitalize the knowledge acquired from this new item. In the update step, the user's preferences are calculated according to the item's score. All those steps put together form the learning phase of the process, the next phase determines the recommendations the system presents to the user.

Once the preferences have been updated, the system can, with the same method as during the learning, calculate the score of items. It decomposes items in criteria and aggregates them with user's preferences. When the scores of all items are available, the system displays the recommendation to the user and the process starts again.

3 Privacy Problem Solving Model

In this section, we present the problem solving model we use to allow the authorization system and the decision support system to communicate with each other.

3.1 Criteria

A criterion represents the basic element of an access request to a protected resource. The criterion can described the name of the user, his age, the name of the resource, the name of the action, etc. Then the request "John wants to read the calendar" is composed by three criteria: John, read and calendar. The system's set of criteria is noted *CR*. A criterion is composed by an identifier and two values corresponding to the user's preferences, one for accept, one for deny. Indeed, if users' preferences were modelised only by one value, it would be difficult to analyze the meaning of a low value. Does this low value mean that this criterion isn't favorable for disclosure or does it mean that this criterion hasn't been updated often and the system has nearly no information about it? It isn't possible to answer this question. We use two values for each criterion to resolve this problem:

- The first one, $g^t : CR \rightarrow [0, \infty]$ represents the user's preference for this criterion in favor of disclosure at time t.
- The second one, $f^t : CR \rightarrow [0, \infty]$ represents the user's preference for this criterion in detriment of disclosure at time t.

We have chosen to have an incremental score calculation method. So during their update, $g^t(x)$ and $f^t(x)$ can only increase. The preferences learning is a continuous phase so if the value of $g^t(x)$ is high, it doesn't necessarily mean that the presence of criterion x is a strong reason for the user to allow disclosure. This can also mean that criterion x has been often updated. To identify a criterion's score, the system has to calculate either:

- $s_D^t(x)$ corresponding to the score of criterion x at time t in favor of disclosure. This score comes from the difference between the value $g^t(x)$ and $f^t(x)$:

$$s_D^t(x) = f^t(x) - g^t(x) \tag{1}$$

- $s^t_{nD}(x)$ corresponding to the score of criterion x at time t in detriment of disclosure. This score comes from the difference between the value $f^t(x)$ and $g^t(x)$:

$$s^t_{nD}(x) = g^t(x) - f^t(x) \tag{2}$$

With this calculation method, $s^t_D(x)$ and $s^t_{nD}(x)$ give the position of the criterion x about disclosure for the user. A low value of $s^t_D(x)$ or $s^t_{nD}(x)$ indicates that there is no reason that justify preferences for disclosure or non-disclosure. On the contrary, a high value of $s^t_D(x)$ or $s^t_{nD}(x)$ indicates clear reason of a preference for one or the other action.

3.2 Classes of Criteria

Access control models propose key elements to take into account into policies like:

- the visibility: who wants to access the resource (a friend, a colleague, a stranger, etc.)
- the temporal aspect: when is the request made (hour of the day, day of the week, work time, etc.)
- the spatial aspect: where is the user (at home, at work, etc.)
- the retention: how the resource will be stored (how many time, who will have access to it, etc.)
- the purpose: how the resource will be used (for statistics, to be sold, etc.)

To express these elements, we introduce the notion of class. Each criterion is part of a class of criteria with the relation $ACC \subseteq CR \times C$ where the set of class of criteria is noted C. Because the system is generic, classes of criteria aren't fixed and all possible classes can be created. To simplify our notation, we define the function *class* that returns the set of criteria associated to one class:

$$\begin{aligned} &class : C \rightarrow \mathcal{P}^{CR} \\ &x \mapsto \{y \in CR | (y, C) \in ACC\} \end{aligned} \tag{3}$$

3.3 Meta-criteria

We defined the notion of meta-criterion to represent abstractions of access control models like the role in RBAC [11], views and activities in OrBAC [15], purposes hierarchies in PRBAC [12], etc. The prefix meta in the scientific vocabulary allows to designate a higher level of abstraction. Here, the level 0 corresponds to criterion and all levels higher than 0 are for meta-criteria. A meta-criterion is a criterion with a higher level of abstraction than one or many criteria of the same class. The set of meta-criteria is noted $MCR, MCR \subset CR$. The two sets can't be equal because we consider that the system can at least have one criterion at level 0, so it can't belong to MCR. A meta-criterion allows to group many criteria having a common feature. For example, criteria "John" and "Jane". These two persons have something in common: they are parents. So we can define the meta-criterion "Parent" that regroups "John"

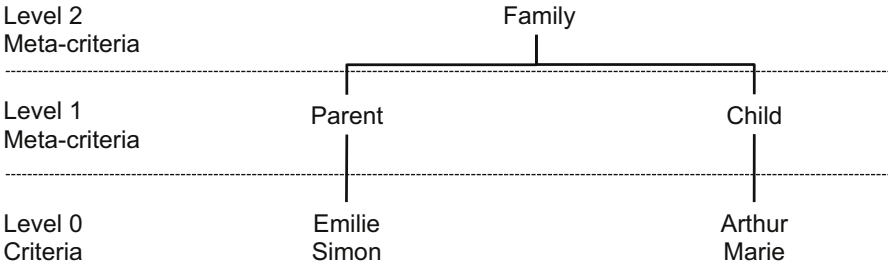


Fig. 2 Example of a hierarchy of criteria

and "Jane". At the same time, we can define another meta-criterion, "Family" who has a higher level of abstraction than "Parent". Values $g^t(x)$ and $f^t(x)$ of a meta-criterion are specific to this meta-criterion.

A meta-criterion is also a criterion, then it is possible for each class of criteria cl to create a hierarchy $H_{cl} \subseteq CR \times MCR$ between these criteria such as $\forall(c_1, c_2) \in H_{cl}, class(c_1) = class(c_2)$ (Fig. 2). There are two cases when a criterion hasn't a meta-criterion in its description:

- the criterion is at the top of the hierarchy (for instance, the criterion "Family" in Fig. 2).
- the criterion is independant and can't be associated to a meta-criterion.

3.4 Groups of Criteria

In order to analyze relations between criteria to understand users preferences in details, we define groups of criteria. A group of criteria is an association of n criteria. A group of criteria has his own preferences values, they are not depending on the values of the criteria composing the group. Each criterion or meta-criterion composing a group of criteria must belong to a different class of criteria. For instance, criteria "Parent" and "Calendar" are from two different classes so we can create the group of criteria {Parent, Calendar}. The set of groups of criteria G is defined by:

$$G \subseteq \mathcal{P}(CR)$$

A group of criteria is composed by at least two criteria.

$$\forall g \in G, \quad |g| \geq 2$$

Two criteria of a same group can't belong to the same class.

$$\forall g \in G, \quad \forall(c_1, c_2) \in g \times g, c_1 \neq c_2 \Rightarrow class(c_1) \neq class(c_2)$$

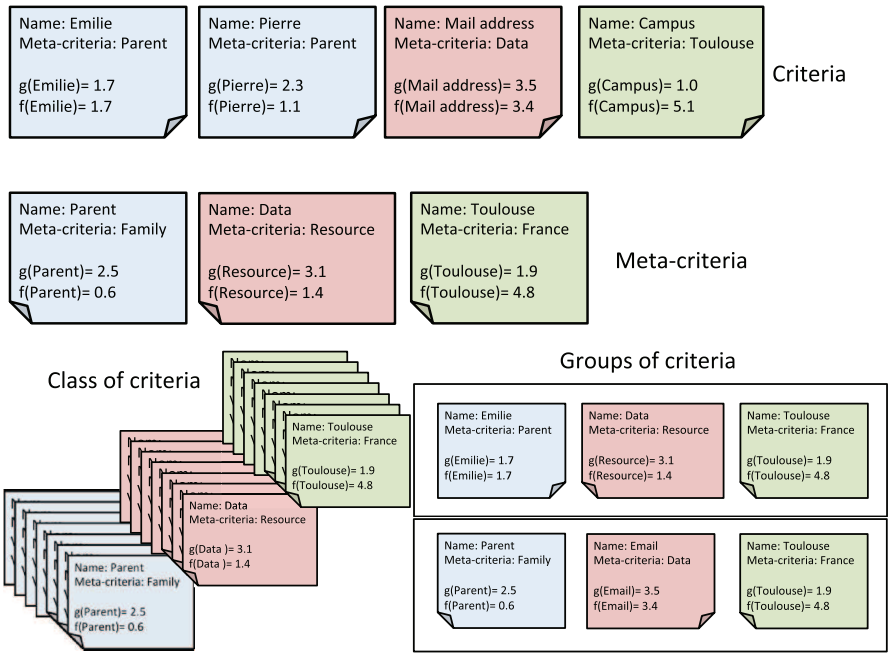


Fig. 3 Notions about criteria

3.5 Example of Formalization

Figure 3 illustrates the different notions we have previously defined.

The set of meta-criteria MCR and the set of criteria CR:

$$MCR = \{Son, Parent, Family, Data, Resource, Toulouse, France\}. \quad (4)$$

$$CR = MCR \cup \{\text{Max}, Pierre, Email, Campus\} \quad (5)$$

The relation of hierarchy between a criterion and a meta-criterion H_{cl}:

$$\begin{aligned} H_{cl} = \{&(\text{Max}, Son), (Pierre, Parent), (Email, Data),\\ &(Campus, Toulouse), (Parent, Family),\\ &(Data, Resource), (Toulouse, France)\}. \end{aligned}$$

The set of classes of criteria C:

$$C = \{Who, What, Where\}. \quad (6)$$

The relation between a criterion and a class of criteria ACC:

$$\begin{aligned} ACC = \{&(\text{Max}, Who), (Pierre, Who), (Email, What), (Campus, Where),\\ &(Son, Who), (Parent, Who), (Data, What), (Toulouse, Where),\\ &(Family, Who), (Resource, What), (France, Where)\}. \end{aligned}$$

The set of groups of criteria G:

$$G = \{(Parent,\ Email,\ Toulouse), (\text{Max},\ Data,\ Toulouse)\}.$$

A time t, the disclosure value g and non-disclosure value f of each criterion:

$$\begin{aligned}
g^t(\text{Max}) &= 1.7 \text{ et } f^t(\text{Max}) = 1.7 \\
g^t(Pierre) &= 2.3 \text{ et } f^t(Pierre) = 1.1 \\
g^t(Email) &= 3.5 \text{ et } f^t(Email) = 3.4 \\
g^t(Campus) &= 1.0 \text{ et } f^t(Campus) = 5.1 \\
g^t(Parent) &= 2. \text{ et } f^t(Parent) = 0.6 \\
g^t(Data) &= 3.1 \text{ et } f^t(Data) = 1.4 \\
g^t(Toulouse) &= 1.9 \text{ et } f^t(Toulouse) = 4.8
\end{aligned}$$

4 KAPUER Integration in XACML

The XACML standard [26] is an XML specification defined by OASIS for the expression of access control policies. XACML provides an universal language for description of policies with the form: who can do what and when? The access control policy allows to define permission to entities (users or applications) on resources (data, services, etc.). XACML is a powerful expression language where all security information can be considered as an attribute of the subject, the resource, the action or the environment. Furthermore, this language uses a policy decision point/policy enforcement point type of architecture to enforce the access control: a request/answer type of protocol gives the possibility to express requests and appropriate answers. The grey elements on Fig. 4 show the key elements defined in XACML:

- the policy decision point (PDP) is a logical entity that takes authorization decisions. Requests for protected resources are evaluated according to policies written in XACML.
- the policy enforcement point (PEP) is a logical entity what applies the authorization decisions taken by the PDP. The PEP is the guardian of the resource and is the one that technically realizes the access control. It receives requests, translates it in XACML and sends it to the PDP. Then, it waits for the answer of the PDP to apply the decision.
- a policy database, a simple database where XACML policies are stored.

The last component of this architecture (the white one) isn't part of XACML. It is the one from our proposition, the DSS for protecting privacy. The next subsection describes the different steps of the process by following the number in the figure.

4.1 Step 1: Interception and Translation of a Request in XACML

A request occurs when an entity asks to have access to one or more protected resources. These resources can be either data related to the user (name, email, etc.) or services

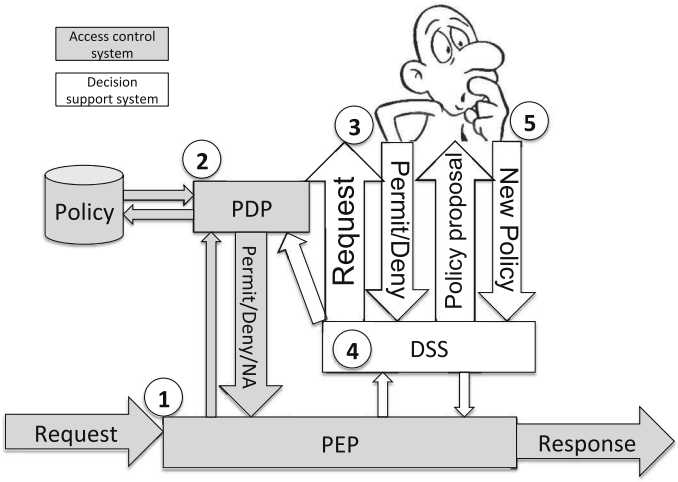


Fig. 4 KAPUER's architecture integrated in XACML

```
<Request>
   <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>fr.irit.siera.testing</AttributeValue>
      </Attribute>
   </Subject>
   <Resource>
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>android.permission.READ_SMS</AttributeValue>
      </Attribute>
   </Resource>
   <Action>
      <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>access</AttributeValue>
      </Attribute>
   </Action>
   <Environment>
      <Attribute AttributeId="When-Day" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>2</AttributeValue>
      </Attribute>
      <Attribute AttributeId="When-Hour" DataType="http://www.w3.org/2001/XMLSchema#string">
         <AttributeValue>10</AttributeValue>
      </Attribute>
   </Environment>
</Request>
```

Fig. 5 Example of a XACML request

that provide data about the user (for instance the gps gives users coordinates). When a request is made, it is intercepted by the PEP. It has to translate it in XACML, send it to the PDP, wait for a decision and then apply it. KAPUER is generic so it is possible to use any available information in a request and translate it as attributes. But all possible attributes don't have to be present in a request. The result of a translated request is a XML file containing attributes under the tags subject, resource, action and environment.

Figure 5 shows an example of a request translated by a PEP. It has four different parts:

- the subject, the entity who ask for an access to a resource. Here it's an application, the attribute's value is the name of the application "fr.irit.siera.testing".

- the resource, the data requested. Here the value of the attribute is the required permission to have access to the data. “android.permission.READ_SMS” is the required permission to read the user’s SMS.
- the action is the type of action the entity wants to have with the resource. Here the application wants to access the ressource so the value of the attribute is ”access”.
- the environment is all other contextual information available at the time of the request. Here the only available information are the day and the hour of the request with the value ”2” and ”10” corresponding to the second day of the week and the tenth hour (tuesday between 10 a.m. and 11 a.m.).

This request is then sent to the PDP who continues the authorization process.

4.2 Step 2: Evaluating the Request According to the Policy Database

The second step of the access control process starts when the PDP receives a XACML request from the PEP. Its goal is to determine if an existing rule in the policy base can be used to make a decision for that request and if it’s the case, to give the decision. A policy is a set of logical expressions, called rules, where the free variables are attributes identifier. When the PDP receives a request formed by a set of couples {attributes identifier, value}, it can make the substitution and evaluate some rules of this policy. When all the attributes and values of a request correspond to those of a rule, the decision associated to the rule is given to the PEP. This decision can either be “PERMIT” if the request is accepted of “DENY” if it is refused. In the case where all attributes and values don’t correpond to any rules of the policy, the decision “NOT APPLICABLE” is sent to the PEP. This situation is a problem in a classic authorization system because it answer to an undefined case. Theoretically, a PEP without a decision needs to ask another PDP but usually, developpers implements the decision “NOT APPLICABLE” as the “DENY” one.

4.3 Steps 3 and 4: Interactions with the User to Learn his Preferences

Starting this step, the DSS takes place in the process. The decision “NOT APPLICABLE” is used when there is no authorization rule fitting the request. In this case, it could be possible to ask the user to write a rule in an editor. But it would need a design phase from the user to write a rule with abstract notions: he will have to take a complex decision.

We prefer to ask him to take a simpler decision limited to this request by interacting with him. Figure 6 is a screenshot showing an example of an interaction with the user. KAPUER informs the user that an entity (here application “fr.irit.tests”) wants to access a resource (his contact list) and asks him if he accepts or refuses to share this resource. The system uses the decision of this interaction to understand the user’s behavior. Two actions are presented to the user, the disclosure D and the non-disclosure nD. Once the user has taken his decision, the couple {request, action} is sent back to the DSS that analyzes and uses it to update the user’s preferences. These preferences are a representation, by a set of criteria, of the user’s prefered authorization policy.

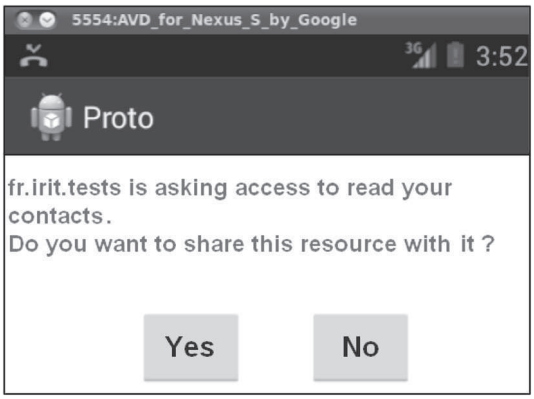


Fig. 6 Example of an interaction with the user

The notion of attribute in XACML being very close of criterion (described in Sect. 3), it is possible to transform a request into a list of criteria.

This way, the DSS makes a continuous learning of the preferences to be as close as posssible to the inkling of authorization policy wanted by the user. Then in our case, decision "NOT APPLICABLE" means that the DSS needs to keep learning user's preferences and keep interacting with the user to inform him and obtain new information about his behavior. This information will allow to update values of the criteria used in the request but also values of their meta-criteria and then have a better view of the user's preferences. When the DSS has learned enough the user's behavior, he is able to propose a new authorization rule to the user (Sect. 4.4).

4.4 Step 5: Abtract Rules Proposition to the User

The main objective of KAPUER is to make high level rules propositions to the user. However, the system must not propose random rules. First of all, a rule needs to have a high level of abstraction to cover a large number of requests. It limits the number of rules manage for the user, avoiding scalability issues like in CyanogenMod (see Sect. 1). Moreover proposing high level rules limits interactions with the user. However, the level of abstraction must fit with the user privacy preferences.

KAPUER makes propositions when the the proposed action is stricly prefered to its contrary. If there is no preference between the two actions, the system will not propose any rule. Two cases bring KAPUER to this situation:

- when the representation of the user's preferences is not accurate enough. Here the system lacks information and needs to interact more with the user.
- when the user doesn't have a fixed behavior and doesn't react in the same way to identical situations. In this case, it's not possible for the system to infer the user's behavior neither to propose any rule.

To manage preferences, we use a perfect relational system of preferences [27]. It is composed by the two following binary and transitive relations:

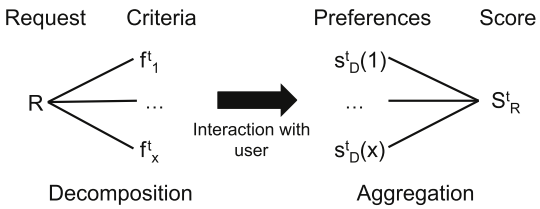


Fig. 7 Multi-criteria decision analysis

- the indifference $\sim$ or non-preference is a lack of reason to justify a preference for one action or the other:

$$\sim: a \sim a' \Leftrightarrow aIa' \tag{7}$$

  Relation I is reflexive and symmetric.

- the strict preference $\succ$ is when there are reasons to justify the preference of an action instead of the other:

$$\succ: a \succ a' \Leftrightarrow aPa' \tag{8}$$

  Relation P is irreflexive and asymmetric.

A proposition of a high level rule is built within an analysis of the request and the decision taken by the user. To do this analysis, we use a method called multi-criteria decision analysis [28] (Fig. 7).

Each request is decomposed into criteria which are then aggregated by an aggregation operator. The user's preferences are used in the step to ponder criteria. The result of the aggregation provides a score S^t_R of request R which reflects the degree of understanding of user's preferences upon the request. With this score and the user's decision, KAPUER can update the values of all criteria and meta-criteria linked to R.

In order to know if an action has a stric preference for the user, KAPUER calculates score S^{t+1}_R of request R with the updated values of criteria and meta-criteria $f^{t+1}(x)$ et $g^{t+1}(x)$. This new score is then compared to λ, a parameter corresponding to the threshold between indifference and strict preference. If S^{t+1}_R is lower than λ, KAPUER identify a situation of indifference and no proposition is made to the user. If S^{t+1}_R is above λ, this situation is strict preference and a proposition can be made to the user. λ is a parameter that impacts the proposition making speed. The lower is its value, the faster propositions are made. On the contrary, the higher its value is, the slower propositions are made. The value of λ has been determined after simulation (view Sect. 6). We are actually working on a dynamic tuning of λ to be specific to each user's behavior.

In the case where S^{t+1}_R is above λ, the system proposes a new rule to the user. This is done through a new interaction with the user. During this interaction, KAPUER asks the user if he wants to add a new rule to the policy database. The attributes of the rules are the criteria or meta-criteria of the proposition. A decision is also linked to the rule and the user can accept, decline or modify it.

Even if attributes of a proposition presented to a user should be relevant, we still let the possibility to modify one or many of these attributes. Either the attribute is a

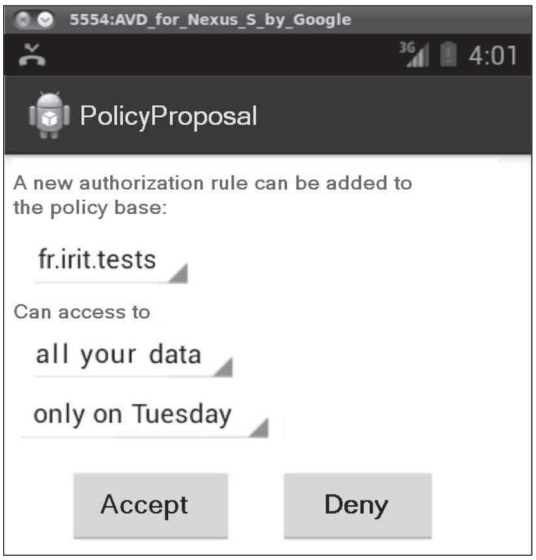


Fig. 8 Example of a proposed rule

criterion and the user can choose its meta-criterion instead to have a higher level rule or the attribute is a meta-criterion and the user can choose the associated criterion instead when he doesn't want an abstraction. Figure 8 shows an example of an interaction for a proposition. Here the proposed rule asks if the user agrees to share his data with the application "fr.irit.tests" every thursday. This rule has one abstract attribute: the resource. The user can make the rule more abstract to all applications or all day of the week. This way, if the rule doesn't fit exactly the user's behavior, he can change it easily without a specification phase nor a writing phase.

If the user accepts this rule, KAPUER writes the corresponding XACML rule into the policy database and gives the decision of the user to the PEP. If the rule is denied, only the decision of the request is given to the PEP. Once the PEP has the decision, he can send it to the entity who made the request.

5 Initializing the System

Before the beginning of the preferences learning, the system can be initialized to increase his efficiency. The initialization of preferences is used to have information about the user to tune some parameters of the calculus and allows it to adapt to each user. With some questions, it's possible to know if the global behavior of an user is simple (disclose all resource without any condition or on the contrary no disclosure at any cost) or if his behavior is more complex and depends on situations. In the first case, KAPUER can increase values in the criteria's update phase to propose rules faster or increase update's values of meta-criteria to work with higher level rules. On the contrary, if the user has a complex behavior, it's necessary to have a finer learning and to lower update's values of criteria or to propose lower level rules.

We started working on the initialization and prepared a set of questions (42 in total) and we have proposed them to three differents groups of people each one with a different view of privacy. The first one was a group of law and computer science students. The second group was composed of PhD students in computer science and the last one was composed of smartphone and tablets users without any technical skills in computer science. The questions put users in situation and answers were their reactions in these situations. Results has shown that questions of these kind didn't give us relevant information on the user despite the large number of questions. We are actually working on a new set of questions.

6 Learning of Privacy Based Preferences

The phase of preferences learning has to be as fast as possible. Moreover, the number of interactions between KAPUER and the user has to be limited not to annoy the user. Using meta-criteria to create high level rules allow to decrease the number of rules created. But it's not enough. The number of interactions depends also on preferences learned and how fast the DSS learned them. The step of criteria aggregation to calculate the score of a request and the step of criteria's values update are the two important step that impact the learning speed. We have tested three different aggregation operators:

- the weighted mean, an operator used in the majority of DSS for his simplicity. Each criterion is evaluated independently from the other.
- the Choquet integral [29], an more complex operator which take into account interaction between criteria and the importance of each criteria to have finer results. We have used Kappalab [30], a plug-in for R to implement our Choquet integrals.
- our own operator, KAGOP (KAPUER AGgregation OPerator) [31] which is a between the weighted mean and the Choquet integral. It works not only with criteria but with groups of criteria (see Sect. 3.4). We use this operator to see if groups of criteria can help the system to find easily interactions between criteria.

To obtain enough data to compare the different learning approaches in real conditions, a lot of users, devices and times is needed. To overcome these constraints, we have developped a simulator. This tool allows us to implement a set of criteria with many classes along with meta-criteria and hierarchies for each class. Thus it is possible to generate a large number of random requests. This allows the simulator to simulate access control requests. The simulator can also simulate an user with a set of predefined behavior's rules. Then, each time the DSS has to interact with the user, the simulator is able to answer using these behavior's rules. The same way, the simulator is able to accept or decline proposition made by the DSS. We have run ten simulations of 200 random requests to compare the three different operators. We evaluate four different metrics. First the number of interactions. It shows the level of abstraction of each operator (Fig. 9). The more policies are created, the lower-level they are. Then, it indicates if an operator can adapt itself to a system with more criteria. The second metric is the number of requests processed by policies. It shows the learning speed. The more requests are handled by policies, the faster preferences are learned. The third metric is the level of completeness, i.e. the ratio between the number of requests covered by the poli- cies created and the number of requests possible with the behavior's

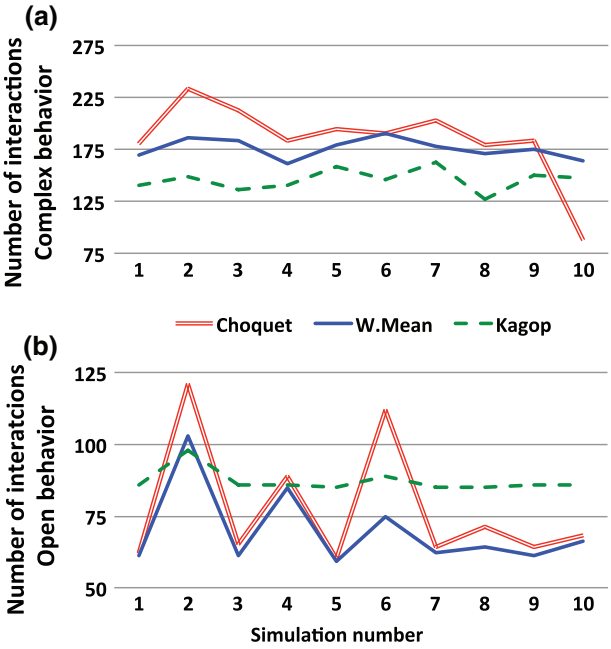


Fig. 9 Simulations results: number of interactions

rules that simulate the user. It shows, after 200 requests, the percentage of requests that will be handled by KAPUER. The last metric is the number of interactions made during the 200 requests. It's the sum of the number of policies and the number of requests non-processed by policies. For our first tests, we have implemented a list of criteria with three classes of criteria:

- **What** data to protect with six criteria. "Contact list" and "Calendar" with the same meta-criterion "Data". "Name" and "E-mail address" with the meta-criterion "Info". "GPS" and "Camera" with the meta-criterion "Service".
- **Who** wants to have access to data with nine criteria. "Jimmy", "Lee" and "Billy" with the meta-criterion "Family". "Bob", "Jay" and "Fred" with the criterion "Friend". "Pierrick" and "Mick" with the meta-criterion "Colleague" and "John" with the meta-criterion "Unknown".
- **When** is the access requested with fourteen criteria for each half-day and two meta-criteria ("Morning" and "Afternoon")

Finally, users are simulated by two different behaviors. The first one, B_{cx}, is complex. It agrees to share all ressources with members of the family all the time, with colleagues on morning, with friends on afternoon and doesn't share anything with unknowns. The system only needs to know what request he has to accept. If a request isn't managed by a rule, the system acts like there is a rule to deny disclosure. Then the user behavior can be formalized by:

- **Rule 1** Share *Data, Info* and *Service* with *Family* on *Morning* and *Afternoon*
- **Rule 2** Share *Data, Info* and *Service* with *Colleagues* on *Morning*

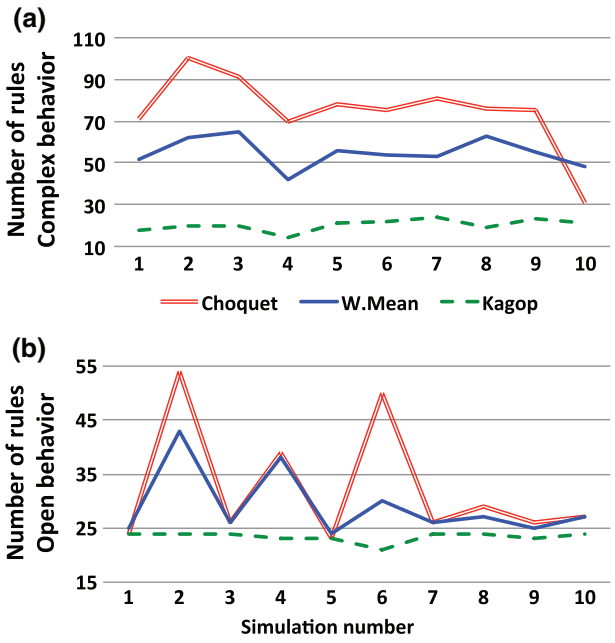


Fig. 10 Simulations results: number of created rules

- **Rule 3** Share *Data, Info* and *Service* with *Friends* on *Afternnon*

The second behavior, B_{op}, is open to all requests. We only have one rule for this behavior:

- **Rule 1** Share *Data, Info* and *Service* with *Family, Colleagues*, *Friends* and *Unknown* on *Morning* and *Afternoon*

The goal of each algorithm is to find all those rules as fast as possible with the requests and decisions of the simulated user. Results are shown in Figs. 10, 11 and 12 (warning, scales are different on each chart).

The results show that there isn't an operator that outperforms the others. None has the best results in all four metrics. KAPUER is interesting for users if it handles the more possible requests. After 200 requests, whatever the behavior, the three operators are above 80 % of completeness. So more than four out of five further requests will be handled by the system. The mixed operator even reaches 98.9 % with B_{op}. This level has to be confronted with the number of policies created. As we already said, the more policies are created, the lower- level they are. Then if we strongly increase the number of criteria in the system, the system needs more time to learn user's preferences. If the created policies are low-level, it will lead to a lower level of completeness. Then, if Choquet and the weighted mean have good results in those cases, our operator will have better results in a system with more criteria.

The other important point for users is to limit interactions. We can see that a complex behavior brings more interactions than an open one. The learning speed has an impact on interactions. Choquet and the weighted mean have more requests processed by

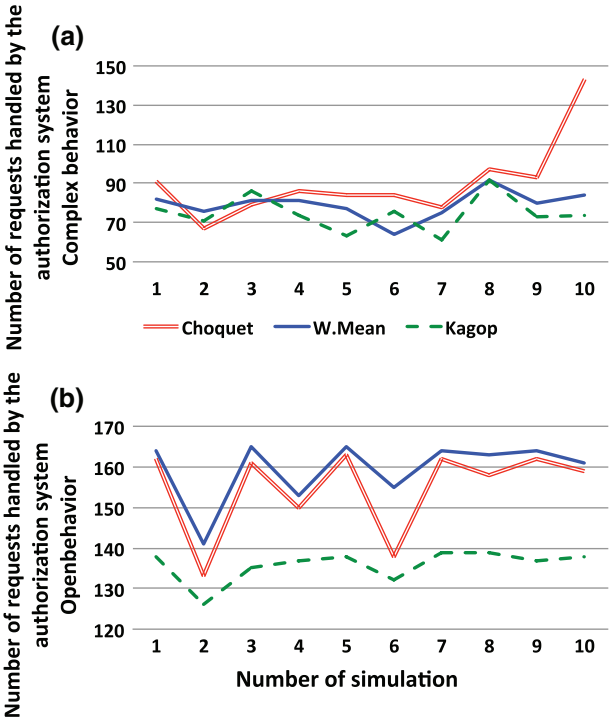


Fig. 11 Simulations results: number of requests handled by the authorization system

policies. It shows that they create policies faster than our mixed operator. But with B_{cx} they create much more rules, thus they require more interactions than our operator. Three times during the simulations, Choquet and the weighted mean have had peaks in the results. Way more policies are created because of their low-level of abstraction. As consequences, the level of completeness is also lower and it increases the number of interactions. On the contrary, whatever the behavior, our operator is more stable than the two others for the four types of metrics.

7 Conclusion and Future Work

Complexity in today's personal computing is increasing with the number and the diversity of connected devices. Now the problem of privacy is present and controling these systems is harder and harder for non expert people in administration of system or security. We have proposed in this article a new approach that combines a decision support system based on multi-criteria analysis named KAPUER with classical access control tools to help users to write high level authorization rules. We have explained how we integrated KAPUER into XACML, in particular how and when we interact with the user. Moreover, we have sudied and compared three aggregation operators for learning user's privacy preferences. Our own operator, Kagop, provides good results

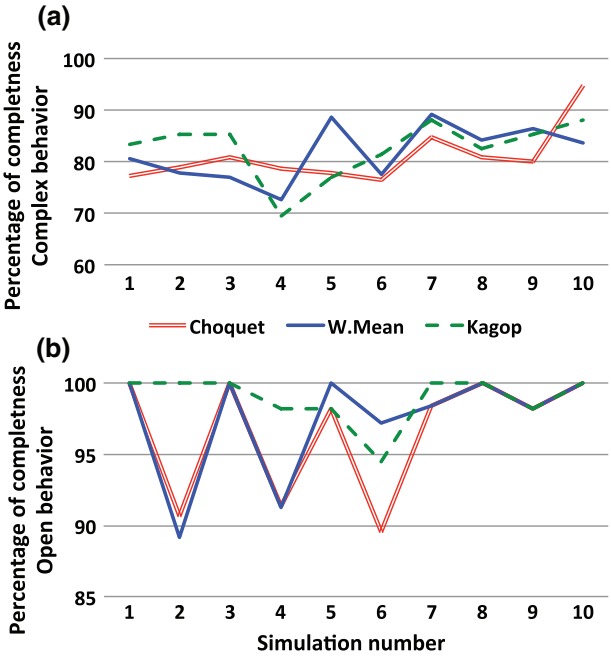


Fig. 12 Simulations results: Percentage of completness

and his behavior is more homogeneous than the weighted mean and the Choquet integral. The work on learning speed and efficiency can be improved on two aspects.

First, we have used these algorithms without any initialization. Convergence of these algorithms will be faster after an initialization of user's preferences. The study we performed on differents groups of people gave us experience and we are actually working on a new way to learn relevant information about the user.

The second aspect to be improved is the criteria values update. Thanks to our simulator, we are running more tests to determine the best combination between the function used to update criteria and the value λ of threshold between indifference and strict preference.

Finally, We are working on new and more complex simulations. We are implementing a realistic Android smartphone scenario involving more applications, more criteria and meta-criteria and more complex behavior. These simulations will strengthen the validation of our system.

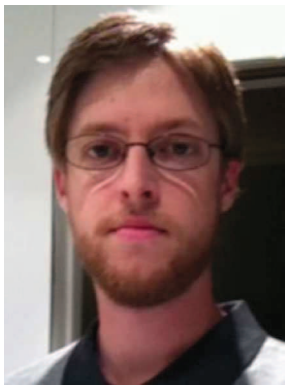
Acknowledgments This work is part of the French National Research Agency (ANR) project INCOME (ANR-11-INFR-009, 2012–2015).

References

1. GFK: gfk-médiamétrie référence des équipements multimédias au 3ème trimestre 2013 (2013). http://www.gfk.com/fr/news-and-events/press-room/press-releases/pages/gfkmediametrie-reference-des-equipements-multimedias-3eme-trimestre-2013.aspx.

2. EMC: the internet of things (2014). http://www.emc.com/leadership/digital-universe/2014iview/internet-of-things.htm. Accessed 22 Sept 2014
3. ZDNET: 80 milliards d'objets connectés en 2020 (2013). http://www.zdnet.fr/actualites/80-milliards-d-objets-connectes-en-2020-39793776.htm. Accessed 31 Jan 2015
4. Chabridon S, Laborde R, Desprats T, Oglaza A, Marie P, Marquez SM (2014) A survey on addressing privacy together with quality of context for context management in the internet of things. Annales Télécommun 69(1–2):47–62
5. ExMachina: programme de serious game 2025 ex machina, production tralalere (2010). http://www.2025exmachina.net/jeu. Accessed 22 Sept 2014
6. Cranor L, Langheinrich M, Marchiori M, Reagle J (2002) The platform for privacy preferences 1.0 (p3p1.0) specification. W3C recommendation. http://www.w3.org/TR/P3P/
7. Kelley P, Cesca L, Bresee J, Cranor L (2009) Standardizing privacy notices: an online study of the nutrition label approach. In: CHI'10 proceedings of the 28th international conference on human factors in computing system
8. Inglesant P, Sasse M, Chadwick D, Shi L (2008) Expressions of Expertness: the Virtuous Circle of Natural Language for Access Control Policy Specification. In: SOUPS'08: proceedings of the 4th symposium on usable privacy and security, ACM, New York, USA
9. Stepien B, Matwin S, Felty A (2011) Advantages of a non-technical xacml notation in role-based models. In: International conference on privacy, security and trust (PST) pp 193–200
10. CyanogenMod: privacy guard manager (2013). http://goo.gl/PU3mw. Accessed 22 Sept 2014
11. Sandhu RS, Coyne EJ, Feinstein HL, Youman CE (1996) Role-based access control models. Computer 29(2):38–47. doi:10.1109/2.485845
12. Byun JW, Bertino E, Li N (2005) Purpose based access control of complex data for privacy protection. In: Proceedings of the tenth ACM symposium on access control models and technologies, SACMAT '05, pp 102–110. ACM, New York, NY, USA. doi:10.1145/1063979.1063998
13. Jiang X, Landay J (2002) Modeling privacy control in context-aware systems. IEEE Pervasive Comput 1(3):59–63. doi:10.1109/MPRV.2002.1037723
14. Wagealla W, Terzis S, English C (2003) Trust-based model for privacy control in context aware systems. In: Second workshop on security in ubiquitous computing at the fifth annual conference on ubiquitous computing (UbiComp2003)
15. Ajam N, Cuppens-Boulahia N, Cuppens F (2010) Contextual privacy management in extended role based access control model. In: Garcia-Alfaro J, Navarro-Arribas G, Cuppens-Boulahia N, Roudier Y (eds) Data privacy management and autonomous spontaneous security, vol 5939., Lecture notes in computer scienceSpringer, Berlin, pp 121–135
16. Graf C, Hochleitner C et al (2011) Towards usable privacy enhancing technologies: lessons learned from the PrimeLife project. http://primelife.ercim.eu/results/documents/149-416d. Accessed 31 Jan 2015
17. Oglaza A, Laborde R, Zaraté P (2013) Authorization policies: using decision support system for context-aware protection of user's private data. In: Proceedings of the 12th IEEE International conference on trust, security and privacy in computing and communications (IEEE Ubisafe-13)
18. Gorry GA, Morton MSS (1971) A framework for management information systems. Sloan Manage Rev (Massachusetts Institute of Technology) 13:21–36
19. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749. doi:10.1109/TKDE.2005.99
20. Ruiz EV (1986) An algorithm for finding nearest neighbours in (approximately) constant average time. Pattern Recogn Lett 4(3):145–157. doi:10.1016/0167-8655(86)90013-9. http://www.sciencedirect.com/science/article/pii/016786558690
21. Martin A, Zaraté P, Camilleri G (2012) Gestion et évolution de profils multicritères de décideur par apprentissage pour l'aide à la décision. In: INFORSID: INFormatique des ORganisation et Systémes d'Information et de Décision, Montpellier, France, pp 223–238
22. McSherry F, Mironov I (2009) Differentially private recommender systems: building privacy into the net. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 627–636. ACM
23. Shyong K, Frankowski D, Riedl J (2006) o you trust your recommendations? An exploration of security and privacy issues in recommender systems. In: Müller G (ed) Emerging trends in information and communication security. Springer, Berlin, pp 14–29

24. Calandrino JA, Kilzer A, Narayanan A, Felten EW, Shmatikov V (2011) You might also like: privacy risks of collaborative filtering. In: IEEE symposium on security and privacy (SP), 2011, pp 231–246. IEEE
25. Simon HA (1972) Theories of bounded rationality. Decis Organiz 1:161–176
26. OASIS: OASIS XACML committee extensible access control markup language (XACML) version 2 (2005). http://www.oasis-open.org/committees/xacml/. Accessed 22 Sept 2014
27. Giard VE, Roy B (1985) Méthodologie multicritère d'aide à la décision. Economica, France
28. Bouyssou D, Dubois D, Pirlot M, Prade H (eds) (2006) Concepts et méthodes pour l'aide à la décision—analyse multicritère. Traité IC2, vol 3. Hérmes-Lavoisier, Paris. http://www.editions-hermes.fr/
29. Grabisch M, Roubens M (2000) Application of the choquet integral in multicriteria decision making. Fuzzy Measures Integrals 40:348–375
30. Grabisch M, Kojadinovic I, Nantes SP, Meyer P (2006) Using the kappalab r package for capacity identification in choquet integral based maut. In: Proceedings of the 11th international conference on information processing and management of uncertainty in knowledge-based systems, pp 1702–1709
31. Oglaza A (2014) Système d'aide à la décision pour la protection des données de vie privée. Thèse de doctorat. Université de Toulouse, Toulouse, France



Arnaud Oglaza received a Ph.D. degree in Computer Science from University of Toulouse in 2014. He is currently working as R&D Engineer at Institut de Recherche en Informatique de Toulouse. His research interests includes privacy, decision support and security.

Pascale Zarate is a Professor at Toulouse 1 Capitole University. She conducts her researches at the IRIT laboratory (http://www.irit.fr). She holds a Ph.D. in Computer Sciences from the LAMSADE laboratory at the Paris Dauphine University, Paris (1991). She is the editor in chief of the International Journal of Decision Support Systems Technologies (Ed IGI Global). Since 2000, she is head of the Euro Working Group on DSS (http://www.euro-online.org). She belongs the Editorial Scientific Committee of six International Journals. She published several studies and works: 3 books, 13 special issues, 27 papers, 35 papers in international journal conferences.

Romain Laborde is an Associate Professor at University of Toulouse (Paul Sabatier-IUT 'A'), France since 2006. He is also member of the Institut de Recherche en Informatique de Toulouse. He received his PhD in Computer Science from University Paul Sabatier in 2005. Then, he was a Research Associate in the Information Systems Security Group in the Computer Science Department, University of Kent at Canterbury, UK. His research focuses on security management applied to network security configuration, identity and access management or privacy.