KAPUER : un assistant à l'écriture de politiques d'autorisation pour la protection de la vie privée

Arnaud Oglaza, Romain Laborde, Pascale Zaraté

KAPUER : un assistant à l'écriture de politiques d'autorisation pour la protection de la vie privée

Arnaud Oglaza, Romain Laborde, Pascale Zaraté

IRIT, Université de Toulouse
118 Route de Narbonne, F-31062 Toulouse Cedex 9, France
Arnaud.Oglaza@irit.fr,Romain.Laborde@irit.fr,Pascale.Zarate@irit.fr

RÉSUMÉ. *Nous utilisons de plus en plus d'équipements informatique connectés à Internet. Nos téléphones, nos tablettes, et maintenant les équipements de notre quotidien peuvent désormais partager des informations pour faciliter notre vie. Partager ces données peut porter préjudice à notre vie privée et il est nécessaire de les contrôler. Cependant, cette tâche est complexe surtout pour des utilisateurs novices. Pour les y aider, nous présentons un système d'aide à la décision appelé KAPUER dont l'objectif est d'apprendre les préférences en termes de protection de la vie privée et de proposer des règles adaptées pour le contrôle de l'accès aux données. Ce système est intégré dans l'architecture de gestion d'autorisation XACML et trois algorithmes d'apprentissages sont évalués.*

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.*

MOTS-CLÉS : *protection de la vie privée, système d'aide à la décision, ecriture de politiques d'autorisation, XACML.*

KEYWORDS: *privacy protection, decision support system, authorization policy, XACML.*

1. Introduction

Aujourd'hui l'informatique personnelle qui consistait dans les foyers à un unique ordinateur personnel relié au réseau Internet via une connexion filaire par foyer tend à disparaître. Une étude réalisée par GFK/Médiamétrie publiée en nombre 2013 (GFK, 2013) a montré que le nombre de foyers « multi-équipés » (ordinateur portable + téléphone mobile + tablette) a plus que doublé en un an et atteint les 4,7 millions de ménages en France. Ajouté à cela, les smartphones et tablettes ont aujourd'hui des capacités de traitement et de stockage permettant d'exécuter de nombreuses applications. A titre d'exemple, les français ont en moyenne 32 applications installées sur leurs smartphones Android selon le recensement effectué par Google en 2013 [1]. Ce chiffre monte jusqu'à 40 applications dans des pays comme la Corée ou la Suisse. De plus, cette diversité d'équipements connectés à des réseaux informatiques va continuer à croître avec l'arrivée de l'internet des objets. Différentes études estiment que le monde compte déjà entre 15 et 20 milliards de « choses » connectées à l'Internet et que ce nombre devrait atteindre entre 50 et 80 milliards en 2020 (ZDNET, 2013), (EMC, 2014). Toutes ces choses connectées et toutes ces applications peuvent être amenées à traiter et communiquer des données relatives aux personnes. Par conséquent, toute personne va devoir contrôler ses propres choses et applications si elle désire protéger sa vie privée.

Aujourd'hui, il est donc nécessaire de fournir des outils permettant aux personnes de comprendre la problématique de la protection de la vie privée et d'appréhender la complexité de cette tâche. Différentes initiatives ont vu le jour dans cette perspective (Chabridon *et al*., 2014). Des travaux ont proposé d'aider les personnes à comprendre les risques liés à la divulgation de données au travers de jeux sérieux tels que 2025 ex-machina (ExMachina, 2010). Le projet Platform for Privacy Preferences (Cranor *et al*., 2002) a défini un standard pour uniformiser les politiques en matière de vie privée des sites web afin de permettre aux personnes de comprendre comment les sites web traitent leurs données. Ces politiques sont ensuite évaluées par rapport à des préférences utilisateurs par un mécanisme *ad-hoc*. Le même objectif est poursuivit par Kelley (Kelley *et al*., 2009). Ils ont remarqué que les gens comprennent les signalétiques nutritionnels que l'on retrouve sur les emballages d'aliments. Ils ont donc proposé une solution similaire pour représenter les politiques de vie privée. Inglesant (Inglesant *et al*., 2008) ont présenté un langage naturel contraint pour faciliter la compréhension de politique d'autorisation. Stiepen (Stepien *et al*., 2011) travaillent sur une notation non technique pour faciliter la compréhension de politique d'autorisation XACML. Ces recherches sont nécessaires pour aider les personnes à comprendre les risques qu'elles encourent et à rendre des documents techniques tels que des politiques en matière de vie privée ou encore d'autorisation compréhensibles par le commun des mortels. Cependant, il n'existe que peu de travaux qui aident les personnes à concevoir et écrire des politiques d'autorisation pour protéger leur vie privée.

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

Une première approche pour aider à la conception et à l'édition de politique d'autorisation consiste à proposer une interface graphique pour accéder aux différents droits. L'exemple type est le composant *privacy guard* de la distribution Android cyanogen-mod (CyanogenMod, 2013). Cette interface offre un tableau de bord où sont centralisées les informations relatives à la gestion des droits des applications installées sur le système Android. Il est possible de définir pour chaque application ses droits exacts. Les avantages de cette approche sont 1) l'interface graphique de gestion qui ne nécessite pas de connaissance spécifique à la définition de politique d'autorisation ainsi que 2) la possibilité de gérer finement les droits des applications. Cependant, cette approche ne peut supporter le passage à l'échelle. En effet, elle ne permet d'exprimer que des politiques d'autorisation de bas niveau. Pour quantifier le problème, nous avons analyser le nombre de permissions moyen d'une application android. Sachant qu'un smartphone comporte en moyenne 32 applications et que chaque application demande en moyenne 11,4 permissions dont 5,72 ont impact fort sur la vie privées (nous avons obtenu ces valeurs en analysant les permissions des 50 applications les plus téléchargées sur Android), un utilisateur doit en moyenne gérer sur son smartphone 364 permissions dont 183 ayant un impact sur sa vie privée.

Le problème de passage à l'échelle a été traité de manière importante dans les travaux de recherche liés aux modèles de politiques de contrôle d'accès. En effet, les administrateurs ont déjà été confrontés au même problème. Le modèle RBAC (Sandhu *et al.*, 1996) par exemple facilite la gestion des permissions en les regroupant par rapport aux rôles que peuvent jouer les utilisateurs dans une organisation. L'abstraction amenée par la notion de rôle limite le nombre de règles. Différents modèles de politique de contrôle d'accès ont proposé des éléments clés à considérer dans le contexte de la gestion de la vie privée ainsi que des abstractions facilitant leur manipulation telles que la finalité d'utilisation d'une ressource (Byun *et al.*, 2005), la sensibilité d'une ressource (Jiang, Landay, 2002), la confiance (Wagealla *et al.*, 2003), ou encore la précision ou le consentement (Ajam *et al.*, 2010). Ces modèles de politiques de contrôle d'accès offrent la possibilité d'écrire des règles de haut niveau plus adaptées à la gestion d'environnement complexe. Cependant, cette approche nécessite la compréhension des abstractions proposées par les modèles de politiques pour pouvoir les utiliser correctement. Ainsi, une phase de conception est nécessaire avant d'écrire des règles de contrôle d'accès. Imposer ces contraintes à des utilisateurs novices est difficilement envisageable. De plus, créer une interface utilisateur générique permettant d'écrire facilement des politiques manipulant ces concepts abstraits est une tâche très complexe (Graf *et al.*, 2011). Comment éviter l'approche mode novice (simple mais limité) *vs* mode expert (complet mais complexe) ?

Partant de ce constat, nous présentons une nouvelle approche complémentaire qui permet à un utilisateur novice d'écrire des politiques de haut niveau tout en limitant la charge cognitive nécessaire leur écriture (*i.e.* phase de conception et interface de spécification). Notre proposition est un système appelé KAPUER (pour *Kapuer is an Assistant for the Protection of UsErs' infoRmation*) qui utilise les techniques d'aide à la décision pour assister les utilisateurs dans l'écriture de règles d'autorisation

abstraites. KAPUER analyse les permissions de bas niveau accordées par un utilisateur pour apprendre ses préférences en termes de protection de la vie privée et lui propose des politiques de haut niveau correspondant à ces préférences. KAPUER ne prend pas de décision à la place de l'utilisateur, il l'aide par ses propositions. Ainsi, l'utilisateur peut alors accepter la règle proposée qui sera mise en oeuvre par le système d'autorisation, la refuser s'il considère que cette règle n'est pas correcte ou alors la modifier. Nous avons intégré KAPUER dans l'architecture système de gestion des autorisations XACML qui offre une flexibilité importante tant en termes d'intégration dans un système cible qu'en termes de spécification de modèles de politique de contrôle d'accès. Ce travail étend la proposition faite dans (Oglaza *et al*., 2013) qui ne portait que sur une première approche d'intégration d'un système d'aide à la décision pour la protection de la vie privée dans l'environnement Android.

La suite de l'article est structurée ainsi. Dans la section 2, nous introduisons les systèmes d'aide à la décision. Dans la section 3, nous définissons notre modélisation des critères de préférences pour la protection de la vie privée. Dans la section 4, nous présentons l'intégration de KAPUER dans XACML. Nous concluons et discutons des améliorations à apporter dans la section 5.

2. Introduction aux systèmes d'aide à la décision

Les systèmes d'aide à la décision ont été introduits dans les années 1970. Gorry et Morton ont été les premiers à employer le terme en 1971 (Gorry, Morton, 1971). Cette approche combine des modèles mathématiques pour analyser le comportement des preneurs de décisions et des ordinateurs pour leur interactivité et les techniques de visualisation disponibles. Avec la puissance actuelle des machines et des logiciels, les systèmes d'aide à la décision peuvent aider des utilisateurs à prendre des décisions de plus en plus complexes impliquant beaucoup d'informations à considérer.

Un système d'aide à la décision (*Decision Support System* - DSS) peut aider les utilisateurs à gérer les prises de décisions complexes en recréant un processus de prise de décision. Afin d'aider le décideur, le système doit avant tout le comprendre. Pour cela, le système et le décideur interagissent l'un avec l'autre. Grâce à ces interactions, le système est capable d'analyser une décision prise par un décideur grâce à des algorithmes d'apprentissage. Le but principal d'un système d'aide à la décision est d'aider le décideur à prendre des décisions ; mais en aucun cas il ne doit prendre la décision pour lui. Le système est là pour prêter assistance à l'utilisateur et non pas le remplacer.

Il existe plusieurs façons d'aider un utilisateur. Cela peut être en lui expliquant le problème qu'il rencontre, en lui donnant les causes qui ont amené ce problème ou encore en décomposant un problème complexe en plusieurs sous-problèmes plus simples à appréhender. Par exemple, il existe des tableaux de bord aidant les utilisateurs en agrégeant plusieurs paramètres pour aider à la prise de décisions sur les marchés financiers. L'utilisateur est guidé à travers divers indicateurs lui permettant de comprendre les différentes informations disponibles afin de prendre une

décision. Une autre manière d'aider le décideur est de proposer différentes solutions à son problème. Ce genre de système est appelé un système de recommandation. Amazon, par exemple, utilise un système de ce genre pour proposer à ses clients une liste d'objets potentiellement intéressants en se basant sur leurs anciens achats, leur historique de navigation mais aussi les achats effectués par les autres clients.

Plusieurs aspects doivent être pris en compte pour construire un système de recommandation efficace. Il doit présenter des solutions qui doivent être aussi satisfaisantes que possibles, et ce le plus rapidement possible. Un système où l'utilisateur doit attendre trop longtemps avant d'avoir le choix entre des solutions intéressantes ne sera au final pas ou peu utilisé. Pour comprendre l'utilisateur et lui présenter les meilleures solutions possibles, le système doit interagir avec lui. Ces interactions sont indispensables mais doivent être minimisées. Un utilisateur dérangé trop souvent par le système le délaissera rapidement. La difficulté pour construire un bon système de recommandation est donc de trouver le bon nombre d'interactions nécessaires avant de trouver des solutions satisfaisantes.

Trois approches existent pour construire un système de recommandation (Adomavicius, Tuzhilin, 2005). La recommandation collaborative utilise les informations sur les autres utilisateurs pour trouver les recommandations à faire. Le gros avantage de cette approche est de pouvoir proposer rapidement des solutions à l'utilisateur même sans avoir d'informations sur ses préférences. Etant donné que le système utilise les préférences de tous les utilisateurs, les solutions proposées ne sont pas aussi précises qu'un système n'utilisant que les préférences de l'utilisateur. Les solutions sont trouvées en calculant les similarités et les différences entre les utilisateurs avec par exemple la technique des k plus proches voisins (Ruiz, 1986).

La deuxième approche est basée sur le contenu et ne prend en compte que les caractéristiques de chaque objet. Par exemple, un film peut être décrit par son titre, sa date de sortie, son réalisateur et ses acteurs. Chaque film peut être décrit par ses caractéristiques et connaître les préférences de l'utilisateur quant à ces caractéristiques peut permettre de faire des recommandations (Martin *et al*., 2012).

La dernière approche est une combinaison des deux premières. Cette approche hybride utilise à la fois la recommandation collaborative pour trouver les habitudes des autres utilisateurs et la recommandation basée sur le contenu pour trouver les objets avec des caractéristiques appréciées par l'utilisateur.

Nous pensons que l'approche basée sur le contenu est plus appropriée pour notre système que les deux autres car le contrôle de la vie privée est quelque chose de très personnel, ainsi deux utilisateurs avec des préférences proches sur certains critères n'auront pas forcément le même comportement sur d'autres critères. Les propositions ne seraient donc pas forcément pertinentes pour l'utilisateur. De plus, garder les préférences localement était un souhait fort de notre part. De nombreux travaux soulignent les problèmes de confidentialité intervenant suite à l'utilisation de système de recommandation collaboratif (McSherry, Mironov, 2009 ; Shyong *et al*., 2006). Dans (Calandrino *et al*., 2011), les auteurs expliquent par exemple qu'il est possible,

en utilisant les informations acquises par les actions d'un utilisateur, d'inférer d'autres informations sur son comportement. L'approche basée sur le contenu permet d'éviter ces problèmes de confidentialité étant donné que les préférences de l'utilisateur ne sont pas partagées.

Dans un système de recommandation, il est nécessaire de pouvoir évaluer les objets à choisir sur plusieurs critères. En effet, une approche mono-critère n'est pas envisageable car les préférences des utilisateurs sont presque toujours basées sur plusieurs critères. Un système d'aide à la décision a des objectifs concrets. Il n'y a pas d'hypothèses fortes faites sur l'environnement. Simon a proposé pour l'aide à la décision le principe de rationalité limitée (Simon, 1972). Les décisions rationnelles sont souvent impossibles à prendre car les critères utilisés pour prendre une décision peuvent être contradictoires. Par exemple, une personne veut effectuer un trajet entre Toulouse et Paris et veut aller le plus vite possible tout mais aussi polluer le moins possible. Hors si l'avion est le moyen de transport le plus rapide, il est aussi un des plus polluants (600 à 900km/h pour 145g/co^2/km). Le train quand à lui est le moyen de transport le plus écologique mais n'est pas aussi rapide (90 à 250km/h pour 13g/co^2/km) [2]. Il n'existe donc pas de solution optimale alliant vitesse maximale et pollution minimale. L'utilisateur devra donc choisir la solution la plus satisfaisante et mettre une priorité sur chaque critère.

Un système de recommandation suit un processus cyclique (cf figure 1). Ce cycle commence par une action de l'utilisateur qui peut être un classement de plusieurs objets, donner une note à un ou plusieurs objets ou directement choisir un objet dans une liste. Le système va décomposer l'objet concerné par l'action de l'utilisateur en une liste de critères. Chaque critère est associé à une valeur dépendante de l'action de l'utilisateur. Cette étape est effectuée en utilisant un opérateur de décomposition et une méthode appelée analyse multicritère. Ensuite, cette liste de critères est mise en correspondance avec les préférences actuelles de l'utilisateur. L'ensemble des préférences de l'utilisateur définit le profil utilisateur et est utilisé pour donner un poids à chaque critère de la liste de critères. Le système est alors capable de calculer le score de l'objet en agrégeant tous les critères. Un opérateur d'agrégation est utilisé pour obtenir ce score. Vient ensuite la dernière étape de la phase d'apprentissage où le système capitalise l'information acquise de l'action de l'utilisateur en mettant à jour le score de tous les critères impliqués en utilisant le score agrégé.

Une fois que les préférences ont été mises à jour, le système peut, avec la même méthode que pendant l'apprentissage, calculer le score de tous les objets. Il décompose les critères et les agrège avec les préférences de l'utilisateur. Une fois que les scores de tous les objets sont disponibles, le système peut les proposer à l'utilisateur et le processus peut recommencer afin d'affiner les recommandations faites à l'utilisateur.

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

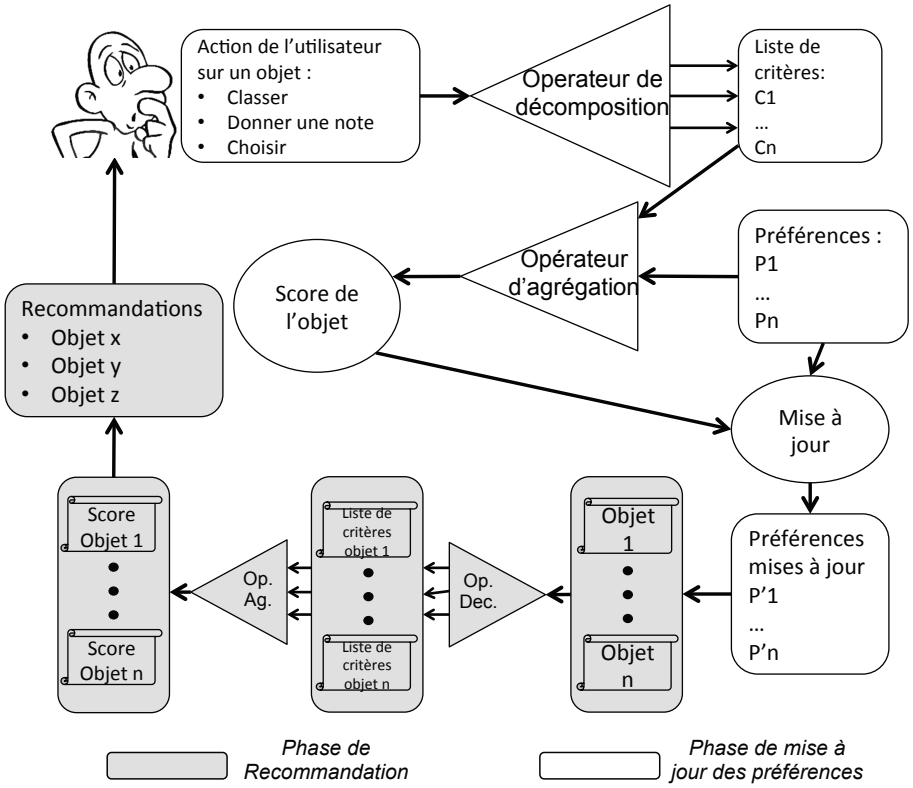

Figure 1. Processus d'un système de recommandation

3. Modèle de résolution de problèmes pour la protection de la vie privée

Nous présentons dans cette section le modèle de résolution de problèmes que nous utilisons pour faire dialoguer ensemble le système d'autorisation et le système d'aide à la décision.

3.1. Les critères

Un critère représente l'élément de base constituant une requête d'accès à une ressource protégée. Le critère peut correspondre au nom de l'utilisateur, son âge, le nom de la ressource, le nom de l'action, etc. Ainsi, *« Jacqueline veut lire le calendrier »* comporte trois critères *Jacqueline*, *lire* et *calendrier*. L'ensemble des critères du système est noté CR. Un critère est composé d'un identifiant et de deux valeurs correspondant aux préférences de l'utilisateur. En effet, si les préférences

avaient été modélisées avec une seule valeur par critère, il aurait été difficile d'interpréter un critère ayant une valeur faible. Cette faible valeur voudrait-elle dire que pour l'utilisateur, ce critère n'est pas un critère favorable à la divulgation ou bien que le système n'a pas encore eu le temps d'apprendre les préférences de l'utilisateur concernant ce critère ? Il n'est pas possible de répondre à cette question. Pour résoudre ce problème, nous utilisons deux valeurs pour chaque critère :

– La première, $g^t : CR \rightarrow [0, \infty[$ représente la préférence de l'utilisateur pour la divulgation par rapport à ce critère à l'instant t.

– La seconde $f^t : CR \rightarrow [0, \infty[$ représente la préférence de l'utilisateur pour la non-divulgation par rapport à ce critère à l'instant t.

Chaque information et critère doivent être notés selon les préférences des utilisateurs. Cette étape constitue le calcul de score. Nous avons choisi d'utiliser une méthode de calcul de score incrémentale uniquement, donc, lors de leur mise à jour, $g^t(x)$ et $f^t(x)$ ne peuvent qu'augmenter. L'apprentissage des préférences se faisant en continu, une valeur élevée de $g^t(x)$ ne veut pas forcément dire que lorsque le critère x est présent, l'utilisateur est fortement enclin à divulguer ses données. Cela peut aussi vouloir dire que ce critère est souvent apparu parmi les requêtes et qu'il a été mis à jour de multiples fois. Pour identifier le score d'un critère, il faut calculer soit :

– $s^t_D(x)$ correspondant au score du critère x à l'instant t en faveur de la divulgation. Ce score est issu de la différence entre la valeur de divulgation et la valeur de non-divulgation :

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

– $s^t_{nD}(x)$ correspondant au score du critère x à l'instant t contre la divulgation. Ce score est issu de la différence entre la valeur de non-divulgation et la valeur de divulgation :

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

Calculés ainsi, les scores $s^t_D(x)$ et $s^t_{nD}(x)$ établissent la position du critère x dans les préférences de divulgation ou non de l'utilisateur. Un faible score de $s^t_D(x)$ ou $s^t_{nD}(x)$ reflète une absence claire de raisons justifiant la préférence pour l'une des deux actions. Au contraire, un score élevé de $s^t_D(x)$ ou $s^t_{nD}(x)$ correspond à l'existence de raisons claires confirmant une préférence stricte en faveur d'une des deux actions.

3.2. Les classes de critères

Les modèles de politique de contrôle d'accès proposent des éléments clés à prendre en compte tels que :

– la visibilité : qui veut avoir accès à la ressource (un ami, un collègue, un inconnu, etc.)

– l'aspect temporel : le moment de la protection (différents jours de la semaine, différentes heures de la journée, etc.)

– l'aspect spatial : le lieu où se trouve l'utilisateur (chez lui, au travail, etc.)

– la rétention : comment la ressource est stockée par la suite (combien de temps, qui y aura accès, etc.)

– l'intention : comment sera utilisé la donnée par la suite (à but philanthropique, pour être revendue, etc.),

Pour exprimer ces éléments, nous introduisons la notion de classe. Chaque critère fait partie d'une classe de critères par la relation $ACC \subseteq CR * C$ où l'ensemble des classes de critères est noté C. Le système étant générique, les classes de critères ne sont pas fixées et n'importe quelle classe de critères peut être créée. Pour simplifier notre notation par la suite, nous définissons la fonction $class$ qui renvoie l'ensemble des critères appartenant à une même classe :

$$class : C \rightarrow \mathcal{P}^{CR}$$

$$x \mapsto \{y \in CR | (y, C) \in ACC\} \tag{3}$$

3.3. Les méta-critères

Nous définissons la notion de *méta-critère* pour représenter les abstractions des modèles de politique de contrôle d'accès comme le rôle de RBAC (Sandhu *et al.*, 1996), les vues/activités de OrBAC (Ajam *et al.*, 2010), les hiérarchies d'intentions de PRBAC (Byun *et al.*, 2005), etc. Le préfixe méta, dans le vocabulaire scientifique, permet entre autre de désigner un niveau d'abstraction supérieur. Ici, le niveau 0 correspond à un critère, les niveaux strictement supérieurs à 0 correspondent à des méta-critères. Un méta-critère est un critère ayant un niveau d'abstraction supérieur à un ou plusieurs critères appartenant à la même classe. L'ensemble des méta-critères du système est noté MCR, $MCR \subset CR$. Les deux ensembles ne peuvent pas être égaux car nous considérons que le système doit contenir au moins un critère de niveau d'abstraction 0 et donc n'appartenant pas à MCR. Un méta-critère permet de regrouper plusieurs critères partageant une caractéristique commune. Par exemple, considérons les critères « Jacqueline » et « Bernard ». Ces deux critères ont une caractéristique commune : être des parents. On peut ainsi définir le méta-critère « Parent » regroupant les critères « Jacqueline » et « Bernard ». De même, le méta-critère « Famille » est un niveau d'abstraction supérieur à « Parent ». Les valeurs $f^t(x)$ et $g^t(x)$ du méta-critère x lui sont propres.

Un méta-critère étant aussi un critère, il est possible pour chaque classe de critères cl de créer une hiérarchie $H_{cl} \subseteq CR * MCR$ entre ces critères tels que $\forall (c_1, c_2) \in$

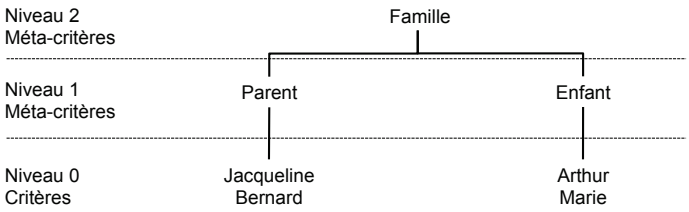

Figure 2. Hiérarchie de critères

$H_{cl}, class(c_1) = class(c_2)$ (cf. figure 2). Il existe deux cas où un critère ne possède pas dans sa description de méta-critère :

– Le critère est en haut de la hiérarchie (exemple le critère « Famille » dans la figure 2),

– Le critère est indépendant et ne peut donc pas être associé à un méta-critère.

3.4. Les groupes de critères

Afin d'analyser les relations inter-critères qui permettent de capter plus finement les préférences des utilisateurs, nous définissons le *groupe* de critères. Un groupe de critères est une association de n critères. Un groupe de critère a ses propres valeurs, indépendantes des valeurs des critères qui le composent. Les critères ou méta-critères composant un groupe de critères doivent appartenir à des classes de critères différentes. Ainsi prenons par exemple les critères « Parent » et « Calendrier » appartenant à deux classes différents, nous pouvons créer le groupe de critères {Père, Calendrier}. L'ensemble des groupes de critères G est défini par :

L'ensemble G est compris dans l'ensemble des parties de CR :

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

Un groupe de critère est composé au minimum de deux critères.

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

Deux critères d'un même groupe ne peuvent appartenir à la même classe de critère.

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

3.5. Exemple de formalisation

La figure 3 illustre par l'exemple les différentes notions autour du critère.

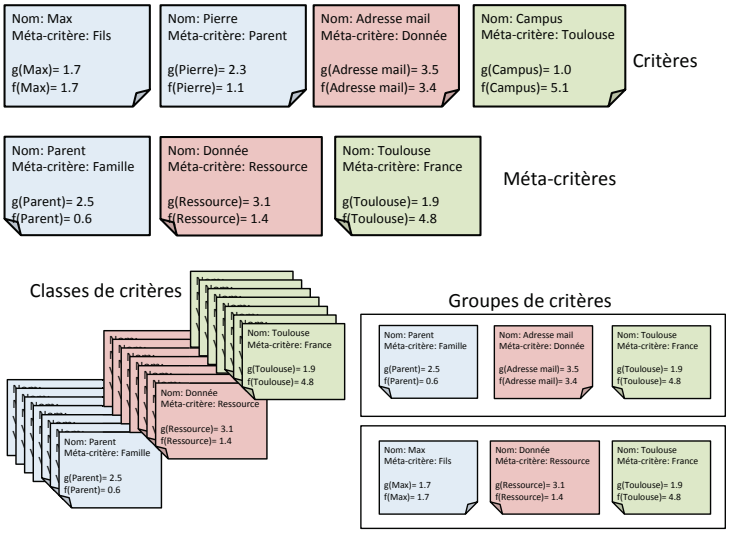

Figure 3. Illustration des notions autour du critère

En reprenant les exemples de la figure 3, nous obtenons :

L'ensemble des méta-critères MCR et l'ensemble des critères CR :

$$MCR = \{Fils, Parent, Famile, Donnée, Ressource, Toulouse, France\}.$$

$$CR = MCR \cup \{Max, Pierre, Adresse\ mail, Campus\}$$

La relation de hiérarchie entre un critère et un méta-critère H_{cl} :

$$H_{cl} = \{(Max, Fils), (Pierre, Parent), (Adresse\ mail, Donnée), (Campus, Toulouse), (Parent, Famille), (Donnée, Ressource), (Toulouse, France)\}.$$

L'ensemble des classes de critères C :

$$C = \{Qui, Quoi, Où\}.$$

La relation entre un critère et une classe de critère ACC :

$$ACC = \{(Max, Qui), (Pierre, Qui), (Adresse\ mail, Quoi), (Campus, Où), (Fils, Qui), (Parent, Qui), (Donnée, Quoi), (Toulouse, Où), (Famille, Qui), (Ressource, Quoi), (France, Où)\}.$$

L'ensemble des groupes de critères G :

G = {(Parent, Adresse mail, Toulouse), (Max, Donnée, Toulouse)}.

A l'instant t les valeurs de divulgation g et de non divulgation f de chaque critère :

$$g^t(Max) = 1.7 \text{ et } f^t(Max) = 1.7$$
$$g^t(Pierre) = 2.3 \text{ et } f^t(Pierre) = 1.1$$
$$g^t(Adresse\ mail) = 3.5 \text{ et } f^t(Adresse\ mail) = 3.4$$
$$g^t(Campus) = 1.0 \text{ et } f^t(Campus) = 5.1$$
$$g^t(Parent) = 2. \text{ et } f^t(Parent) = 0.6$$
$$g^t(Donne) = 3.1 \text{ et } f^t(Donne) = 1.4$$
$$g^t(Toulouse) = 1.9 \text{ et } f^t(Toulouse) = 4.8$$

4. Intégration de KAPUER dans XACML

Le standard XACML (OASIS, 2005) est une spécification XML définie par OASIS pour la définition de politiques de contrôle d'accès. XACML fournit un langage universel de description des politiques de contrôle d'accès de la forme : qui peut faire quoi et à quel moment ? La politique de contrôle d'accès permet de définir les droits des utilisateurs (personne ou application) sur les ressources informatiques (données, services, etc.). XACML est un langage d'expression puissant où toute information de sécurité peut être considérée comme un attribut du sujet, de la ressource, de l'action ou encore de l'environnement.

De plus, ce langage s'appuie sur une architecture de type *Policy Decision Point/Policy Enforcement Point* pour la mise en oeuvre du contrôle d'accès : un protocole de type requête/réponse donne les moyens d'exprimer des requêtes d'accès et les réponses appropriées. Les éléments grisés de la figure 4 reprennent les principaux composants définis dans XACML :

– le PDP (*Policy Decision Point*) est une entité logique qui prend des décisions d'autorisation. Les requêtes d'accès aux ressources protégées sont évaluées par rapport à des politiques écrites dans le langage XACML.

– le PEP (*Policy Enforcement Point*) est une entité logique qui applique la décision d'autorisation prise par le PDP. C'est le PEP, gardien de la ressource, qui réalise techniquement le contrôle d'accès. Il reçoit les requêtes d'accès, les traduit en XACML et les envoie au PDP. Il attend ensuite la réponse du PDP pour faire appliquer sa décision.

– une base de politiques, simple base de données où sont stockées les politiques XACML.

Le dernier composant (de couleur blanche) de cette architecture ne fait pas partie du standard XACML. Il correspond à notre proposition KAPUER pour la protection de la vie privée (*Decision Support System* - système d'aide à la décision). Les sections suivantes du chapitre décrivent les différentes étapes du système en suivant l'ordre de la numérotation de la figure.

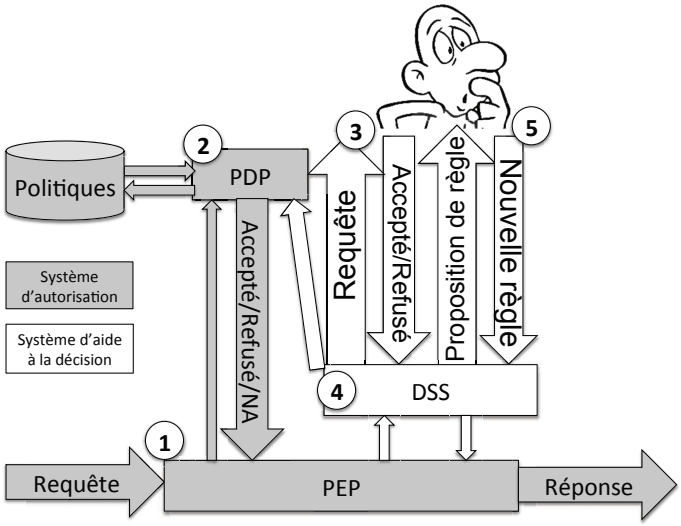

Figure 4. Architecture globale de Kapuer intégré dans XACML

4.1. Etape 1 : Capture et traduction d'une requête d'accès en XACML

Une demande de partage d'une ou plusieurs ressources de la part d'un utilisateur ou d'une application est appelée une requête. Les ressources peuvent soit directement concerner l'utilisateur indépendamment du périphérique utilisé (son nom, son adresse-mail, etc.), soit utiliser des services du périphérique pour récolter les informations (le module GPS donne les coordonnées de l'utilisateur, l'agenda son emploi du temps, etc.).

Lorsqu'une requête arrive, elle est interceptée par le PEP. Son rôle est de traduire la requête sous forme d'attributs pour l'envoyer au PDP, et d'appliquer la décision qui est prise concernant la requête. Le système est générique et il est possible d'utiliser et de transformer n'importe quelle information disponible dans la requête sous forme d'attributs. Tous les attributs ne sont pas nécessairement renseignés à chaque requête.

Les requêtes sont donc traduites selon les informations qu'elles contiennent ou que le PEP peut obtenir via d'autres bases d'informations (telles que le carnet d'adresses, etc). La traduction d'une requête d'accès est un fichier XML regroupant les attributs sous les balises sujet, ressource, action et environnement.

```
<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>
```

Figure 5. Exemple de requête XACML

La figure 5 montre un exemple de requête traduite par le PEP. Cette requête est découpée en quatre parties :

– le sujet, l'entité qui demande l'information. Ici c'est une application, l'attribut prend le nom de l'application comme valeur « fr.irit.siera.testing ».

– la ressource, l'information demandée. Ici la valeur de l'attribut correspond à la permission requise pour avoir accès à l'information. « android.permission.READ_SMS » est la permission requise pour avoir accès en lecture aux SMS de l'utilisateur.

– l'action qui correspond au type d'action que le demandeur veut effectuer. Ici le demandeur veut avoir un accès à la ressource d'où la valeur de l'attribut « access ».

– l'environnement qui correspond à toutes les autres informations de contexte disponibles au moment de la requête. Ici les seules informations disponibles sont le jour et l'heure de la demande, avec les valeurs d'attributs « 2 » et « 10 » qui correspondent au deuxième jour de la semaine et à la dixième heure (mardi entre 10h et 11h).

Cette requête est ensuite transmise au PDP qui va continuer le processus de contrôle d'accès.

4.2. Etape 2 : Analyse de la requête par rapport aux politiques de sa base

La deuxième étape du processus de contrôle d'accès intervient lorsque le PDP reçoit une requête XACML à analyser. Le rôle du PDP est de vérifier s'il existe une correspondance entre les politiques existantes dans la base de politiques et de donner la décision résultante. Une politique est un ensemble d'expressions logiques, appelées règles, dont les variables libres sont des identifiants d'attributs. Lorsque le PDP reçoit une requête XACML contenant un ensemble de couples <identifiant d'attribut, valeur>, il peut effectuer la substitution et ainsi évaluer certaines règles de cette politique.

Dans le cas où les attributs et valeurs des attributs de la requête correspondent à ceux d'une règle, la valeur de la règle de cette politique est retournée au PEP. Une règle peut prendre deux valeurs, « PERMIT » si la requête est acceptée, « DENY » si elle est refusée.

Dans le cas où les attributs et valeurs des attributs de la requête ne correspondent à aucune des règles présentes dans la politique, la décision « NOT APPLICABLE » est renvoyée au PEP. Le PDP n'a pas pu prendre de décision par rapport à la politique existante. Cette situation est problématique pour un système d'autorisation classique car il correspond à un état non défini. En théorie, le PEP n'ayant pas de réponse à appliquer doit contacter un autre PDP. En pratique, les développeurs de PEP mettent en oeuvre une décision « NOT APPLICABLE » comme une décision « DENY ».

4.3. Etapes 3 et 4 : Interactions avec l'utilisateur pour apprendre ses préférences

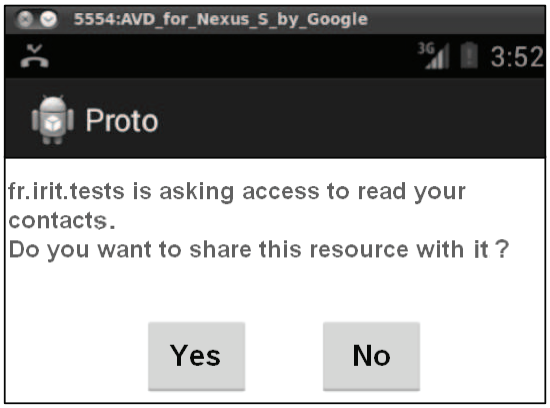

Figure 6. Interaction avec l'utilisateur

C'est à cette étape qu'intervient notre système d'aide à la décision (DSS pour Decision Support System). La décision « NOT APPLICABLE » indique qu'il

n'y a pas de règle d'autorisation correspondant à une requête donnée. Il serait envisageable de demander à l'utilisateur d'écrire via une interface graphique une règle d'autorisation. Cependant ce choix nécessiterait une phase de conception de la part de l'utilisateur pour écrire une règle utilisant des concepts abstraits : il devrait prendre une décision complexe.

Nous préférons lui demander de prendre une décision plus simple qui est limitée à cette requête précise. La figure 6 est une copie d'écran d'un exemple d'interaction avec l'utilisateur. Le système informe l'utilisateur qu'une entité (l'application « fr.irit.tests ») veut accéder à une ressource (sa liste de contacts) et lui demande s'il accepte de partager cette ressource. Nous utilisons cette interaction pour apprendre les préférences de l'utilisateur. Deux actions sont donc présentées à l'utilisateur, la divulgation D et la non divulgation nD. Une fois que l'utilisateur a répondu, le couple (requête, action) est envoyé au système d'aide à la décision qui l'analyse et utilise l'information pour faire évoluer les préférences de l'utilisateur. Les préférences de l'utilisateur sont une représentation, par un ensemble de critères, de la politique de confidentialité préférée par l'utilisateur. La notion d'attribut dans XACML étant très proche de la notion de critères telle que définie dans la section 3, il est possible de transformer une requête en une liste de critères.

Le DSS fait ainsi un apprentissage continu de ces préférences afin d'être au plus proche de la politique de confidentialité voulue par l'utilisateur. Ainsi dans notre cas, une décision « NOT APPLICABLE » indique que le DSS doit parfaire l'apprentissage des préférences de l'utilisateur et pour cela, il informe l'utilisateur de la requête en cours et lui demande de prendre une décision par rapport à la divulgation ou non de la donnée de vie privée concernée. Cette interaction permet de mettre à jour les valeurs des critères de la requête ainsi que leurs méta-critères et donc d'affiner les préférences de l'utilisateur. Lorsque le DSS aura une connaissance suffisante des préférences de l'utilisateur, il pourra lui proposer l'ajout d'une règle d'autorisation (section 4.4).

4.4. Etape 5 : Propositions de règles abstraites à l'utilisateur

L'objectif principal de KAPUER est de faire des propositions de règles d'autorisation abstraites à l'utilisateur. Cependant, il ne doit pas proposer n'importe quelle règle. Tout d'abord, la règle doit être de haut niveau afin qu'elle couvre un nombre de requêtes d'accès important. Cela permet d'éviter un grand important de règles bas niveau difficilement gérables par l'utilisateur et donc retrouver les mêmes difficultés qu'un système comme cyanogen-mod (voir section 1). De plus, définir rapidement des règles de haut niveau limite le nombre d'interactions avec l'utilisateur. Le deuxième point est de proposer des règles qui conviennent à l'utilisateur. Si le système propose des règles « hors sujet », l'utilisateur sera enclin à ne plus utiliser le DSS.

Faire une proposition à l'utilisateur revient à déterminer que l'action associée à cette proposition est strictement préférée à son contraire. Autrement dit, soit l'utilisateur préfère autoriser la divulgation, soit il préfère la refuser. Si le système

n'est pas en mesure de faire une proposition, alors il n'y a pas de préférence entre les deux actions. Deux situations peuvent entraîner cette non-préférence :

– la première lorsque le système n'a pas une représentation précise du comportement de l'utilisateur, donc quand il manque d'information.

– la deuxième lorsque l'utilisateur n'a pas un comportement fixe et n'agit pas de la même façon à chaque fois. Dans ce cas, le système ne peut pas inférer le comportement de l'utilisateur ni lui proposer une règle.

Afin de gérer les préférences, nous utilisons un système relationnel parfait de préférences (Giard, Roy, 1985). Il est constitué des deux relations binaires transitives suivantes :

– l'indifférence $\sim$ ou non-préférence qui correspond à une absence de raisons qui justifieraient une préférence en faveur d'une action ou de l'autre :

$$\sim\text{: } a \sim a' \Leftrightarrow aIa' \tag{4}$$

I étant une relation symétrique réflexive.

– la préférence stricte $\succ$ qui correspond à l'existence de raisons justifiant la préférence en faveur d'une des deux actions :

$$\succ\text{: } a \succ a' \Leftrightarrow aPa' \tag{5}$$

P étant une relation asymétrique irréflexive.

Une proposition de règle d'autorisation est construite en analysant la requête en cours et la décision prise par l'utilisateur lors de l'interaction. Pour faire l'analyse de ce couple, nous utilisons une méthode appelée analyse multicritère (Bouyssou *et al.*, 2006) (figure 7).

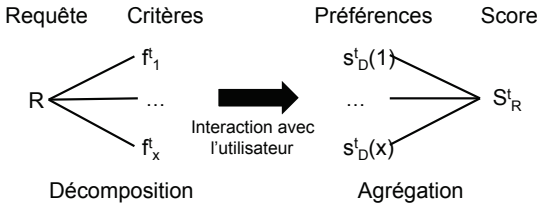

Figure 7. Méthode de l'analyse multicritère

Chaque requête est décomposée en critères qui sont ensuite agrégés grâce à un opérateur d'agrégation. Les préférences de l'utilisateur sont utilisées pendant cette étape pour pondérer les critères. Le résultat de l'agrégation fournira un score S^t_R de

la requête R, évaluant le degré de connaissance des préférences de l'utilisateur face à cette requête. Avec ce score et la décision de l'utilisateur, le système peut mettre à jour les valeurs des critères de R et de leurs méta-critères.

Afin de savoir si une proposition correspond à une préférence stricte, le système doit calculer le score S_R^{t+1} de la requête avec les valeurs $f^{t+1}(x)$ et $g^{t+1}(x)$ des critères et méta-critères mis à jour. Ce nouveau score est ensuite comparé à un paramètre λ, correspondant à la valeur seuil entre la relation d'indifférence et de préférence stricte. Si S_R^{t+1} est inférieur à λ, nous nous trouvons dans une situation d'indifférence et aucune proposition ne sera faite à l'utilisateur. Si S_R^{t+1} est supérieur à λ, nous nous trouvons dans une situation de préférence stricte et la proposition peut être présentée à l'utilisateur. λ est un paramètre qui influe sur la vitesse de proposition à l'utilisateur. Plus il est faible, plus le système fait des propositions à l'utilisateur rapidement. Inversement, plus il est élevé, plus le système est lent à faire des propositions à l'utilisateur. La valeur de ce paramètre a été affinée au moyen de simulations (cf section 4.6). Nous travaillons sur un réglage dynamique de λ qui s'adaptera au comportement de l'utilisateur.

Dans le cas où S_R^{t+1} est supérieur à λ, le système propose une nouvelle règle à l'utilisateur. Cela correspond à une nouvelle interaction avec lui. Durant cette interaction, le système propose à l'utilisateur d'insérer dans la base de politiques une nouvelle règle, dont les attributs correspondent aux critères ou méta-critères de la proposition. La décision de cette règle est aussi communiquée à l'utilisateur. Le choix est donné à l'utilisateur d'accepter cette règle ou de la refuser.

Bien que les attributs présentés à l'utilisateur soient censés être pertinents pour lui, celui-ci peut malgré tout modifier chaque attribut de la proposition. Soit l'attribut représente un critère et l'utilisateur peut choisir un méta-critère pour obtenir une règle plus agrégée. Soit l'attribut représente un méta-critère et l'utilisateur peut choisir le critère de la requête s'il ne veut pas que la règle soit agrégée avec ce méta-critère. La figure 8 présente un exemple de proposition de règle à l'utilisateur. Ici la règle présentée à l'utilisateur lui propose d'accepter de partager sa liste de contacts tous les jeudis avec l'application « fr.irit.tests ». Le système propose ici une règle dont l'abstraction a pu être déterminée sur la ressource et l'aspect temporel. L'utilisateur peut modifier les attributs pour que la règle soit étendue par exemple à toutes les applications ou à tous les jours de la semaine. Ainsi, l'utilisateur peut construire une règle lui convenant, sans avoir besoin de la spécifier et de l'écrire.

Si l'utilisateur accepte cette règle, le système d'aide à la décision l'ajoute dans la base de politiques du système de contrôle d'accès avant de communiquer la décision au PEP. S'il refuse, seule la décision est transmise au PEP. Une fois que le PEP reçoit la décision correspondant à la requête, elle est traduite pour être comprise par l'entité ayant fait la demande puis envoyée.

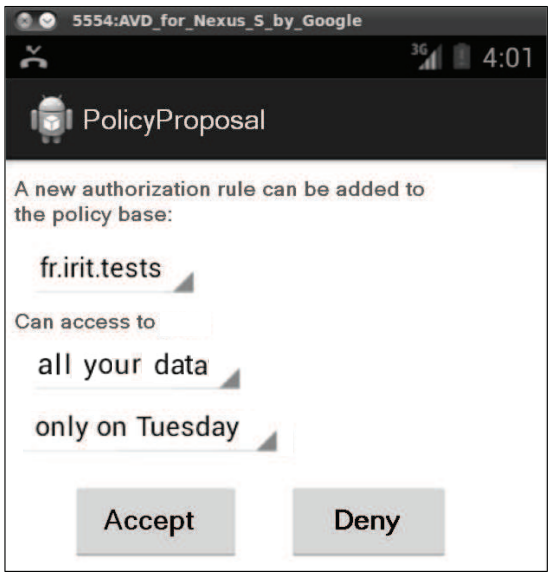

Figure 8. Exemple de proposition de règle

4.5. L'initialisation du système

Avant le début de l'exécution de Kapuer, le système peut être initialisé afin de diminuer le temps nécessaire pour proposer des règles. L'initialisation des préférences sert à donner des renseignements au système qui lui permettra de s'adapter selon les résultats obtenus. Ainsi, en posant quelques questions à l'utilisateur, il est possible de savoir si son comportement à tendance à être simple (divulguer toutes les ressources sans condition ou au contraire ne rien divulguer) ou bien si il va dépendre des différentes situations. Dans le premier cas, Kapuer pourra adapter la vitesse d'apprentissage à la hausse afin de proposer des règles plus rapidement à l'utilisateur et de favoriser l'apprentissage sur les méta-critères pour proposer des règles de haut niveau. Au contraire, si l'utilisateur a un comportement complexe, il est nécessaire, pour avoir un apprentissage plus fin, de baisser la vitesse d'apprentissage et de ne pas proposer de règles trop abstraites pour un apprentissage homogène sur les critères et les méta-critères.

Nous avons commencé à travailler sur cette initialisation en préparant un ensemble de questions (42 au total) et en les proposant à trois groupes d'utilisateurs ayant chacun une vision différente de la protection de la vie privée. Le premier groupe était composé d'étudiants en troisième année de droit et informatique. Le deuxième groupe était composé de doctorants et d'enseignants chercheurs en informatique. Le troisième

groupe était composé d'utilisateurs de smartphones non informaticiens et n'ayant aucune notion technique de la protection de la vie privée. Les questions mettaient en situation les utilisateurs et la réponse portait sur la décision qu'ils auraient prise dans cette situation. Mais les résultats ont montré que ces questions ne permettaient pas d'obtenir suffisamment d'informations pertinentes sur l'utilisateur malgré leurs nombres important. Nous sommes actuellement en train de travailler sur une nouvelle version de cette phase d'initialisation.

4.6. Apprentissage des préférences relatives à la vie privée

L'apprentissage des préférences utilisateur doit être le plus rapide possible. De plus, le nombre d'interactions entre le DSS et l'utilisateur doit être aussi limité que possible afin de ne pas surcharger l'utilisateur de questions. L'utilisation des méta-critères pour agréger les politiques est un moyen de baisser ce nombre. Cependant, ce facteur n'est pas suffisant. Le nombre d'interactions dépend aussi des préférences apprises et de la vitesse à laquelle le DSS les apprends. L'agrégation des critères pour calculer le score d'une requête et la mise à jour des critères sont les deux étapes qui influent le plus sur la vitesse d'apprentissage. Nous avons testé trois opérateurs d'agrégation différents :

– la moyenne pondérée, un opérateur utilisé dans la majorité des DSS pour sa simplicité. Chaque critère est évalué indépendamment.

– l'intégrale de Choquet (Grabisch, Roubens, 2000), un opérateur plus complexe qui utilise l'importance de chaque critères et les interactions entre eux pour avoir un meilleur apprentissage. Nous avons utilisé Kappalab (Grabisch *et al*., 2006), un plug-in de R pour mettre en place nos intégrales de Choquet.

– notre propre opérateur, Kagop (*Kapuer AGgregation OPerator*) (Oglaza, 2014), qui se place entre la moyenne pondérée et l'intégrale de Choquet. En plus des critères de la requête, il utilise tous les groupes de critères (voir section 3.4). Nous utilisons cet opérateur pour voir si les groupes de critères peuvent aider le système à trouver des interactions entre plusieurs critères.

Afin de pouvoir obtenir suffisamment de données pour pouvoir comparer les différentes approches d'apprentissage, de nombreux utilisateurs et périphériques sont nécessaires. Pour passer outre ces contraintes, nous avons développé un simulateur. Ce simulateur permet deux choses. Tout d'abord, il est possible de le configurer avec un ensemble de critères sur plusieurs classes ainsi qu'une hiérarchie pour chacune des classes afin qu'il puisse générer un nombre important de requêtes en choisissant aléatoirement un critère de chaque classe. Ceci permet de simuler des requêtes d'accès. Ensuite, il permet de simuler le comportement d'un utilisateur via un ensemble de règles d'autorisation prédéfinies. Ainsi, lorsque le DSS demande une interaction lors de sa phase d'apprentissage, le simulateur répond automatiquement par accepter ou refuser selon ces règles. De la même manière, chaque fois que le DSS fait une proposition de règle, le simulateur compare cette proposition avec cette liste de règles.

Lors de nos tests, nous avons joué dix simulations de 200 requêtes aléatoires pour comparer les trois opérateurs d'agrégation. Nous les évaluons selon quatre métriques. Premièrement, nous analysons le nombre d'interactions avec l'utilisateur. Deuxièmement, nous étudions le nombre de règles d'autorisation créées qui montre le niveau d'abstraction de chaque opérateur. Pour définir une même politique, plus ce nombre est important, plus le niveau d'abstraction sera bas. Troisièmement, nous regardons le nombre de requêtes qui, pendant la simulation, n'ont pas nécessité d'interaction avec l'utilisateur. Finalement, le niveau de complétude, c'est-à-dire le pourcentage de requêtes parmi l'ensemble des requêtes possibles qui, à la fin de la simulation, sont couvertes par la politique proposée. Pour nos premiers tests, nous avons implémenté une liste de critères et trois classes de critères :

– **Quoi** : la ressource à protéger avec six critères. « Liste de contacts » et « Calendrier » ayant le méta-critère « Donnée ». « Nom » et « Adresse mail » avec le méta-critère « Information ». « GPS » et « Appareil photo » avec le méta-critère « Service ».

– **Qui** : la personne qui veut accéder à la ressource avec neuf critères. « Jimmy », « Lee » et « Billy » avec le méta-critère « Famille ». « Bob », « Jay » et « Fred » avec le critère « Ami ». « Pierrick » et « Mick » avec le critère « Collègue » et « John » avec le critère « Inconnu ».

– **Quand** : la demi journée où la demande a été faite. Deux méta-critères « Matin » et « Après-midi » regroupent les critères « lundi matin », « lundi après midi », « mardi matin », « mardi après midi », etc..

Les utilisateurs ont été simulés avec deux comportements différents. Le premier, B_{op}, est ouvert à toutes les demandes. Il n'y a qu'une seule règle à ce comportement :

– **Règle 1** : Partage Donnée, Information et Service avec Famille, Collègue, Ami et Inconnu les Matin et Après-midi.

Le deuxième comportement, B_{cx}, est plus complexe. Il accepte de tout partager, tout le temps avec la famille, avec les collègues le matin et avec les amis l'après-midi et refuse toute demande d'un inconnu. Les règles sont donc :

– **Règle 1** : Partage Donnée, Information et Service avec Famille le Matin et Après-midi

– **Règle 2** : Partage Donnée, Information et Service avec Collègue le Matin

– **Règle 3** : Partage Donnée, Information et Service avec Amis les Après-midi

– **Règle 4** : Ne partage rien avec Inconnu

Les règles ne donnent que les cas où l'utilisateur simulé accepte la divulgation. Si il n'y a pas de correspondance entre une règle de comportement et une requête, la divulgation est refusée. L'objectif de chaque algorithme est de trouver toutes ces règles aussi rapidement que possible à partir des requêtes et des décisions de l'utilisateur simulé. Les résultats sont présentés dans les figures 9 à 12 (attention, les échelles diffèrent selon les graphiques).

Les résultats montrent qu'aucun opérateur n'est vraiment meilleur qu'un autre sur les quatre métriques. Kagop montre des résultats intéressant pour les utilisateurs car il offre le meilleur taux de complétude. Après 200 requêtes, quel que soit le comportement, les trois opérateurs sont au dessus de 80 % de complétude. Plus de quatre cinquième des requêtes arrivant après seront donc directement gérés par Kapuer. Kagop atteint même une complétude de 98,9 % avec le comportement ouvert. Ce niveau doit être confronté au nombre de règles créées. Kagop arrive à un très bon niveau de complétude tout en créant moins de règles que les deux autres opérateurs car les règles proposées par Kagop fournissent plus d'abstractions.

L'autre point important pour les utilisateurs est de limiter le nombre d'interactions. Comme nous pouvions nous y attendre, le comportement complexe nécessite plus d'interactions que le comportement ouvert. La vitesse d'apprentissage a un impact sur ces interactions. L'intégrale de Choquet et la moyenne pondérée ont plus de requêtes directement gérées par le système. Cela montre que les politiques sont créées plus rapidement qu'avec Kagop, mais avec le comportement complexe, ils créent tellement de politiques qu'au final, le nombre d'interactions est plus important que celui de Kagop. L'intégrale de Choquet et la moyenne pondérée montrent des pics dans trois simulations. Dans ces simulations, dues à l'ordonnancement aléatoire des requêtes, l'apprentissage des préférences à conduit à créer très rapidement des politiques d'autorisation. Bien plus de politiques sont créées que dans les autres simulations mais il s'agit de politiques de bas niveau. Avec un faible niveau d'abstraction des politiques, le niveau de complétude est lui aussi faible et cela augmente le nombre d'interactions. Au contraire, et ce quel que soit le comportement, Kagop est plus stable que les deux autres opérateurs sur les quatre métriques.

5. Conclusion et améliorations futures

La complexité de l'informatique dite personnelle tend à exploser avec le nombre et la diversification des équipements connectés. Aujourd'hui où le problème de la protection de la vie privée est d'actualité, contrôler ces systèmes sera de plus en plus difficile surtout pour des personnes non expertes dans l'administration de systèmes ou dans la sécurité. Nous avons proposé dans cet article une nouvelle approche qui combine un système d'aide à la décision basé sur l'analyse multicritère, appelé KAPUER, aux outils classiques de contrôle d'accès afin d'aider les utilisateurs à écrire des règles d'autorisation de haut niveau. Nous avons expliqué comment intégrer KAPUER à XACML, en particulier quand et comment interagir avec l'utilisateur. De plus, nous avons étudié trois opérateurs d'agrégation pour la partie apprentissage des préférences en termes de protection de la vie privée. Notre opérateur Kagop fournit de bons résultats et son comportement est plus homogène que la moyenne pondérée ou l'intégrale de Choquet. Le travail sur la qualité et la vitesse d'apprentissage peut être amélioré sur deux aspects. Tout d'abord, nous avons utilisé ces algorithmes sans initialisation. Hors, ces algorithmes convergeraient plus rapidement après une première phase qui initialiserait les préférences de l'utilisateur. L'étude que nous avons effectué auprès de différents groupes de personnes nous a permis d'engranger des

informations sur ce qu'il fallait faire et ne pas faire durant cette phase d'initialisation. Nous travaillons actuellement sur une nouvelle façon d'apprendre rapidement des informations pertinentes sur le comportement de l'utilisateur pour pouvoir proposer plus rapidement des règles d'autorisation aux utilisateurs.

Le deuxième point à améliorer consiste à affiner les valeurs de coefficients d'apprentissage. Grâce au simulateur, nous effectuons de nouveaux tests afin de déterminer la meilleure combinaison entre les fonctions de mise à jour de valeurs de critères et la valeur λ de seuil entre l'indifférence et la préférence stricte.

Nous travaillons actuellement sur de nouvelles expérimentations plus complexes. Pour cela, nous utilisons un scénario dont les critères sont tous tirés d'éléments présents sur un smartphone. Ainsi nous expérimentons des situations pouvant se retrouver dans des situations réelles qui permettront une validation plus approfondie du système.

Remerciements

Ce travail a été partiellement financé dans le cadre du projet ANR INCOME (INfrastructure de gestion de COntexte Multi-Echelle pour l'Internet des objets - http://www.irit.fr/income/)

Bibliographie

Adomavicius G., Tuzhilin A. (2005, juin). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, n° 6, p. 734–749. Consulté sur http://dx.doi.org/10.1109/TKDE.2005.99

Ajam N., Cuppens-Boulahia N., Cuppens F. (2010). Contextual privacy management in extended role based access control model. In *Data privacy management and autonomous spontaneous security*, p. 121–135. Springer.

Bouyssou D., Dubois D., Pirlot M., Prade H. (Eds.). (2006). *Concepts et méthodes pour l'aide à la décision - analyse multicritère* (vol. 3). http://www.editions-hermes.fr/, Lavoisier.

Byun J.-W., 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*, p. 102–110. New York, NY, USA, ACM. Consulté sur http://doi.acm.org/10.1145/1063979.1063998

Calandrino J. A., Kilzer A., Narayanan A., Felten E. W., Shmatikov V. (2011). You might also like: Privacy risks of collaborative filtering. In *Security and privacy (sp), 2011 ieee symposium on*, p. 231–246.

Chabridon S., Laborde R., Desprats T., Oglaza A., Marie P., Marquez S. M. (2014). A survey on addressing privacy together with quality of context for context management in the internet of things. *Annales des Télécommunications*, vol. 69, n° 1-2, p. 47-62.

Cranor L., Langheinrich M., Marchiori M., Reagle J. (2002, 16 avril). *The platform for privacy preferences 1.0 (p3p1.0) specification.* W3C Recommendation. Consulté sur http://www.w3.org/TR/P3P/

CyanogenMod. (2013). *Privacy guard manager.* https://plus.google.com/+CyanogenMod/posts/86LLXrDpVWY. ([Online; accessed 22-september-2014])

EMC. (2014). *The internet of things.* http://www.emc.com/leadership/digital-universe/2014iview/internet-of-things.htm. ([Online; accessed 22-september-2014])

ExMachina. (2010). *Programme de serious game 2025 ex machina, production tralalere.* http://www.2025exmachina.net/jeu. ([Online; accessed 22-september-2014])

GFK. (2013). *Gfk-médiamétrie référence des équipements multimédias au 3ème trimestre 2013.* http://www.gfk.com/fr/news-and-events/press-room/press-releases/pages/gfkmediametrie-reference-des-equipements-multimedias-3eme-trimestre-2013.aspx. ([Online; accessed 22-september-2014])

Giard V. E., Roy B. (1985). *Méthodologie multicritère d'aide à la décision*. Editions Economica.

Gorry G. A., Morton M. S. S. (1971). *A framework for management information systems* (vol. 13). Massachusetts Institute of Technology.

Grabisch M., Kojadinovic I., Nantes S. P., Meyer P. (2006). Using the kappalab r package for capacity identification in choquet integral based maut. In *Proc. 11th int. conf. inform. process. and management of uncertainty in knowledge-based systems*, p. 1702–1709.

Grabisch M., Roubens M. (2000). Application of the choquet integral in multicriteria decision making. *Fuzzy measures and integrals*, n° 40, p. 348–375.

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. Consulté sur http://primelife.ercim.eu/results/documents/149-416d

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.*

Jiang X., Landay J. (2002, July). Modeling privacy control in context-aware systems. *Pervasive Computing, IEEE*, vol. 1, n° 3, p. 59-63.

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.*

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*, p. 223–238.

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*, p. 627–636.

OASIS. (2005). *OASIS XACML committee "eXtensible Access Control Markup Language (XACML) Version 2.* http://www.oasis-open.org/committees/xacml//. ([Online; accessed 22-september-2014])

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.

Oglaza A., Laborde R., Zaraté P. (2013, July). 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).

Ruiz E. V. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Letters*, vol. 4, n° 3, p. 145 - 157. Consulté sur http://www.sciencedirect.com/science/article/pii/0167865586900139

Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E. (1996, février). Role-based access control models. *Computer*, vol. 29, n° 2, p. 38–47.

Shyong K., Frankowski D., Riedl J. *et al*. (2006). Do you trust your recommendations? an exploration of security and privacy issues in recommender systems. In *Emerging trends in information and communication security*, p. 14–29. Springer.

Simon H. A. (1972). Theories of bounded rationality. *Decision and organization*, vol. 1, p. 161–176.

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)*, p. 193-200.

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).*

ZDNET. (2013). *80 milliards d'objets connectés en 2020.* http://www.zdnet.fr/actualites/80-milliards-d-objets-connectes-en-2020-39793776.htm. ([Online; accessed 22-september-2014])

Annexe

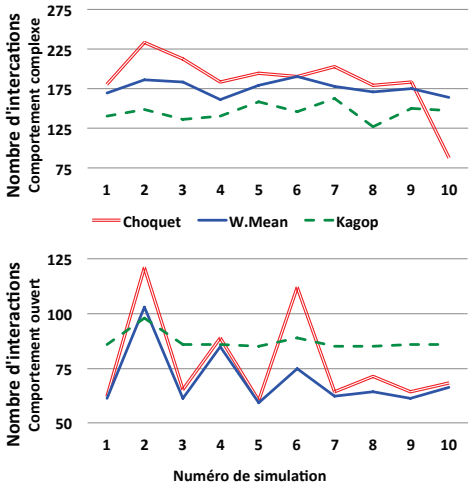

Figure 9. Résultats des simulations : Nombre d'interactions

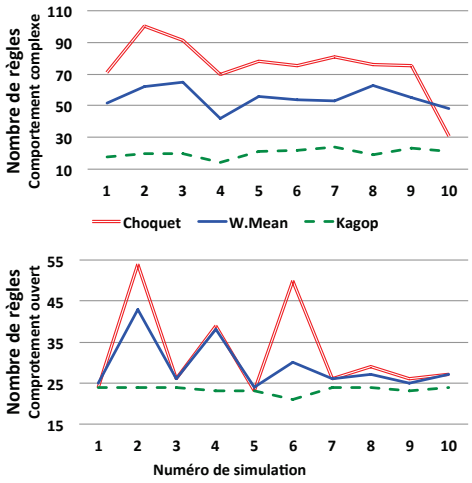

Figure 10. Résultats des simulations : Nombre de règles d'autorisation créées

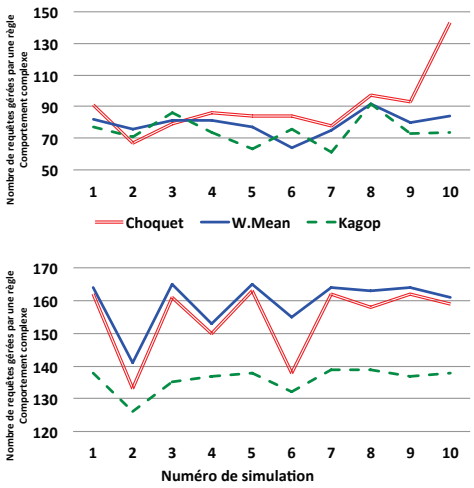

Figure 11. Résultats des simulations : Pourcentage de complétude

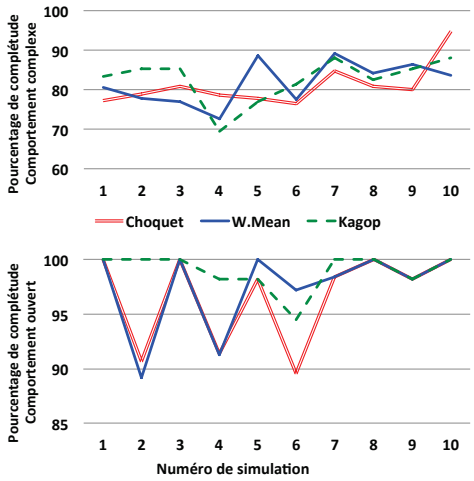

Figure 12. Résultats des simulations : Nombre de requêtes gérées par une règle d'autorisation