2010

A framework for privacy policy management in service aggregation

Peishun Wang
University of Wollongong, peishun@uow.edu.au

L. Dong
University of Wollongong, liju@uow.edu.au

Yi Mu
University of Wollongong

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Jun Yan
University of Wollongong, jyan@uow.edu.au

A framework for privacy policy management in service aggregation

Abstract

With a rapid growth of the Internet, exploring cost-effective and time-efficient methods for creating Internet services has become critical. As an emerging technology, service aggregation has been regarded as a promising candidate. However, it also raises serious issues on privacy management, as a service is usually provided by multiple providers that are usually transparent to its users. We observe that these issues have not been formally studied in the literature. In this paper, we propose a formal model for the privacy management in service aggregation and present a negotiation strategy on different privacy policies between two organizations.

Disciplines

Physical Sciences and Mathematics

Publication Details

Wang, P., Dong, L., Mu, Y., Susilo, W. & Yan, J. (2010). A framework for privacy policy management in service aggregation. In W. Shen, N. Gu, T. Lu, J. BarthÃ¨s & J. Luo (Eds.), Proceedings of the 2010 14th International Conference on Computer Supported Cooperative Work in Design (pp. 166-171). Piscataway, New Jersey, USA: IEEE.

A Framework for Privacy Policy Management in Service Aggregation

Peishun Wang*, Liju Dong*‡, Yi Mu*, Willy Susilo* and Jun Yan†
*School of Computer Science and Software Engineering
†School of Information Systems and Technology
University of Wollongong
Sydney, NSW 2522, Australia
‡School of Information Engineering
Shenyang University
Shenyang 110044, China
Email: peishun,liju,ymu,wsusilo,jyan@uow.edu.au

***Abstract*—With a rapid growth of the Internet, exploring cost-effective and time-efficient methods for creating Internet services has become critical. As an emerging technology, *service aggregation* has been regarded as a promising candidate. However, it also raises serious issues on privacy management, as a service is usually provided by multiple providers that are usually transparent to its users. We observe that these issues have not been formally studied in the literature. In this paper, we propose a formal model for the privacy management in service aggregation and present a negotiation strategy on different privacy policies between two organizations.**

Keywords-privacy policy; service aggregation.

I. Introduction

The emergence of Service Oriented Architecture (SOA) and web service technologies is making the boundaries between network domains fade out. It is a trend for a business entity to combine the various services offered by various service providers in different domains along with some of its own services and resell the aggregated services to users. Service aggregation becomes a cost-effective and time-efficient way to develop new applications and services [5], [7]. Its benefit originates from the added value generated by the possible interactions and by the large scale rather than by the capabilities of its individual service provider separately. Clearly, this paradigm creates tremendous opportunities in e-businesses. However, its nature of cross-organisation raises serious challenges for privacy management. Because aggregating services implies the sharing of information between services, privacy protection must be considered. Businesses are often prohibited by law or by contract from disclosing the personal/private information of users/partners to third parties. To protect users' privacy, organizations require privacy policies. When a user accesses proprietary services provided by a single service provider, user's privacy is protected by the privacy policy of the provider. However, when a user's information needs to be shared by multiple service providers, which might have some inconsistent or even conflicting privacy policies, privacy protection obviously becomes a more challenging problem. Therefore, advanced privacy management should be investigated. This includes the autonomous comparison of privacy policies, detection of inconsistency/conflicts, analysis of risks associated with the detected inconsistency/conflicts, and resolution of inconsistency/conflicts.

There are several useful tools for privacy policy management, such as XACML [10], P3P [12], APPEL [3], EPAL [4] and ASL [6]. These tools provide formal approaches for describing privacy policies. With novel functionalities, several privacy policy comparison and negotiation protocols are introduced [11], [2], [1], [14], [9], [8], [13]. However, these methods do not address the privacy management for service aggregation. In particular, they do not consider large privacy policy sets from multiple parties. In this paper, we address these issues by proposing a formal method for privacy policy management in service aggregation and a privacy policy negotiation method.

The remainder of the paper is organised as follows. In Section II the high-level architecture of service aggregation with privacy policies is demonstrated. Section III shows the basic components. The syntax and semantics are introduced in Section IV and V, respectively. Section VI gives several privacy policy matching protocols. Finally, Section VII concludes with future work.

II. Service Aggregation with Privacy Policies

Service aggregation is defined as logically combining the functionalities of multiple services as parts of a single, descriptive and meaningful information abstraction. In short, it is the combination of a set of services to achieve a common goal. Service aggregation allows users, service providers and a service aggregator to collaborate in highly distributed environments, and establish on-demand, short-term and dynamic business relationships for maximizing profitability. Each participant in service aggregation has its own privacy policy. A general diagram that illustrates the high-level architecture of service aggregation with privacy policies is shown in Figure 1.

978-1-4244-6763-1/10/$26.00 ©2010 IEEE

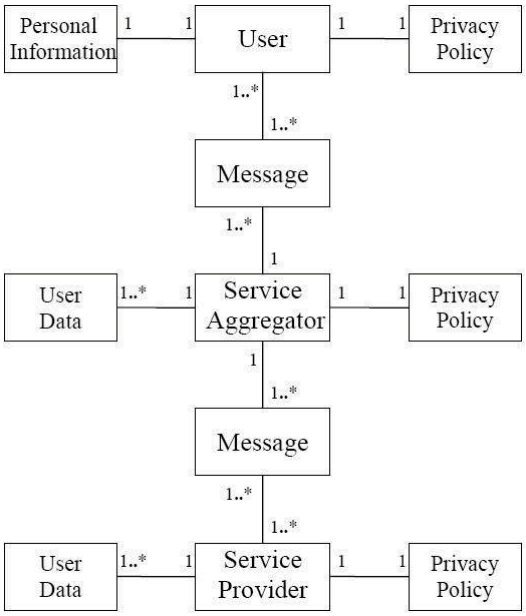

Fig. 1. High-level Architecture of Service Aggregation with Privacy Policies

As depicted in Figure 1, there are three types of parties in service aggregation: user, service providers and service aggregator. The service providers provide various public and professional services ranging from simple information retrieval services to more complex transaction oriented services, and the service aggregator works as an agency for service providers to aggregate various services offered by service providers along with some of its own services to serve users. Service aggregator and service providers have their associated privacy policies, and users have their own privacy policies (also called privacy preferences), too.

Privacy policy, as a special case of the authorization for controlling access to personal information, specifies under what conditions the personal information can be exchanged and what the information is used for. The privacy policies of service aggregator and service providers state what personal information it requires from a user and how the information will be used. The privacy policy of a user states what private information she is willing to share, with whom and under what circumstances it may be shared.

A privacy policy is attached to a software agent that acts for a user, a service aggregator or a service provider. Prior to the activation of a particular service, the agent for the user and the agent for the service aggregator will undergo a privacy policy exchange, in which the policies are examined for compatibility. The service is only activated if the policies are consistent. Similarly, when the service aggregator would integrate a service from a service provider who needs some personal information, the agent for the service aggregator and the agent for the service provider will also examinate their privacy policies for compatibility. The services are aggregated only if the policies are not conflictive.

III. Basic Components

Throughout the paper, we denote the set of natural numbers by $\mathbb{N}$ and the set of numbers $\{1, 2, \cdots, n\}$ by $[n]$.

Our model relies on the following three basic components.

The first component is a set $\mathcal{S}$ = U $\cup$ SA $\cup$ SP of $subjects$ that can operate an object actively. Subjects can be $users$ (i.e., elements of U), or the $service\ aggregator$ (the only element of SA), or $service\ providers$ (i.e., elements of SP).

The second basic component is a set $\mathcal{O}$ = D $\cup$ P $\cup$ M of $objects$, which are the passive resources to be operated by subjects. Objects in our model include $data$ (i.e., elements of D) of users, service aggregator and service providers, their $privacy\ policies$ (i.e., elements of P), and $messages$ (i.e., elements of M) that is associated with two subjects (a user and the service aggregator, or the service aggregator and a service provider).

The last basic component is a set $\mathcal{OP}$ of $operators$, denoting the operations that subjects can execute on the objects in the system. Operators to be considered depend on the underlying activities in the system, usually include read, write, add, delete, create, revoke, send, receive, etc.

The basic components of our model are formalized by the notion of Service Aggregation as follows.

Definition III.1 (Service Aggregation) *A* **Service Aggregation** *consists of the following components:*

1) **Subjects.** *A countable set $\mathcal{S}$ of labels, called $subject\ identifiers$. This set is partitioned into three subsets, namely, U ($users$), SA ($service\ aggregator$) and SP ($service\ providers$), where there is only one element in SA. The service aggregator interacts with users and service providers, but users do not interact with service providers.*
2) **Objects.** *A countable set $\mathcal{O}$ of labels, called $object\ identifiers$. This set is partitioned into three subsets, namely, D ($data$), P ($privacy\ policies$) and M ($messages$).*
3) **Operators.** *A countable set $\mathcal{OP}$ of labels, called $operator\ identifiers$.*

IV. Syntax of Service Aggregation

We assume that a service aggregation $\mathcal{W}$ has been fixed. Over $\mathcal{W}$, a set of rules and vocabularies can be specified. In addition, we assume that the following sets are given:

1) A finite set $\mathcal{C}$ of data categories with a fixed vocabulary, which are structured in hierarchies as follows. $\mathcal{C}$ contains the special category sc, called $super\ category$. On sc the partial order relation $\preccurlyeq$ is defined such that $sc \preccurlyeq c$, for all $c \in \mathcal{C}$. Given two categories c_i and c_j, if $c_i \preccurlyeq c_j$ we say that c_j is a $subcategory$ of c_i.
2) A finite set $\mathcal{V}$ of context variables with a fixed domain.
3) A finite set $\mathcal{D}$ of context data values with a fixed semantics.

Definition IV.1 (Personal Information) Personal information *is a set PI of data about an individual.*

Personal information can be partitioned into a set of data categories, such as contact information, financial information, medical data, demographic data, internet protocol information, cookie information, purchasing data, and others. Figure 2 gives an example of the hierarchy of Personal Information [6].

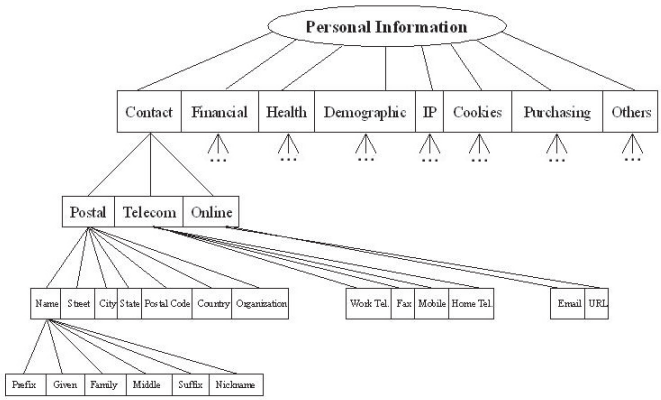

Fig. 2. Hierarchy of Personal Information

Definition IV.2 (User Data) **User data** *includes some of personal information about a customer and some information that the system creates in the process of replying the customer's service request.*

Usually user data about a customer contains not only the customer's personal information but also the services the customer requested and the corresponding service providers.

Definition IV.3 (Message) *A* **message** *is a piece of information one subject sends to another subject.*

Definition IV.4 (Constraint) *A* **component** *is a pair* $com = (dv, val)$ *of a variable* dv *and a set* val*, where* $dv \in \mathcal{V}$ *and* $val \subseteq \mathcal{D} \wedge val \neq \emptyset$*.* val *is a set of values assigned to the variable* dv*. Given two components* $com_i = (dv_i, val_i)$ *and* $com_j = (dv_j, val_j)$*,* $com_i \subseteq com_j$ *iff* $dv_i = dv_j \vee val_i \subseteq val_j$*.*

A **constraint** *is a finite set* $cstr_i = (com_{i_1}, com_{i_2}, \cdots, com_{i_n})$ *(*$n \in \mathbb{N}$*) of components* com_{i_j} *(*$j \in [n]$*). Given two constraints* $cstr_i = (com_{i_1}, com_{i_2}, \cdots, com_{i_n})$ *(*$n \in \mathbb{N}$*) and* $cstr_j = (com_{j_1}, com_{j_2}, \cdots, com_{j_m})$ *(*$m \in \mathbb{N}$*), if* $\forall com_{i_k}$ *(*$k \in [n]$*),* $\exists com_{j_l}$ *(*$l \in [m]$*), such that* $com_{i_k} \subseteq com_{j_l}$*, we say* $cstr_j$ *is a subconstraint of* $cstr_i$*, or* $cstr_i$ *is a supconstraint of* $cstr_j$*, denoted by* $cstr_i \preccurlyeq cstr_j$*.*

Definition IV.5 (Term) *A* **term** *is a reference* t *to a data category on specific pieces of personal information, where* $t \in \mathcal{C}$*.*

Definition IV.6 (Privacy Policy Vocabulary) *A* **privacy policy vocabulary** *is a pair* $\mathcal{PPVOC} = (\mathcal{T}, \mathcal{CSTR})$ *where* $\mathcal{T}$ *and* $\mathcal{CSTR}$ *are a finite set of terms and a finite set of constraints, respectively.*

Definition IV.7 (Privacy Policy) *For a privacy policy vocabulary* $\mathcal{PPVOC}$*, a* **privacy policy** *is a tuple* $Pol = \{(T_1, CSTR_1), (T_2, CSTR_2), \cdots, (T_n, CSTR_n)\}$ *(*$n \in \mathbb{N}$*) of pairs* $(T_i, CSTR_i)$ *(*$i \in [n]$*) where* $T_i \in \mathcal{T}$ *and* $CSTR_i \in \mathcal{CSTR}$*, which defines what kinds of personal information* $\{T_i\}_{i=1,\cdots,n}$ *are exchanged between two subjects and how the personal information are used and stored under the corresponding constraints* $\{CSTR_i\}_{i=1,\cdots,n}$*.*

For privacy policies, usually there are following context variables: Recipient, Data, Purpose, Retention, Disclose-to, Consent, Obligation, Condition, Remedy, and Action. The first four variables are compulsory in a privacy policy, and others are optional. For the optional variables, different privacy policy languages define different parties of them. Notice that, as stated in [10], "Purpose" is the only variable that must exist in a privacy policy but not occur in a tranditional access control policy. The notion of purpose plays a major role in protecting privacy, since privacy policies are concerned with the purposes that personal information is cllected for rather than the actions that recipients perform on the personal information, and traditional access control policies cannot easily achieve privacy protection.

For a context variable, there are a lot of context values, which are defferent in different privacy policy language. For example, the values to the variable "Recipient" in P3P are "ours", "delivery", "same", "other-recipient", "unrelated", and "public".

Example IV.1 *Consider the privacy policy set by a subject* s *as follows.*

$$\begin{aligned} Pol^s &= ((T_1^s, CSTR_1^s), (T_2^s, CSTR_2^s)) \\ &= ((T_1^s, (com_{1-1}^s, com_{1-2}^s)), (T_2^s, (com_{2-1}^s))) \\ &= ((T_1^s, ((dv_{1-1}^s, val_{1-1}^s), (dv_{1-2}^s, val_{1-2}^s)), \\ &\quad (T_2^s, (dv_{2-1}^s, val_{2-1}^s))) \end{aligned}$$

It states that, the privacy policy protects two categories of personal information, T_1^s and T_2^s. The constraint $CSTR_1^s$ is for T_1^s and the constraint $CSTR_1^s$ is for T_1^s. For the constraint $CSTR_1^s$, there are two variables, dv_{1-1}^s and dv_{1-2}^s, with the values, val_{1-1}^s and val_{1-2}^s, respectively, and for the constraint $CSTR_2^s$, there are one variable, dv_{2-1}^s, with the value, val_{2-1}^s. Now we assign "Home-Telephone-Number", "Purpose", "Recipient", "contact", "ours & delivery", "Credit-Card-Information", "Retention" and "no-retention" to T_1^s, dv_{1-1}^s, dv_{1-2}^s, val_{1-1}^s, val_{1-2}^s, T_2^s, dv_{2-1}^s and val_{2-1}^s, respectively, the detailed privacy policy is as follows.

$$\begin{aligned} Pol^s &= ((Home - Telephone - Number, \\ &\quad (Purpose, contact), \\ &\quad (Recipient, ours\ \&\ delivery)), \\ &\quad (Credit - Card - Information, \\ &\quad (Retention, no - retention))) \end{aligned}$$

This means, in order to contact the user, s collects a user's home telephone number. However, not only s knows the

telephone number, she will give the telephone number to the third party for delivery. s also requires the information of the user's credit card, but she will not keep the information any more after it is used for current business matter.

Definition IV.8 (Service Aggregation Vocabulary) *A* **service aggregation vocabulary** *is a triple $\mathcal{SAVOC}$ = ($\mathcal{S}$, $\mathcal{O}$, $\mathcal{OP}$) where $\mathcal{S}$ is a set of subjects including users, service aggregator and service providers, $\mathcal{O}$ is a set of objects including personal information, privacy policies, user data and messages, and $\mathcal{OP}$ is a set of operators.*

V. Semantics of Service Aggregation

We point out that semantic aspects play a crucial role in our proposal, since they make the model innovative with respect to existing approaches. This matter will be deeply analyzed in this section. In this section, we provide the formal semantics of service aggregation.

Definition V.1 (Request) *A* **request** *is a message that a subject sends to another subject to ask for some resource. For a privacy policy vocabulary $\mathcal{PPVOC}$, a* **personal information (PI) request** *is a request taking the form of a finite set $req = ((t_1, cstr_1), \cdots, (t_m, cstr_m))$ ($m \in \mathbb{N}$) of pairs of term t_i and constraint $ctsr_i$ ($i \in [m]$) to ask for some personal information.*

Definition V.2 (Matching Rule) *Given a PI request $req = ((t_1, cstr_1), \cdots, (t_m, cstr_m))$ ($m \in \mathbb{N}$) and a privacy policy $Pol = \{(T_1, CSTR_1), \cdots, (T_n, CSTR_n)\}$ ($n \in \mathbb{N}$), the request matches the privacy policy, denoted by $req \sqsubset Pol$, iff $\forall (t_i, cstr_i) \in req$ ($i \in [m]$), $\exists (T_j, CSTR_j) \in Pol$ ($j \in [n]$), such that $T_j \preccurlyeq t_i \wedge CSTR_j \preccurlyeq cstr_i$.*

If a request matches a constraint, then it matches all supconstraints of the constraint. i.e., the match rules with positive results are additionally inherited up the hierarchies. If a request does not match the constraint (in all hierarchies), then it does not match all subconstraints of the constraint. i.e., the match rules with negative results are inherited down the hierarchies.

The set of privacy policies P is partially ordered by $\preccurlyeq$. The partial order is defined by a *cover* relationship. It models a $part-of$ relation among privacy policies.

Definition V.3 (Cover) *The* **cover** *relationship $\preccurlyeq$ is defined as follows: a privacy policy $Pol^y = \{(T_1^y, CSTR_1^y), \cdots, (T_n^y, CSTR_n^y)\}$ covers another privacy policy $Pol^x = \{(T_1^x, CSTR_1^x), \cdots, (T_m^x, CSTR_m^x)\}$ ($n, m \in \mathbb{N}$), denoted with $Pol^y \preccurlyeq Pol^x$, iff $\forall (T_i^x, CSTR_i^x)$ ($i \in [m]$), $\exists (T_j^y, CSTR_j^y)$ ($j \in [n]$), such that $T_j^y \preccurlyeq T_i^x \wedge CSTR_j^y \preccurlyeq CSTR_i^x$.*

Definition V.4 (Incomparable) *Given two privacy policies Pol^x and Pol^y, if neither $Pol^x \preccurlyeq Pol^y$ nor $Pol^y \preccurlyeq Pol^x$ holds, the two privay policies are said to be* **incomparable**.

Definition V.5 (Formula) *For a service aggregation vocabulary $\mathcal{SAVOC}$, a* **formula** *is a construction of form: $op(s_1, o) :: s_2$, where $s_1, s_2 \in \mathcal{S}$, $o \in \mathcal{O}$, and $op \in \mathcal{OP}$, which states that the subject s_1 executes the operator op on object o and outputs the result to the subject s_2. If two subjects are the same one, the formula is denoted with $op(s_1, o)$. If there are more than one subject executing a operator on more than one object, all the subjects should be included in a pair of parentheses (), and so do all objects.*

Example V.1 *A user u creates her personal information pi, the formula is $create(c, pi)$.*

Example V.2 *The service aggregator sa uses its own information inf_{sa} to replace a user's personal information pi, the formula is $replace(sa, (pi, inf_{sa})$.*

Example V.3 *The service aggregator sa and a service provider sp negotiate on the personal information request req and the result of negotiation is send to the service provider sp, the formula is $negotiate((sa, sp), req) :: sp$.*

Definition V.6 (Conditional Formula) *A* **conditional formula** *is a formula with a condition. It takes the form: $op(s_1, o) :: s_2 \leftarrow con(X)$, which states that subject s_1 executes the operation op on object o and outputs the result to the subject s_2 when the condition $con(X)$ is true, where the X can be a formula or matching rule. A condition is either a positive condition $con(X)$ or a negative condition $\neg con(X)$. Two conditions are complementary if they are of the form $con(X)$ and $\neg con(X)$.*

Example V.4 *Let's consider the following conditional formula,*

$send(u, pi) :: sa \leftarrow con(req^{sa} \sqsubset Pol^u)$.

It states that, a user u sends her personal information pi to the service aggregator sa if the request of the service aggregator req^{sa} matches the privacy policy of the user Pol^u.

VI. Matching Privacy Policy

To make their services available to users, service providers have to register the services in the service aggregator. Different service may collect different personal information of users. For every service that would be registered, the service provider sends a PI request to the service aggregator. The service aggregator checks if the request matches its privacy policy. If the request does not match the privacy policy, the service aggregator will negotiate with the Service Provider on the unmatched part in the request. The strategy of negotiation is as follows.

Case 1: Service Aggregator sa does not collect the data $data_{unmatch}$ in the unmatched part ump.

Rule 1.1 Service Aggregator sa asks Service Provider sp to remove the unmatched part ump from the request req^{sp}.

Rule 1.2 sa may collect the data $data_{unmatch}$ by negotiating with the user who needs the service.

Rule 1.3 sa may replace the data $data_{unmatch}$ with its own information inf_{sa} to reply sp.

Case 2: The constraints $cstr^{sp}_{unmatch}$ on the data $data_{unmatch}$ in the request req^{sp} are conflictive to the constraints in the privacy policy of Service Aggregator sa.

Rule 2.1 Service Aggregator sa asks Service Provider sp to change the constraints $cstr^{sp}_{unmatch}$ to make the request req^{sp} match the privacy policy of sa.

Rule 2.2 sa may replace the data $data_{unmatch}$ with its own information inf_{sa} to reply.

We model the above rules as the following algorithm.

Algorithm: $negotiate((sa, sp), ump)$

INPUT: sa, sp, ump
OUTPUT: $TRUE$ if the negotiation is fail, $FALSE$ otherwise.
METHOD:

```
1:  if ¬con(collect(sa, data_unmatch)) then
2:      if remove(sp, ump) then
3:          return TRUE
4:      else if collect(sa, data_unmatch) then
5:              return TRUE
6:          else if replace(sa, (data_unmatch, inf_sa) then
7:                  return TRUE
8:              else return FALSE
9:              endif
10:         endif
11:     endif
12: else
13:     if change(sp, cstr^sp_unmatch) then
14:         return TRUE
15:     else if replace(sa, (data_unmatch, inf_sa) then
16:             return TRUE
17:         else return FALSE
18:         endif
19:     endif
20: endif
```

Now we give the detailed procedure of matching privacy policies as follows.

Step 1 Service Provider sp sends a PI request req^{sp} to Service Aggregator sa, who will check whether the request req^{sp} matches its privacy policy Pol^{sa} or not.

Step 2 If req^{sp} does not match Pol^{sa}, sa negotiates with sp on the unmatched part ump.

step 3 If the result of negotiation is false, then sa rejects sp to register the service.

Step 4 If req^{sp} matches Pol^{sa} or the result of negotiation is true, sa accepts sp as a legitimate service provider for the service.

It is modeled as follows.

Matching Protocol

1: $send(sp, req^{sp}) :: sa$
2: $negotiate((sa, sp), ump) \leftarrow \neg con(req^{sp} \sqsubset Pol^{sa})$
3: $reject \leftarrow \neg con(negotiate((sa, sp), ump))$
4: $accept \leftarrow con(negotiation((sa, sp), ump)) \cup con(req^{sa} \sqsubset Pol^{sa})$

When a service provider updates its privacy policy, if the old privacy policy can cover the new one, it is fine with the system. Otherwise, the service provider has to check if the change of privacy policy affects the original PI request for the registered service. If not, it is ok for the system. Otherwise, the matching protocol is excuted. If the output of the protocol is “reject", the registered service should be removed from the available service list of the service aggregator. Otherwise, it is retained.

When service aggregator updates its privacy policy, it also checks if the new privacy policy can cover the old one. If yes, no more action needs to be taken. Otherwise, the service aggregator will run the matching protocol for every registered service. The service aggregator will hold the service with the output “accept" of the protocol or remove the service from its available service list with the output “reject".

We omit the modelling of above two scenarios here.

VII. Conclusions and Open Problem

In this paper, we proposed a logic model that naturally supports the encoding of complex privacy policy specifications in service aggregation. The syntax and semantics are both simple and intuitive. Based on them, we showed how to negotiate and match two privacy policies.

This paper represents our first attempt at a formal study of privacy policies in service aggregation. We plan to extend our semantics-centered approach to areas including privacy policy enforcement and privacy practice auditing.

Acknowledgment

The authors would like to thank Smart Service CRC Australia for the support in making this work and the industry partners who provided valuable input on the direction of this work.

References

[1] M. Backes, G. Karjoth,W. Bagga, and M. Schunter,“Efficient comparison of enterprise privacy policies”, The 2004 ACM Symposium on Applied Computing (SAC), ACM Press, pp.375-382. 2004,

[2] M. Bennicke and P. Langendorfer, “Towards automatic negotiation of privacy contracts for internet services”, The 11th IEEE International Conference on Networks, Sydney, Australia, October 2003.

[3] L. Cranor, M. Langheinrich, and M. Marchiori, “A P3P Preference Exchange Language 1.0 (APPEL1.0)”, W3C Working Draft, 2002. Available: http://www.w3.org/TR/P3P-preferences/.

[4] IBM, “The Enterprise Privacy Authorization Language (EPAL)”. Available: http://www.zurich.ibm.com/security/enterprise-privacy/epal.

[5] R. Kanneganti and P. Chodavarapu, “SOA Security”, Manning Publications Co. Greenwich, CT, 2008

[6] G. Karjoth and M. Schunter, “A Privacy Policy Model for Enterprises”, The 15th IEEE Computer Security Foundations Workshop (CSFW’02), pp. 271-281, 2002
[7] R. Khalaf and F. Leymann, “On web services aggregation”, Technologies for E-Services 2003. LNCS, vol. 2819, pp. 1-13. Springer, Heidelberg (2003)
[8] K. Kursawe, G. Neven, and P. Tuyls, “Private Policy Negotiation”, FC2006, LNCS 4107, pp. 81-95. 2006
[9] M. Maaser and P. Langendoerfer, “Automated negotiation of privacy contracts”, 29th Annual International Computer Software and Applications Conference (COMPSAC’05), July 2005.
[10] OASIS. Security services technical committee. “eXtendible Access Control Markup Language (XACML)” Version 2.0. Available: http://docs.oasis-open.org/xacml/xacmlrefs.html. 2006
[11] K. E. Seamons, M. Winslett, and T. Yu, “Limiting the disclosure of access control policies during automated trust negotiation”, NDSS 2001. The Internet Society, 2001.
[12] W3C, “The platform for privacy preferences 1.1 (P3P1.1) specification”, 2005. Available: http://www.w3.org/TR/2005/WD-P3P11-20050701/
[13] D.D. Walker, E.G. Mercer, and K.E. Seamons, “Or Best Offer: A Privacy Policy Negotiation Protocol”, 2008 IEEE Workshop on Policies for Distributed Systems and Networks, pp. 173-180. 2008
[14] W. Winsborough and N. Li, “Safety in Automated Trust Negotiation”, 2004 IEEE Symposium on Security and Privacy (S&P’04), page 147-160, 2004