

## **DATABASE-CENTRIC TOOL FOR ANALYZING AND MANAGING POLICY CHANGES IN WEB SERVICES**

Tu Thanh Le<sup>1,2</sup>, Halvard Skogsrud<sup>1,2</sup>, & Boualem Benatallah<sup>1,2</sup>

<sup>1</sup> School of Computer Science & Engineering, University of New South Wales, Sydney NSW 2052 Australia

<sup>2</sup> National ICT Australia (NICTA), Sydney NSW 1430, Australia  
{tlex518, halvards, boualem}@cse.unsw.edu.au

**ABSTRACT:** The development of e-commerce systems has enabled companies to expand their business to more customers in many geographical areas. However, in providing services to more strangers, companies face security challenges. The issue of access control is not new in the e-commerce field, since customers are traditionally required to register accounts within the companies' local domains. However, in the Web services context, parties involved in transactions are often determined at run-time, which means they may be completely unknown to each other. Hence, traditional access control approaches do not fit in the paradigm of Web Services. Trust negotiation is a form of access control where trust between a requester and the services provider is established gradually during a negotiation by exchanging credentials. Most existing trust negotiation languages are complex and the issue of policy lifecycle management is often overlooked. Trust-Serv is a negotiation framework which provides both abstractions and tools to address those issues. The aim of this project is twofold: Firstly, we will provide a database component for the Trust-Serv framework to improve scalability. Secondly, we will provide a management toolset for analysing the impact of policy evolution and to guide in handling ongoing negotiations.

### 1. INTRODUCTION

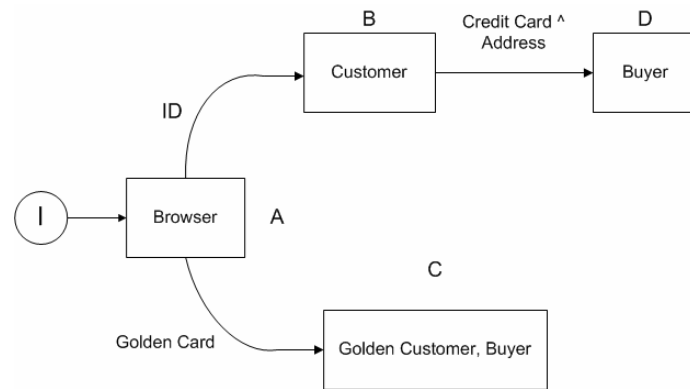
The development of the Internet has enabled new ways of conducting business, one of which is the Web-based commerce. In particular, Web services have attracted attention as an effective means of enabling B2B e-commerce. Web Services are self contained, self describing, modular application that can be published, located and invoked across the Web [V01]. With this new technology, enterprises are able to expand their businesses to more customers from diverse geographical areas. This promising prospect, however, also creates new security vulnerabilities. For example, a company may want to give a stranger outside of its security domain the permission to use its services. Before granting the access, the company must examine the eligibility of the requester based on its access control policies. The access control issue is not new in the e-commerce field since customers are traditionally required to register accounts within the companies' local domains. Such approach does not suit Web services, since parties involving in transactions may be determined at run-time, which means they may be completely unknown to each other [HBC04]. Simply providing an identity as part of the registration is not enough information to make decisions regarding access requests.

Trust negotiation is an access-control model that establishes trust between service requesters and service providers in a negotiation, avoiding the requirement of requester identities. The requester and the provider exchange credentials, which are signed assertions describing attributes of the owner, to determine access [HBC04]. Several trust negotiation systems exist, such as [BFIK00], [HMNR00] and [WY02], however, there are still several issues that need to be addressed. First, the trust negotiation policy specification using most existing policy languages are complex. Second, the lifecycle management of policies is often overlooked while policies are usually changed to reflect the changing business strategies [HBC04].

Trust-Serv [HBC04] is a framework that proposes a model-driven approach to trust negotiation in Web services. It features a trust negotiation policy model based on state machines that is supported by both abstractions and tools that permit the lifecycle management of trust negotiation policies. Our focus of this project, is first, to implement a database component to support the Trust-Serv framework and second, to design a management toolset for analysing the impact of policy changes and to guide in handling ongoing negotiations through policy changes.

## 2. MODELLING TRUST NEGOTIATION POLICIES

Trust Negotiation policies define which credentials are to be disclosed at a given state of the negotiation and the outcome of fulfilling this request (a transition to another state). The Trust-Serv framework developed by Skogsrud, Benatallah and Casati [HBC04] employs state machines to specify trust negotiation policies. This policy model is briefly presented here to provide a foundation for the work described in this paper. An example of an online flower shop trust negotiation policy P.I is given in Figure 2.1.



Role	Operation	Credential
Browser	Register, Search	Verified by Visa
Customer	Checkout	Flower shop member
Golden Customer	Special offer	
Buyer	Purchase	

Figure 2.1: A trust negotiation policy P.I of a flower shop

A trust negotiation policy consists of states, transitions, roles and privileges. States represent levels of trust achieved in a negotiation. They are mapped with roles which represent permissions. By entering a new state, a requester is awarded the roles mapped to that state. This means that the instance gains permissions for performing some functions or using services assigned to that role. Those rights are called privileges. Roles and privileges are cumulative, meaning that they are not lost if higher levels of trust are attained. In order to move from one state to another to achieve more roles or privileges, a negotiation instance is required to follow the appropriate transition and to satisfy labelled conditions if any.

In Figure 2.1 above, a requester starts the negotiation by sending a request to the provider. The provider places the requester at the initial state, State A, with a role of Browser so he can search the service content from the provider. If the requester wants to select items and check out the shopping cart, a membership identity of the flower shop must be shown. When these conditions are satisfied, the user can follow the transition from State A to State B whose mapped role is Customer. To be able to purchase the selected flowers, the customer is required to disclose his credit card and address details. Alternatively, if a Golden Card is presented, the customer will be granted Golden Customer and Buyer role straight away

## 3. POLICY VIOLATION DETECTION

This section describes violations that can occur when a trust negotiation policy is changed while negotiations are active.

### 3.1 Policy violation

Trust negotiation policies are rarely set in stone. They are modified and refined due to various factors such as business strategy changes, new applications, or laws and regulations. For example, the owner of the flower shop mentioned above may desire a more restricted condition for being a customer of the shop. The policy is modified so that a guarantee from a golden customer is now

required to be disclosed in conjunction with the identity to gain the Customer role. The new policy P.F is given as in Figure 3.1 on the next page.

The changes in negotiation policy might lead to violations for ongoing negotiations because the promises of giving privileges if requirements are satisfied in the old policy might be no longer hold in the new one [HBCF04]. In the flower shop example, if all the negotiations are moved (“migrated”) to the new policy, the changes will cause requesters in State B to lose privileges. This is because they are now required to disclose a Guarantee credential, not just an ID credential as in the old policy, to be granted the role of Customer. When the new policy is applied and the negotiations are migrated, the Customer role is revoked for every negotiation instance in State B, and consequently they have to move back to State A.

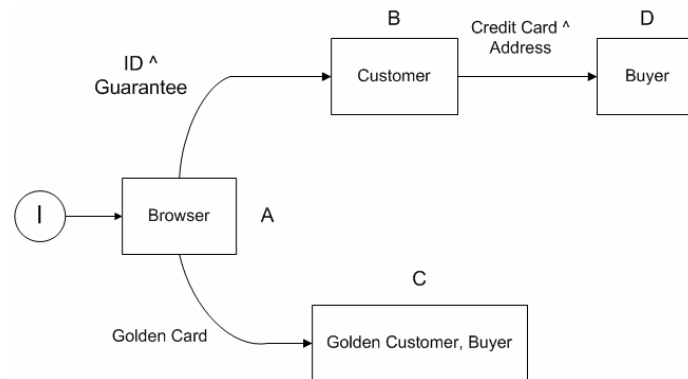


Figure 3.1: A modified policy P.F of a flower shop

The policy changes, however, are not necessary to cause privilege loss. There might be changes which make conditions of a transition to a state less restrictive so requesters who had satisfied the old condition will also comply with the new one. Another case is when the modification provides more privileges mapped to a role, for example, if the change in the flower shop policy is simply adding more privileges to Golden Customer role such as premium delivery. For all of those changes, no violation arises.

### 3.2 Replaceability analysis

When policy changes occur while there are probably thousands of negotiations are going on, simply aborting all negotiations is not adequate. A huge amount of work has been done might be lost if all customers have to start from the beginning. Another straightforward solution is allowing all requesters to complete according to the old policy. This method requires that the service providers tolerate negotiations to continue using the old policy. This is not always applicable, say if a mistake had been made in the old policy that meant some requesters could gain privileges they should not be able to obtain.

The inadequacy of those two solutions above has raised the need of more efficient strategies. This can be achieved by analysing policy changes in the old policy and instances involving in the negotiation. First, we should examine whether the changes restrict any privileges. If changes relax condition, it is safe to migrate negotiation instances from the old policy to the new one. However, if the changes restrict privileges, there will be violation in the migration. Second, the location of changes needs to be investigated. When a change causes more restricted conditions at one state, requesters from states before it have not been affected by the change yet. There are also requesters following different paths in the policy which the change does not apply at all. Those instances can be migrated into the new policy without any violation.

In order to determine best approaches to the violation caused by policy changes, we identify the compliance of the new policy with the old one with respect to individual or sets of negotiations into three replaceability classes: total, prefix and postfix [HBCF04]. The term of “replaceability” means the capability which a new trust negotiation can replace the old one under certain circumstances.

- Total replaceability: a new policy P.F can totally replace an old policy P.I if every negotiation from P.I can be migrated into P.F without violation.

- Prefix replaceable: when P.F cannot totally replace P.I, there are still some states or some paths that are not affected by the changes. For example, in the flower shop case above, requesters who achieved Golden Customer role do not violate any policy rules because the requirements to get the Golden Customer role are still the same. These instances are classified as prefix replaceable and can be migrated into the new policy P.F.
- Postfix replaceable: when P.F cannot totally replace P.I, it is important to identify states from which there is no more restricted change. In the flower shop policies (Figure 2.1 and 3.1), from State B onward, there is no change for the rest of the policy. If the changes from previous states are tolerable, it is not necessary to migrate requesters at State B to the new policy. Instead, they can be migrated into a temporary ad hoc policy such the one shown in Figure 3.2.

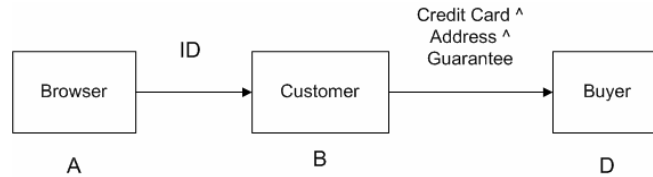


Figure 3.2: A temporary ad hoc policy P.T of a flower shop

The ad hoc policy states that requesters at State B can keep their roles even though the conditions for obtaining them have been further restricted. However, if they desire to purchase from the flower shop, they have to disclose credit card and address details according to the old policy, and guarantees from Golden Customers according to the new policy.

### 3.3 Credential requisite approach

We introduce a method of storing the credential requisites of a privilege to support the replaceability analysis. Let the privileges corresponding to State A, B, C and D be PrivA, PrivB, PrivC and PrivD respectively. We define a function  $\gamma$  as a mapping from a privilege to a set of required credential needs to be disclosed to be granted that privilege. The  $\gamma$  function applied to the two policies P.I (Figure 2.1) and P.F (Figure 3.1) is as Table 3.1 below.

Table 3.1: Required Credential Function  $\gamma$

$\gamma$	P.I	P.F
PrivA	$\emptyset$	$\emptyset$
PrivB	{ ID }	{ ID, Guarantee }
PrivC	{ GoldenCard }	{ GoldenCard }
PrivD	{ ID, CreditCard, Address }	{ ID, Guarantee, CreditCard, Address }

We also store the credentials that a requester has disclosed so far in the negotiation. For example, if a requester R1 has achieved the Buyer role in the old policy, he has given his ID, credit card and address details to the provider. A requester R2 at State B has to give only his identity to achieve the Customer role. The function  $\eta$  is defined as the set of already disclosed credentials by a requester. The function  $\eta$  applied to the requesters R1 and R2 can be presented as:

$$\eta (R1) = \{ID, CreditCard, Address\}$$

$$\eta (R2) = \{ID\}$$

The replaceability analysis can be carried out at policy-level or instance-level. The policy-level method is called *static replaceability analysis* which compares two policies to determine the

replaceability. Instance-level analysis takes a deeper investigation to each individual requester and its already disclosed credentials. It is called *dynamic replaceability analysis*.

Static analysis can be applied to determine total replaceability by comparing the credential requisites from both policies.

**Definition 1 – Total replaceability:** *A trust negotiation policy P.I is totally replaceable by a policy P.F if the following condition holds:*

$$\forall p \in \text{privileges} : \gamma(p)^{P.F} \subseteq \gamma(p)^{P.I}$$

Total replaceability occurs when every set of credential requisites to be granted a privilege in the new policy is a subset of a corresponding set in the old one. If P.I is totally replaceable by P.F then  $\gamma(\text{PrivA})^{P.F}$  must be a subset of  $\gamma(\text{PrivA})^{P.I}$ ,  $\gamma(\text{PrivB})^{P.F}$  must be a subset of  $\gamma(\text{PrivB})^{P.I}$  and the same with PrivC and PrivD. It is not true in the flower shop case because for PrivB, {ID, Guarantee} is not a subset of {ID} and for PrivD, {ID, Guarantee, CreditCard, Address} is not a subset of {ID, Guarantee, CreditCard}.

Prefix replaceability, however, can be approached by both static and dynamic analysis.

**Definition 2 – Prefix replaceability (static analysis):** *A privilege p from a policy P.I is prefix replaceable by that privilege p at P.F if the following condition holds:*

$$\gamma(p)^{P.F} \subseteq \gamma(p)^{P.I}$$

The definition states that a privilege from P.I is prefix replaceable in P.F if the credential requisites to be granted that privilege in P.I are a subset of the corresponding set of credential requisites in P.F. Looking at Table 3.1, it can be seen that State A and State C satisfy this condition and can be prefix replaceable. It means that every negotiation instances at State A and C of the old policy are safe to be migrated in to the new policy. The static analysis, however, in more complex policies, might omit those requesters who are not in these prefix replaceable state but follow some paths which are not affected by the changes.

**Definition 3 – Prefix replaceability (dynamic analysis):** *A requester r who has been granted a privilege p from a policy P.I is prefix replaceable if the following condition holds:*

$$\gamma(p)^{P.F} \subseteq \eta(r)$$

For a requester who has been granted a privilege p from the old policy, if the set of credential requisites to get that privilege p in the new policy are a subset of the set of all credentials he has disclosed so far, the requester has the prefix replaceability and can be migrated into the new policy. Note that we only need to examine the current privilege of the requester, not all privileges that it has acquired, because the credential requisite method has taken into account all the credentials that the requester has disclosed from the beginning. The dynamic analysis identifies more cases of replaceability because it can take into account individual circumstances of requesters and the paths they have followed.

In order to determine postfix replaceability, we introduce a slightly different approach with the credential requisite method. Instead of storing the required credentials needs to be disclosed to be granted a privilege, we collect the credential requisites needs to be disclosed in the future to get to the final states counting from a given privilege of a state. There might be several final states so we have to take into account all of them. We define function  $\tau$  as a mapping from a privilege to the set of the to-be-disclosed credential requisites mentioned above. The function  $\tau$  applied to the two policies P.I and P.F (Figure 2.1 and 3.1) is as the following table:

Table 3.2: The to-be-disclosed credential requisite function  $\tau$

$\tau$	P.I	P.F
--------	-----	-----

PrivA	{ ID, CreditCard, Address } U { GoldenCard }	{ ID, Guarantee, CreditCard, Address } U { GoldenCard }
PrivB	{ CreditCard, Address }	{ CreditCard, Address }
PrivC	∅	∅
PrivD	∅	∅

**Definition 4 – Postfix replaceability:** A privilege  $p$  of a new policy  $P.F$  can postfix replace a corresponding privilege  $p$  in the old policy  $P.I$  if the following condition holds:

$$\tau(p)^{P.F} \subseteq \tau(p)^{P.I}$$

If the set of to-be-disclosed credential requisites at the privilege  $p$  in the new policy  $P.F$  is a subset of the set of corresponding requisites in the old policy  $P.I$ , the privilege  $p$  is postfix replaceable.

#### 4. IMPLEMENTATION

This section describes the implementation of our tool for identifying replaceability classes and managing migration of negotiations.

##### 4.1 Database model

We have developed a database to store all information of polices such as states, transitions, roles and privileges, negotiation history of requesters including past states, current state, satisfied transitions, accumulated roles and privileges. A database is particularly helpful in managing policies, negotiation instances, violation analysis and migration due to the following reasons. First, databases are designed for large data which can be thousands of requesters and complex polices with many states and transitions. Second, powerful queries of databases are suitable for analysing details of polices and requesters to migrate instances in minimized amount of time.

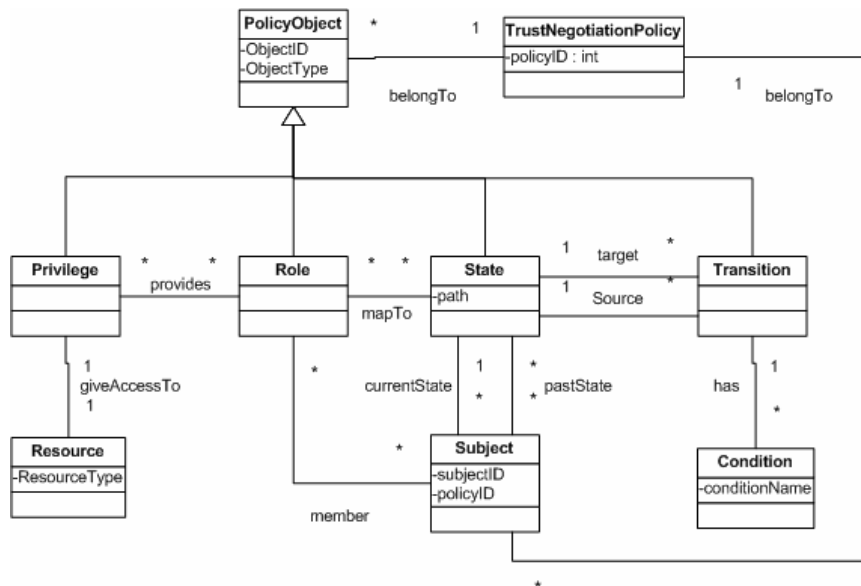


Figure 4.1: Database model

The credential requisites for each privilege are stored at the PrivilegeCondition table as Table 4.1. They are broken into individual disjunctions, each of which is given an index. For example, if  $\gamma(\text{Priv1}) = \{C1, C2\} \vee \{C3\}$  then it will be presented as  $\gamma(\text{Priv1})_0 = (C1, C2)$  and  $\gamma(\text{Priv1})_1 = C3$ .

Table 4.1 Credential requisites of privilege Priv1

Privilege ID	Index	Credential	Policy
Priv1	0	C1	P.I

Priv1	0	C2	P.I
Priv1	1	C3	P.I

We apply the definitions mentioned in Section 3.3 to perform replaceability analysis with the support of the database.

#### 4.2 Performance evaluation

We tested the efficiency of our credential requisite approach by taking complete analysis involving total replaceability, prefix replaceability (static and dynamic methods) and postfix replaceability in an increasing number of ongoing negotiation instances from 10, 100, 1000, 10'000 to 100'000. Details of the process are as following:

1. If the new policy cannot totally replace the old one, first find which states are postfix replaceable. Categorize all instances belonging to those states as postfix replaceable.
2. Use static approach to find which states are prefix replaceable. Categorize all instances belonging to those states as prefix replaceable.
3. For the rest of the instances, individually examine each instance using dynamic approach. Those who satisfy the requirements in the new policy are categorized as prefix replaceable.
4. The remaining instances are categorized as undefined.

The analysis was carried out with a complex policy which is consisted of 12 states and 22 transitions. At this stage, we only created one role per state and one privilege per role. More complex roles and privileges will be added in the future. We also wrote a program that could create any number of negotiation instances at various states of the policy, including information about previously submitted credentials. The testing environment was Windows XP SP1a in a computer with Intel Pentium 4 1.6 GHz CPU and 512 MB RAM. The DBMS used was PostgreSQL version 8.1. Changes were made so that the numbers of instances falling into each replaceability class are similarly equal. Results are displayed in Figure 4.2. The x axis represents the number of active negotiation instances, while the y axis shows the time taken by the various steps of the identification process above.

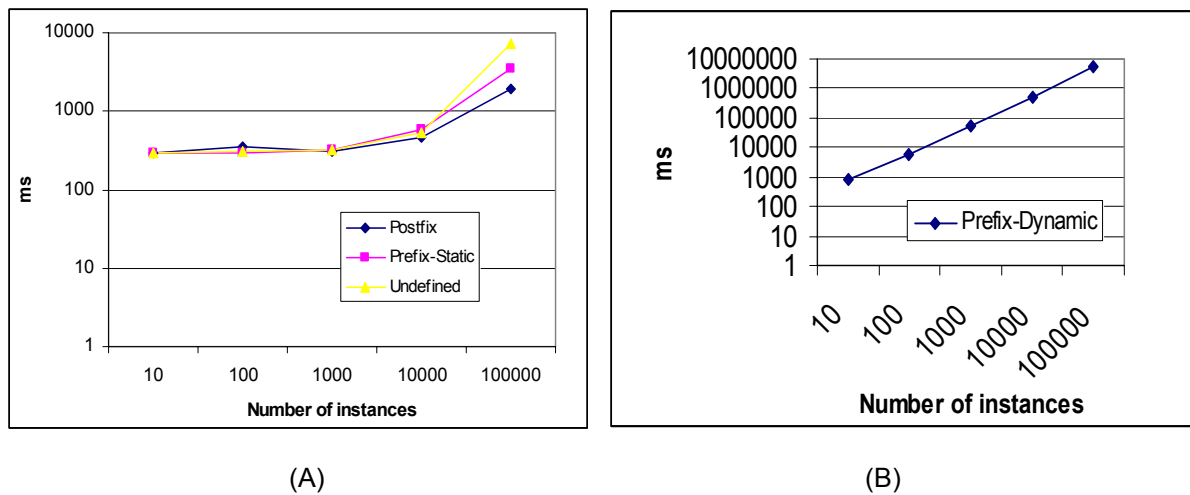


Figure 4.2: (A) Testing results for the analysis of postfix, prefix with static approach and undefined.

(B) Testing result results for prefix analysis with dynamic approach

The running time of each analysis was measured in millisecond (note that the scales are all logarithmic). The testing result showed a good performance of the credential requisite approach. For all static approaches including postfix replaceable, prefix replaceable with static analysis and undefined instances, the running time changes are fairly constant until about 10'000 instances. This is expected, since static analysis only compared the policies and does not look at the instances. There is an increase for those analyses with 100'000 instances but this could be because the time taken to count the number of instances identified at each step became more time-consuming. The

dynamic approach of prefix replaceability analysis rises linearly with the number of instances. This also expected, since the dynamic approach looks at each instance.

The testing results suggested that the static analysis is applicable even when there are a huge number of ongoing negotiations since the running time is constant. The dynamic analysis identifies more instances as replaceable than just static analysis, but it may not always be possible to perform when the number of negotiations is particularly high, since the time taken could inconvenience the requester. This is a trade off between accuracy and performance. Therefore, the dynamic analysis should be taken place after all static analyses are carried out to minimize the number of negotiations needed to be processed.

#### 4.3 Management Toolset

We are now developing a visual interface application aiming at providing simplicity and efficiency for service administrators to manage policies and negotiation requesters. In reality, the Trust-Serv framework should be friendly to non-professionals who might not necessarily know a lot about programming. The GUI application will help by simplifying complex operations into interactions with a friendly user interface. The results of those operations will be arranged and displayed in a highly readable layout. In details, using the application, an administrator will be able to:

- View the current policy and the new policy that negotiation instances would be migrated into.
- View active negotiation instances.
- Choose a particular instance and view its current position and negotiation history (including passed states and transitions, accumulated roles and privileges).
- Highlight the path in the policy figure that an instance took.
- Categorize this particular instance to appropriate replaceability class.
- Perform a complete analysis of replaceability and display the results.
- Highlight all instances belonging to a certain replaceability class.
- Migrate one or all instances of a particular replaceability class into the new policy.
- Show the differences of a particular instance before and after migrating, such as which privileges it has gained or lost.

The application will connect to the database and display into the screen the results of a variety of PL/pgSQL functions that have been already implemented. Some of those functions are viewing a requester's negotiation history, examining prefix and postfix replaceability of a state or migrating a requester from one policy to another. Visual aids like highlighting the path that a requester took in an image of a corresponding policy will also be implemented.

#### 5. CONCLUSION

In this paper, we have implemented a method for analysing the impacts of policy changes to trust negotiation policies on ongoing trust negotiations. We have carried out the performance evaluation of the method using a database, and found performance to be as expected. We are now developing the management toolset to support administrators in managing policies and negotiations. It is expected to be completed at the end of November.

#### REFERENCES

- [BFIK00] M. Blaze, J. Feigenbaum, J. Ioannidis and A. Keromytis, The KeyNote Trust-Management System, <http://www.cis.upenn.edu/~angelos/keynote.html>.
- [HBC04] H. Skogsrud, B. Benatallah and F. Casati, Trust-Serv: Model-Driven Lifecycle Management of Trust Negotiation Policies for Web Services, *Proc. 13<sup>th</sup> Int'l World Wide Web Conf. (WWW2004)*, New York, ACM Press, pp. 53-62
- [HBCF04] H. Skogsrud, B. Benatallah, F. Casati and F. Toumani, Managing Dynamic Changes in Trust Negotiation Protocols for Web Services.
- [HMNR00] A Herzberg, Y Mass, J. Michaeli, D. Naor and Y. Ravid, Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Stranger, *Security & Privacy 2000*.

- [V01] V. Vasudevan, A Web Service Primer, 2001,  
<http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/>.
- [WY02] M. Winslett, T. Yu, K. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith and L. Yu,  
Negotiating Trust on the Web, *IEEE Internet Computing Nov-Dec 2002*.