A Requirements-Driven Trust Framework for Secure Interoperation in Open Environments

Suroop Mohan Chandran, Korporn Panyim, James B. D. Joshi

Department of Information Sciences and Telecommunications
University of Pittsburgh
smc44@pitt.edu, kop1@pitt.edu, jjoshi@mail.sis.pitt.edu

Abstract. *A key challenge in emerging multi-domain open environments is the need to establish trust-based, loosely coupled partnerships between previously unknown domains. An efficient trust framework is essential to facilitate trust negotiation based on the service requirements of the partner domains. While several trust mechanisms have been proposed, none address the issue of integrating the trust mechanisms with the process of integrating access control policies of partner domains to facilitate secure interoperation. In this paper, we propose a requirements-driven trust framework for secure interoperation in open environments. Our framework tightly integrates game-theory based trust negotiation with service negotiation, and policy mapping to ensure secure interoperation.*

1. Introduction

In emerging application evnvironments, loosely coupled entities typically collaborate to provide unified solutions. This has led to the development of service-based applications like Web Services, P2P and Grid applications. Business organizations and commercial entities are now moving towards service-based applications to provide integrated solutions with reusable components [21]. The components themselves may be distributed and only Internet-accessible [22]. Typically, services distribution is managed in a centralized manner, either through some service-broker or some public directory [23]. Typically, in such cases, even trust establishment and management is centralized. But with emerging applications, service requirement specification and provision requires a distributed framework. In such cases, recognizing service requirements and composing services that can satisfy these requirements, becomes quite complex. Furthermore, establishing secure interoperation is crucial because of the variety of requirements and the possibility of many domains interoperating in a collaborative framework. Establishment of trust in such environments is the first significant step to establishing secure interoperation. Trust must be negotiated to satisfy the security requirements of all the domains involved. This is done by the disclosure of sensitive information such as credentials, policies, context of service use etc. A trust framework should address all of the above issues.

Several trust negotiation mechanisms have been proposed in the literature including *Trust-Serv* [1], *TrustBuilder* [2], *H-Trust* [4], *Trust-X* [3] and others [5, 6, 7, 8, 9]. Earlier work has addressed the issue of trust negotiation and trust establishment separately. But none of these frameworks have used negotiation and trust computation together. The level of trust to be established is inherently linked to service requirements. These methods fail in the following aspects: (1) primarily based on the client-server interaction model, (2) based on credential exchange and do not handle credential types, and (3) do not consider service requirements as a factor in trust negotiation or establishment.

In this paper, we propose a requirements-based trust framework to support integrated trust and service negotiation, policy mapping, and a ticketing mechanism for fast cross domain accesses. The proposed framework includes the trust sustenance and evolution components. Following are the key contributions of the paper:

- Trust negotiation is driven by service requirements. It supports bi-directional negotiation of service and context requirements.
- Trust negotiation involves establishing agreeable *trust levels* and *trust token types* to facilitate mapping of policy elements for secure interoperation. Once negotiation is done, *trust tokens* are used for authentication and *trust tickets* are generated to support fast authorized accesses for agreed-upon services under the given context.

The rest of this paper is organized as follows: In section 2, we present related work. In section 3, we present the proposed trust framework. In section 4, the details of service and trust negotiation are presented. In section 5, we discuss the issues behind trust sustenance and evolution and some naïve solutions to the problem.

2. Related Work

The notion of trust among interoperating domains has been loosely divided into two types– negotiation of trust based on credentials and establishing trust based on peer-measured values such as reputation and ranking. Existing work on trust negotiation focuses on the negotiation of credentials, with little focus on the generic requirements of secure interoperation, such as in *Trust-Serv* [1], *TrustBuilder* [2], *H-Trust* [4], *Trust-X* [3], and others [5, 6, 7, 8, 9]. *Trust-Serv* is a model-driven framework that uses state machines to represent and determine credential exchanges for access to resources [1]. Both *TrustBuilder* and *Trust-X* use credential disclosure trees and negotiation strategies to facilitate protection of credential information during negotiation. *TrustBuilder* defines families of disclosure trees to facilitate negotiation between entities that have different disclosure trees for the same resource [2]. The *Trust-X* system introduces the notion of *trust ticket* for efficient negotiation [3], which has been adopted in our framework.

Decentralized systems typically use trust negotiation based on peer reviews and reputations. *HTrust* defines functions to establish, sustain and evolve trust based on entity behavior history [4]. Work in [5] defines a trust establishment and sustenance framework for peer to peer systems using reputation as a basis for trust establishment. Reputation is

distributed across peers through the formation of peer grids or p-grids. The notion of sustenance is based on the concept of complaints, where peers can make complaints regarding other peers to reduce their rank among other peers [7]. A certainty factor can be calculated to quantify the belief (and/or disbelief) a peer has on another peer [6]. Another reputation-based model calculates the reputation for every session based on the number of authentic responses to a query, where authentic responses are defined as original documents matching the query [12]. An approach similar to the reputation-based approaches is taken by [13] for Grid systems, where a trust index is calculated using fuzzy logic, based on the success rate of a job and the defense capability of the domain. A trust index is also calculated as a function of the direct relationship between the domains and the reputation of the target domain. The direct trust relationship is itself a function of the trust level assigned by the domain through interactions and the temporal decay of that trust level [14]. A privacy-enhanced reputation based method can be used to attach a trust value to an entity based on certain events, but these events cannot be traced back from the trust value [15]. A hybrid approach can also be taken, as in [16], where reputation and negotiation is mixed by negotiation of trust tokens between the interoperating domains and the domains confirming the trustworthiness of these tokens through security/trust agents. Similar to the reputation approach is the recommender approach [17, 18].

These systems do not satisfy all the requirements for peer-to-peer trust negotiation and also are not flexible in terms of their credential exchange technique. Our framework is suited for a distributed environment where trust is negotiated based on the service requirements of each domain involved. We introduce trust token types (discussed in Section 4), for establishing a generic security requirement, but still allow negotiation of trust based on the acceptability of different trust token types. Further, we also consider negotiation of trust integrated with service negotiation, such that different trust levels are established for different services exchanged. Trust levels are computed based on a variety of direct and indirect factors, which we shall discuss in Section 4.

3. The Proposed Requirements-Driven Trust Framework

The proposed trust framework, as shown in Figure 1, is composed of two principle modules – the requirements-based *Trust Establishment* (TE) module and the Trust Sustenance and Evolution (TSE) module, which are briefly overviewed below.

3.1 Requirements-based Trust Establishment

Trust establishment involves establishing the services that will be exchanged between the interoperating domains and establishing a negotiated trust level for service access.

Service/Context Negotiation. A service requesting domain will publish its requirements, but it is not necessary that there exists a domain that can exactly satisfy

these requirements. Even if there is one, it may not be able to provide them all. Under such circumstances, services and their contexts may need to be negotiated to converge on a set of service requirements that can be satisfactorily provided by the other domain.

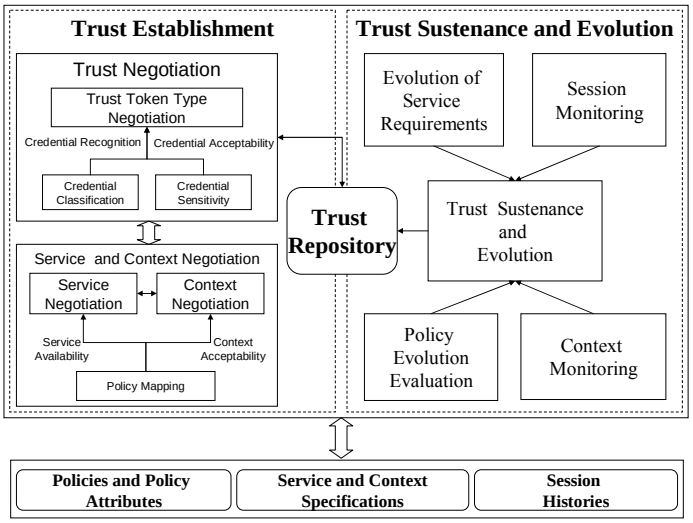


Fig. 1. The proposed Requirements-driven Trust Framework

Trust Negotiation. Trust negotiation involves negotiation of the set of trust tokens that need to be disclosed based on the trust token type required for service access. *Trust token types* are sets of attributes and their allowed range of values, while *trust tokens* represents any set of digital certificates that collectively can show that all the *TT* attributes have values from the specific range. For instance, a trust token type may indicate the requirement for proof of *age* to be above 18. Digital credentials that form valid trust tokens may include *Passport*, *university ID* or *Driver's License*. The negotiation phase establishes which of these credentials could be used as trust tokens. Note that credential certificates used as tokens may have attributes with varying *protection requirements*.

A key result of the negotiation and trust establishment phase is the mapping of the policies in domains if each provides a service to the other, or within the provider domain. Our proposed trust framework assumes that the individual domains employ GTRBAC policies. The fine-grained service requests are represented as a set of abstract permissions that a particular role within the requesting domain needs to access in the provider domain. Our preliminary work on integration of GTRBAC policies reported in [26] is currently used in the proposed framework. The policy mapping facilitates mapping in presence of timing constraints and hybrid hierarchies. We have also extended the GTRBAC model for location-based access control, in LoT-RBAC [28], and the same policy mapping

techniques now be used for secure interoperation in mobile environments as well. A brief overview of the policy mapping process is presented in Section 4.1.

3.2 Trust Sustenance and Evolution

Trust sustenance refers to maintaining trust levels when domain characteristics change during the period of interoperation. Trust evolution refers to the change in trust levels because of changes in domain characteristics.

Evolution of Service Requirements. During a session, a new service requirement can arise or some services may no longer be required. Since trust is requirements-based, evolution of service requirements may trigger a decision on whether to sustain the trust value or re-evaluate, or even renegotiate. Changes in trust values could also be used to renegotiate services; for instance, to reduce the set of accesses given originally.

Context Monitoring. In highly dynamic environments, context changes are inevitable. Since trust levels are context-dependant, it is important to monitor the changes in the context and consequently sustain or calculate the changes to the trust level.

Policy Evolution Evaluation. Changes in policies could cause service usage/provision to be affected (like change in contextual constraints on services), leading to either trust re-evaluation or re-negotiation. Policy mapping will be particularly affected.

Session Monitoring. Anomalous and malicious behavior should be tracked and immediately recognized, so that trust levels can be changed based on the behavior of the other domain. This is a run-time decision on trust sustenance or evolution.

Trust sustenance is usually associated with changes in domain characteristics that are not very significant and can be handled to gracefully end interoperation. Some examples of these changes are change of context, policy changes causing conflict in access resolution, etc. Trust evolution is usually associated with more significant changes, like complete change of context, or access to highly sensitive information. In such cases, trust threshold levels are recomputed and if necessary, trust is renegotiated.

4. Requirements-based Trust Establishment

A distinct feature of our framework is the negotiation and establishment of trust based on the service requirements of the interoperating domains. Next, we briefly discuss how service requests are made and the need for policy mapping for service negotiation.

4.1 Service Requests and Policy Mapping

Typically, service requests are made by member entities of a domain (like users that have assumed certain roles). The requests are usually access to resources and can have a

context associated with them. We assume abstract permissions. Following definition captures the generalization of a service request [26]:

***Definition* 1 (Service Request):** *A requesting domain d_x's service request is defined as:*

$$d_x.SR = <\{r_1, (P_{11}, C_{11}), ..., (P_{1n}, C_{1n})\}, \ldots, \{r_n, (P_{n1}, C_{n1}), ..., (P_{nn}, C_{nn})\}>$$

where r_i is a role in domain d_x, P_{ij} is the j^{th} permission set requested by r_i in context C_{ij}

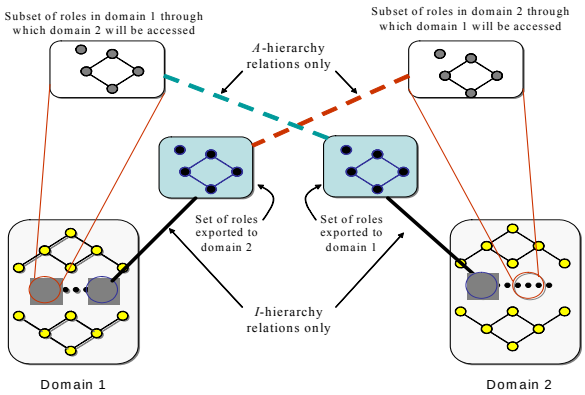


Fig. 2. Role Mapping and Secure Interoperation in GTRBAC-based systems [26]

The roles $\{r_1, .., r_n\}$ may or may not be regular roles in the domain but could also be special roles created by the local policy for interoperation management. The service provider domain will then determine if the service can be provided by doing a preliminary policy mapping, where roles $\{r_1, \ldots, r_n\}$ are mapped to some local roles for access to the requested permissions. The mapping is done by looking up which roles in the role hierarchy are authorized for the requested permissions. Based on hierarchy structures and permission-role assignments, roles may be exported for use by other domains as such or by creating temporary roles in the hierarchy. Export roles are created specifically for the purpose of interoperation. For details on policy mapping for secure interoperation please refer to [26]. with explanation of the use of Inheritance-only, Activation-only and Inheritance and Figure 2 shows how policy mapping is done in GTRBAC-based systems.

4.2 Services and Trust

In general, the interoperating domains try to negotiate what services they require and can provide, in order to match each other's service requirements. If any domain provides services worth less than it received, then it can pay some incentive to the domain that provided more services. Such service requirements-driven service negotiation can be seen in practical applications and should be facilitated to support ad hoc partnerships between a pair of domains. Various cost factors may play a significant role as to how the negotiation may proceed.

***Definition* 2** (**Service Negotiation Parameters**): *Let d_x and d_y be service domains such that services requested by each are satisfied by the other after negotiation. Then we define the parameters for negotiation as shown in Table* 1.

***Definition* 3** (**Service Negotiation Convergence**): *We say that the negotiation between d_x and d_y converges when the following condition holds for both d_x and d_y:*

$$c \leq b + i$$

Table 1. Cost parameters for trust negotiation

$m_{d_x}^{d_y}$	*Cost incurred to d_y for policy mapping, to satisfy requirements of d_x (d_x.SR)*
$r_{d_x}^{d_y}$	*Cost incurred to d_y for resources used by d_x when using services provided by d_y (as per d_x.SR)*
$i_{d_x}^{d_y}$	*Incentives that d_y may receive (or lose) in the interoperation*
$c_{d_x}^{d_y}$	*Cost incurred by d_y for providing services to satisfy d_x.SR: $c_{d_x}^{d_y} = m_{d_x}^{d_y} + r_{d_x}^{d_y}$*
$b_{d_x}^{d_y}$	*Benefits for d_y when using service provided by d_x (as per d_y.SR)*

Ideally, the cost incurred to a domain during interoperation should be less than the benefits and incentives it gets. Note that the condition for convergence may never occur as internal constraints on the services required or provided may restrict further negotiation. In such a situation, secure and desirable interoperation may not be possible.

Trust negotiation is carried out simultaneously with service negotiation to enable establishment of interoperation. Typically, if two domains (say d_x and d_y) are involved in interoperation through exchange of services, each domain requests the other to disclose some information of a certain type as proof of trustworthiness. We introduce the notion of *trust token type* that indicates a set of attributes and the range of values they should be constrained to. Formally we define them as follows:

***Definition* 4** (**Trust Token Type, Trust Token**): *Let TT and T denote a trust token type and a trust token, respectively. Further, let* $A=\{a_1,\ldots,a_n\}$ be a generic set of attributes, *Dom* (a_i) be the evaluation domain of attribute a_i, and $A_1 \subseteq A$. *Then,*

- $TT = (A_1, VS)$, *where* $VS=\{V_1,\ldots,V_{|A|}\}$ such that $V_i \subseteq Dom(a_i)$.
- $T = (A_1, V)$, *where* $v_i \in V$ *is such that* $v_i \in V_i \subseteq Dom(a_i)$ $(i = 1..|A|)$;

Further, a trust token T is said to satisfy *a trust token type TT* (*denoted as* $T \equiv TT$*) if the following conditions hold:*

- $\forall a_i \in TT.A,\ V_i \in TT.VS,\ [v_i \in T.V_i \wedge v_i \in V_i]$

The service-provider domains demand the disclosure of credentials that verify a set of trust token types. Some typical examples of trust token types are ({*age*}, {`greater than` 18}) and ({*nationality, residence*}, {*US and US Minor Islands*, *Pennsylvania*}). Credentials are digitally signed endorsements of some attributes of an entity. They are basically attribute certificates, as specified in [27]. A trust token is constructed by selecting a set of *candidate credentials* that collectively satisfy the trust token type. It is possible that only a subset of the attributes endorsed by each credential is needed to satisfy the trust token type. Formally a trust token can be defined as follows:

Definition 5 (Certificates for Trust Token): *Let TT be a trust token type, CA_i be certification authority, and $C=\{Cert_{CA_1}(A_1), ..., Cert_{CA_n}(A_n)\}$ be such that*

- *each element of C at least has one unique a $\in$ TT.A*
- *the attribute set over all elements of $C \subseteq$ TT.A.*

Then $C_{CA}^{TT.A}$ represents a trust token generated by projecting over attribute set TT.A. of C and then certified by CA. If $C_{CA}^{TT.A} \equiv$ TT, then $C_{CA}^{TT.A}$ is a valid trust token for TT. Note that $n = 1$ is possible in which case the certificate either exactly represents a trust token or a projection over its attributes is needed to generate a trust token

As per the definition, a trust token may need to be generated dynamically to satisfy the required trust token type. The requesting domain may decide to generate such an on-the-fly trust token using the credentials he has by creating a third party certified certificate (CA is a third party). In such a case trust factor will relate to who certifies the trust toke. For instance, a military personnel may have certificates given to him by the military department and may contain many sensitive attributes and while interacting with a private agency may decide to have the military agency certify his token to satisfy the trust token type required by the public agency. It is possible that the CA is the provider himself. In such a case, to satisfy the trust token type, the requester may simply submit a set of credential certificates. An issue here is the protection requirements of the attributes in the certificates that are not required. Exposure of such is a risk that the requester may take based on the trust that it has on the provider and should be incorporated in the trust computation. For the military personnel in our example earlier, exposure of such attributes to the private agency may not be an option at all.

Trust Factors. Prior to negotiation, the interoperating domains also compute $tr_{S,C}^{d_y \to d_x}$, which denotes the trust d_x has with regards to d_y for services defined by S in contexts C. As we shall see later in this section, this is a value that is used to compute the payoff of a negotiation strategy. The computation of the overall trust values is the weighted sum of the *recommended* trust and *direct* trust values [14]. It is possible that a domain does not have both these values for another domain. The direct trust variables are *historical satisfaction level* (*h*) and *risk* (*rk*). Here, *h* indicates the cumulative level of satisfaction that a domain has had for another domain on their previous interactions and is computed based on session histories and older *h* values. Variable *rk* captures risks associated with the desired interoperation. An example is the risk of too many claimed trust tokens being invalid. Another risk is that of services promised but not provided. The historical satisfaction level is also affected by the result of the verification of trust tokens in the earlier sessions. That is, if a domain presents valid trust tokens, then in interoperation, during actual cross domain accesses, the historical satisfaction level will not be negatively affected. The sustenance of the direct trust is based on a family of functions, and can typically be a time-decaying value [14]. Recommended trust is determined by the recommendation value $r_{S,C}^{d_R \to d_y}$ and the trust level for the recommender [16, 17, 20],

denoted by $tr_{S,C}^{d_x \to d_y}$, where d_R is the recommending domain, and d_R is the recommender. Recommended trust can also be a result of a chain of recommendations, where each recommender assigns a trust value for the previous recommender [16].

The parameters that affect the trust relationship are context and the service specifications. Earlier works have found the dependence of trust on contextual parameters like time and location [14, 19]. With respect to temporal context, it is different from time decay of trust, because time decay only shows trust value changing over some time, while temporal context for trust refers to the trust levels at different instances of time. Trust is also specific to service specifications for a particular session – for different services being provided (or requested) the trust levels may be different

***Definition* 6** (**Trust Level**): *Let S and C be the services provided by d_y and the corresponding contexts of interoperation. The trust level $tr_{S,C}^{d_y \to d_x}$ that d_y has on d_x, for services S in contexts C, is defined in Table 2 follows.*

Table 2. Trust level computation

$tr_{S,C}^{d_y \to d_x} = (\alpha \times dtr_{S,C}^{d_y \to d_x}) + (\beta \times rtr_{S,C}^{d_y \to d_x})$	• $\alpha, \beta, \gamma, \delta, \psi, \lambda$ and ε are weights • α is typically greater than β, as direct trust is usually more influential than recommended trust. • Very often α is a result of a time-decay function which represents the degradation in the trust for a domain, due to the lack of interaction. • $h_{S,C}^{d_y \to d_x}$ is the historical satisfaction level that d_y has for d_x • $h_{S,C}^{d_y \to d_x}$ is bound by the previous risk levels as follows: $h_{S,C}^{d_y \to d_x} = \eta \times rk_{S,C}^{d_y \to d_x}$, where $0 \le \eta \le 1$ • $rk_{S,C}^{d_y \to d_x}$ is the *risk* • $r_{S,C}^{d_R \to d_x}$ is the recommendation given by d_R for domain d_x.
$dtr_{S,C}^{d_y \to d_x} = (\gamma \times h_{S,C}^{d_y \to d_x}) - (\delta \times rk_{S,C}^{d_y \to d_x})$	
$rtr_{S,C}^{d_y \to d_x} = (\psi \times tr_{R,C}^{d_y \to d_R}) + (\lambda \times r_{S,C}^{d_R \to d_x})$	

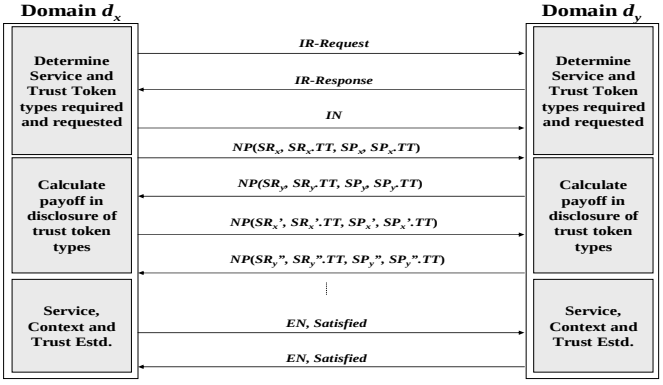
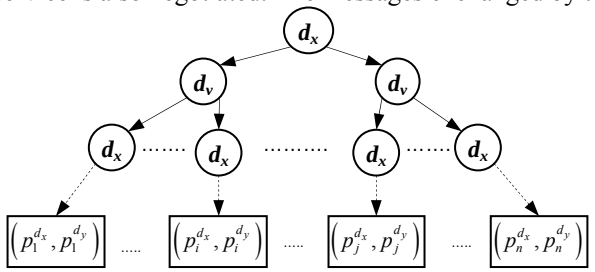


Fig. 3. Protocol for Service, Context and Trust Negotiation

$rk_{S,C}^{d_y \to d_x}$ is a complex parameter with a simple quantification done by computing a value from previous validations of trust tokens of the same type from the same domain. $tr_{S,C}^{d_y \to d_x}$ is computed for two purposes – (*i*) primarily to compute the payoff that is determined for each negotiation strategy, described later in this section; or (*ii*) to set a threshold (minimum) level on the trust that a domain must establish with the other. This facilitates trust token negotiation as well.

4.3 Negotiation Protocol

Negotiation between the domains is done to determine the services required/available and to establish trust, based on the trust tokens. Negotiation of services and associated trust tokens is done simultaneously as can be seen from Figure 3, which describes a protocol for negotiation of services and trust tokens. Note that, simultaneously, even context of service is also negotiated. The messages exchanged by the domains are given in Table 3.



- Negotiation Tree = {*V, E*} *V*={*Root*, *Non-Leafs*, *Leafs*}
- The domains alternate every level of the tree.
- *Root*: Requesting Domain
- *Edges*: Strategy execution by a domain at the previous level
- *Non-Leafs*: State of Negotiation after previous domain's strategy
- *Leaf Nodes*: Payoff for a sequence of strategies

Fig 4. The negotiation tree

Table 3. Message Description for Trust Negotiation

Message	Syntax and Description
Interoperation Request/Response **(IR)**	<**IR**, *Required* (*or Provided*), *Name, Service*, *Context*>
	Such messages are sent by the initiator domain and the responder domains
Initiate Negotiation (**IN**)	<**IN**, Accept>
	This is a message sent by the initiator to the domain(s) which it has selected from a set of domains that responded to its request, to start negotiation of services, context of service and trust token types required.
Negotiation Proposal (**NP**)	**<NP**, *Name, SR, SR.C, SR.TT, Sp, SP.C, SP.TT*>
	The negotiation messages exchanged between the domains
End Negotiation **(EN)**	**<EN**, *Satisfied* (*or Not Satisfied*)>
	This message is sent to end the negotiation either in satisfaction or disapproval

To determine the convergence point of the negotiation, we take the game-theoretic approach of defining payoffs for different strategies. Here trust tokens are strategies, and each trust token has a different overall protection requirement. Based on the choice of trust tokens for disclosure, corresponding domains have gains (or losses). The payoff for each domain is the linear sum of the payoffs from services and trust token negotiations respectively.

The trust token negotiation payoff is the difference between the trust level established and the protection level required of the trust tokens disclosed, as given below:

$$\phi'_{ij}(p_i^{d_x}, p_j^{d_y}) = ((tr_{S,C}^{d_x \to d_y} - ProtLevel(d_x.T_i)), (tr_{S,C}^{d_y \to d_x} - ProtLevel(d_y.T_j))),$$

The service negotiation payoff is the difference between the benefits from usage of services and the losses incurred through service exchange and service provision.

$$\phi''_{ij}(p_i^{d_x}, p_j^{d_y}) = (b_{d_y}^{d_x} - c_{d_y}^{d_x} - i_{d_y}^{d_x}, b_{d_x}^{d_y} - c_{d_x}^{d_y} - i_{d_x}^{d_y})$$

Thus the overall negotiation payoff is given as:

$$\phi_{ij}(p_i^{d_x}, p_j^{d_y}) = \phi'_{ij}(p_i^{d_x}, p_j^{d_y}) + \phi''_{ij}(p_i^{d_x}, p_j^{d_y})$$

The negotiation is essentially modeled as a negotiation tree. The different strategies used by the domains are the disclosure of different trust tokens that satisfy the other domain's requirements but have different protection requirements. It is reasonable to assume that protection requirement of a trust token is directly related to trust level desired. For instance, a passport is a more trustworthy proof of age, but it also contains more sensitive details. Traversal of the tree represents negotiation exchanges between the domains. Each domain computes the payoffs at the leaf nodes and selects a set of *candidate payoffs*. Using a goal-driven approach (goal being any of the candidate payoffs), the domains negotiate the payoffs. Ideally, both domains select the same candidate payoffs, because in game-theory-based negotiation, strategies are selected that optimize payoff for both parties. The candidate payoff values are selected through empirical studies. Consequently, backtracking is also facilitated in the negotiation – if say d_y proposes a set of services and trust tokens that would lead to poor payoff for say d_x, then d_x will reject the proposal and d_y will have to go back and try another proposal. The negotiation tree structure is given in Figure 4.

Figure 5 shows the flowcharts for service and trust token negotiation, for both atheservice requester and provider. For service provision, the domain checks the availability of those services before determining the trust token type(s) required for each service. The domain may reject the request message if required service is not available. If the requested service is available, the domain determines the trust token types required. The domain grants interoperation of requested services if the trust tokens, claimed to match the trust token types, are satisfactory, otherwise it determines a new set of trust tokens required for the next round of negotiation. For service requests, the domain checks if the set of services from the provider is enough. If so, then the domain checks the availability of trust tokens matching the trust token type requested from the other domain. The domain may reject the service request, if it does not possess trust tokens of the requested type. Otherwise, it determines the set of trust tokens to disclose, that has the

best payoff for both domains. We believe that although the open environment is assumed, most trust-based relations may be established well before there is any access of resources.

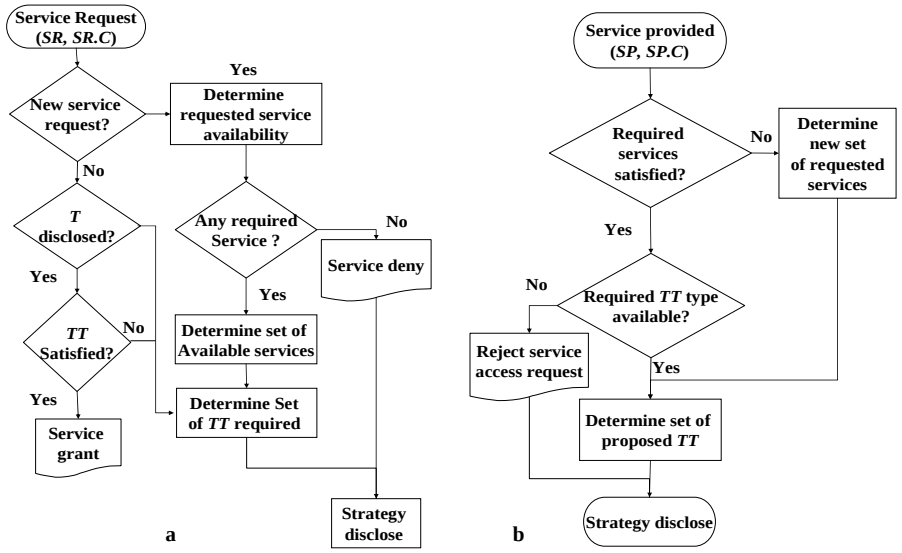


Fig. 5 a. Service and Trust Token Disclosure for Service Provider; b. Service and Trust Token Disclosure for Service Requester

The time between the trust establishment and resource access can be long enough to make some trust-tokens become invalid. In earlier systems, this would lead to renegotiation of credentials [1, 2, 3]. But in our model, we would only renegotiate the one trust-token type in case the peer might actually have a trust token with different protection requirements associated with credential attributes. Thus, when the *StudentID* is proved invalid, E_2 asks for another trust token type, and the customer discloses the possession of *StateID* which is then accepted, with the same trust level and same set of privileges.

Trust Ticket. One enhancement to the system is the use of a *trust ticket*. The *trust ticket* can be used to by pass the trust token validation process. By disclosing a trust ticket, a domain can access a set of requested services indicated in a trust ticket. Service provider issues a trust ticket for each successful interoperation. Trust tickets offer the flexibility in future interoperations, since a set of services and context indicated in the ticket may be a part of service requests in other interoperations. The trust ticket issued to service requester is encrypted by an established session key k_s to ensure integrity of the ticket.

Ticket ID	*Services*	*Trust Token ID*	*Ticket Issuer*	*Ticket Holder*	*Lifetime*	*Shared Secret*

Figure 6: Trust Ticket data fields

The data structure of a trust ticket is shown in Figure 6. The detail of the trust ticket is as follow: Trust ticket identifier is stored in *Ticket ID*. The *Ticket Issuer* indicates domain, which issued the ticket and *Ticket Holder* indicates the domain or a specific users that uses the ticket. A set of service identifiers associated to a ticket is specified in *Services*. *Lifetime* is an expiration time of the trust ticket. A random number, *Shared Secret* increases with each of multiple accesses. Validity of the ticket is specified in *Lifetime*. The *Lifetime* indicates the time-interval that the ticket is valid, which is usually not greater than duration of interoperation session. It is essential to ensure that valid duration of trust ticket is no longer than all lifetime of all certificates associated with the ticket.

During subsequent accesses, trust tickets can now be used instead of the trust token which requires credential validation. Once negotiation of services and trust token types succeeds, service provider creates a trust ticket to service requester. Both domains establish a session key k_s for encryption of trust ticket used between both parties. Service provider domain evaluates the trust ticket by checking validity of the trust ticket and all associated certificates. If the ticket and all the certificates are valid, the credential validation process grants access to the requested services without actual credential validation. The trust ticket is encrypted by established shared secret key k_s to guarantee privacy and integrity of the ticket. Requester domain uses *Shared Secret* value as a counter to keep track of number of service accessing by increasing *Shared Secret* value by one each time he accesses the resource

4.4. Implementation

We have implemented a very basic *proof of concept* system to ensure that the framework works. The implementation involves each domain running three Java threads – a Peer, Recommender and Certifier. Peers request services amongst each other and credentials. The proposed negotiation trees are created for the prototype. We are currently working on a full-fledged implementation along with integration with an access-control framework based on the location and time based RBAC model (LoT-RBAC [28]).

5. Conclusions and Future Work

We introduced the notion of requirements-based trust negotiation to induce more effective trust negotiation and establishment. We have used the notion of trust token types to abstract the requirements of a domain to establish trust. Some concepts that we have touched upon in our work (like protection requirements of trust tokens and risk) are out of scope of our discussion, because of which we have not elaborated on their computation. But these are important to the trust negotiation and trust framework, and we are currently exploring methods of good estimations of these values. We have also used the game-theoretic approach for disclosure strategy selection and shown flowcharts for strategy

selection based on payoffs. Computation of the set of potential payoffs is still complex and we are currently working on efficient search and computation algorithms for these. We have also briefly addressed the issues of trust sustenance and evolution, but the decision to perform either under the given conditions is empirically determined. We are currently working on the implementation of this framework and will obtain empirical results for trust evaluation and sustenance.

Acknowledgement: This research has been supported by the US National Science Foundation award IIS-0545912. We thank the anonymous reviewers for their helpful comments.

References

1. Skogsrud, H., Benatallah, B., Casati, F., "Model Driven Trust Negotiation for Web Services", IEEE Internet Computing, November-December 2003, Pages 45-52.
2. Yu, T., Winslett, M., Seamons, K. E., "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation", ACM Transactions in Information Systems Security, Vol. 6, No.1, February 2003, Pages 1-42.
3. Bertino, E., Ferrari, E., Squicciarani, A.C., "Trust-X: A Peer to Peer Framework for Trust Establishment", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 7, July 2004, Pages 827-842
4. Capra, L., "Engineering Human Trust in Mobile System Collaborations", in Proceedings of ACM SIGSOFT/FSE-12, Pages 107-116, Oct 31-Nov 6, 2004, Newport Beach, CA
5. Aberer, K., Despotovic, Z., "Managing Trust in a Peer-2-Peer Information System", in Proceedings of ACM CIKM'01, Pages 310-317, November 5-10, 2001, Atlanta, GA
6. Xianliang, H. M. L., Chuan, Z.-x-Z., "A trust model of P2P system based on confirmation theory", ACM SIGOPS Operating Systems Review**,** Volume 39, Issue 1 (January 2005), Pages: 56 - 62
7. Gupta, M., Judge, P., Ammar, M., "A Reputation System for Peer-to-Peer Networks", in Proceedings of *NOSSDAV'03,* June 1–3, 2003, Monterey, California, USA.
8. Damiani, E., di Vimercati, S. de C., Paraboschi, S., Samarati, P., Violante, F., "A Reputation-Based Approach for Choosing Reliable Resources in Peer-to-Peer Networks", CCS'02, November 18-22, 2002, Washington, DC, USA.
9. Ye, S., Makedon, F., Ford, J., "Collaborative Automated Trust Negotiation in Peer-to-Peer Systems", in Proceedings of the Fourth International Conference on Peer-to-Peer Computing, 2004. 25-27 Aug. 2004 Page(s):108 – 115
10. Khedr, M., Karmouch, A., "Negotiating context Information in Context-Aware Systems", IEEE Intelligent Systems, Volume 19, Issue 6, Nov-Dec 2004 Page(s):21 – 29
11. Ryutov, T., Zhou, L., Neuman, C., Leithead, T., Seamons, K. E., "Adaptive Trust Negotiation and Access Control", in Proceedings of SACMAT 2005, June 1-3, 2005, Stockholm, Sweden, Page(s): 139-146
12. Marti, S., Garcia-Molina, H., "Identity-Crisis: Anonymity vs. Reputation in P2P Systems", in Proceedings of The Third International Conference on Peer-to-Peer Computing, 2003. (P2P 2003). 1-3 Sept. 2003 Page(s):134 - 141

13. Song, S., Hwang, K., Macwan, M., "Fuzzy Trust Integration for Security Enforcement in Grid Computing," in Proceedings of IFIP International Symposium on Network and Parallel Computing (NPC-2004), Wuhan, China. Oct. 18-20, 2004. pp. 9-21.
14. Azzedin, F., Maheswaran, M., "Towards Trust-Aware Resource Management in Grid Computing Systems", in Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02), 18-21 Aug. 2002, Page(s):47 – 54
15. Bussard, L., Roudier, Y., Molva, R., "Untraceable Secret Credentials: Trust Establishment with Privacy", in Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04), 14-17 March 2004 Page(s):122 – 126
16. Au, R., Looi, M., Ashley, P., "Automated Cross-organisational Trust Establishment on Extranets", in Proceedings of Workshop on Information Technology for Virtual Enterprises, 2001. ITVE 2001, 29-30 Jan. 2001 Page(s):3 – 11
17. O'Donovan, J., Smyth, B., "Trust in Recommender Systems", in Proceedings of IUI'05, January 9-12, 2005, San Diego, California, Page(s): 167-174
18. Shand, B., Dimmock, N., Bacon, J., "Trust for Ubiquitous, Transparent Collaboration", Wireless Networks 10, 711-721, 2004, Kluwer Academic Publishers.
19. Manchala, D. W., "E-Commerce Trust Metrics and Models", Internet Computing, IEEE Volume 4, Issue 2, March-April 2000 Page(s):36 - 44
20. Daskapan, S., Vree, W. G., Eldin, A. A., "Trust Metrics for survivable security systems", in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 2003. Volume 4, 5-8 Oct. 2003 Page(s):3128 - 3135
21. Patrick, P., "Impact of SoA on Enterprie Information Archietctures", Proceedings of *SIGMOD 2005*, June 14–16, 2005, Baltimore, Maryland, USA.
22. Benatallah, B., Dumas, M., Fauvet, M.-C., Rabhi, F. A., Sheng, Q.-Z., "Overview of some Patterns for Architecting and Managing services", ACM SIGecom Exchanges, Vol. 3, No. 3, August 2002, Pages 9 -16.
23. Baresi, L., Heckel, R., Thone, S., Varro, D., "Modeling and Validation of Service-Oriented Architectures: Application vs. Style", Proceedings of *ESEC/FSE'03,* September 1–5, 2003, Helsinki, Finland.
24. Joshi, J.B.D., Bhatti, R., Bertino, E., Ghafoor, A., "Access-control language for Multidomain environments", IEEE Internet Computing, Volume 8, Issue 6, Nov.-Dec. 2004 Page(s):40 – 50
25. Joshi, J.B.D.; Bertino, E.; Latif, U.; Ghafoor, A., "A generalized temporal role-based access control model"**,** IEEE Transactions on Knowledge and Data Engineering, Volume 17, Issue 1, Jan 2005 Page(s):4 – 23
26. Piromruen, S., Joshi, J. B. D., "An RBAC Framework for Time Constrained Secure Interoperation in Multi-domain Environment," in Proceedings of IEEE Workshop on Object-oriented Real-time Databases (WORDS-2005), 2005.
27. Farrell, S., Housley, R., "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.
28. Mohan Chandran, S., Joshi, J. B. D., "LoT-RBAC : A Location and Time-based RBAC Model", Proceedings of 6th International Conference on Web Information Systems Engineering, November 20-22, 2005, New York City, NY.