Telecommun Syst (2013) 52:1353–1365
DOI 10.1007/s11235-011-9648-4

HiTrust: building cross-organizational trust relationship based on a hybrid negotiation tree

Jianxin Li · Xudong Liu · Lu Liu · Dazhi Sun · Bo Li

Published online: 20 October 2011

Abstract In a pervasive computing environment, the need to establish trust amongst distributed services has attracted increasing attentions from both the industry and academia. As a widely adopted solution to carry a principal's identity and attributes of different organizations, the credential-based trust establishment has become popular over Internet. In this paper, we propose a hybrid negotiation tree based modeling approach, named HiTrust, to build cross-organizational trust relationship. The HiTrust is used to characterize the gradual interactions state during the trust establishment between the principals from different security organizations. Compared with the original disclosure tree model, the hybrid tree model in HiTrust can embed both policies and credential sets in a tree node, and is able to describe fine-grained security policy with attributes or negotiation context information. This property endows the HiTrust with the capability of describing complex trust establishment requirements, and makes it more efficient to search desired tree node. Furthermore, to enhance the usability and efficiency of negotiation service, we propose a session state maintenance mechanism based on a policy stack and an asynchronous trust chain propagation mechanism. We have implemented the HiTrust prototype system, and experimentally verified that the HiTrust is effective and scalable.

Keywords Information security · Access control · Trust management · Trust negotiation · Policy · Privacy

J. Li (✉) · X. Liu · B. Li
School of Computer Science & Engineering, Beihang University, Beijing, China
e-mail: lijx@act.buaa.edu.cn

B. Li
e-mail: libo@act.buaa.edu.cn

X. Liu
e-mail: liuxd@act.buaa.edu.cn

L. Liu
School of Computing & Mathematics, University of Derby, Derby, UK
e-mail: l.liu@derby.ac.uk

D. Sun
School of Computer & Technology, Tianjin University, Tianjin, China
e-mail: sundazhi@tju.edu.cn

1 Introduction

Pervasive computing environments is a distributed and mobile space [1], and entities may be operating in an unknown context. In a pervasive computing environment, many entities (e.g., services) generally should collaborate to deliver some business services. However, in a dynamic cross-organizational collaboration environment, services involved in a business process are often provided by different organizations (as shown in Fig. 1), and lack supports of common security mechanisms and centralized management middleware [2]. On such occasions, the participating services involved in a pervasive computing environment may collaborate dynamically to achieve business objectives at run time, and a participating service may have to collaborate with multiple participating services which it has no pre-existing knowledge in prior.

However, compared with traditional distributed system, the Internet-based pervasive computing systems are loosely coupled and more dynamic, which bring more challenges to the trust establishment of collaboration. For example, when a mobile client accesses a service provided by another unfamiliar security organization, the trust must be established firstly. Therefore, the dynamic trust establishment

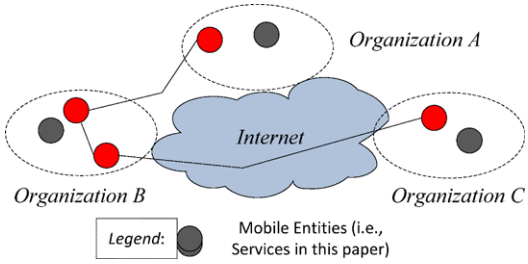

Fig. 1 Cross-organizational collaboration of services in a pervasive computing environment

problem between principals without pre-existing trust relationship becomes a fundamental and crucial problem for cross-organizational collaboration.

Moreover, during the trust establishing process, a more important but difficult problem is how to preserve the privacy of participants' credentials and policies. The traditional centralized authority mechanism is not suitable since it has several limitations. Firstly, it requires users to fully trust service providers. Secondly, there is no guiding mechanism about for which credentials should be submitted. These two limitations make users always blindly disclose all their attribute credentials in order to acquire service authorization, which not only increases network traffic overhead, but also lack of the ability to protect users' sensitive information.

To address such issues, Winsborough et al. proposed an automated trust negotiation (ATN) [3, 4] approach, which aims to gradually build trust relationship between strangers through the interactively disclosing of credentials and security policies. In ATN, the status of participants for service providers and requesters is equivalent, so both sides have the privileges to protect their disclosed information during trust establishment, and a negotiation model is used. Meanwhile, this mechanism can accurately guide users to disclose information to build trust relationship. In ATN, the representative model describing this process is *Negotiation Tree Model,* which is essentially a tree with credentials generated according to security policies of the principals on both sides. However there are some problems limitations as follows:

First, during the process of trust negotiation, only the credential nodes are extracted to generate a credential tree from security policies, while the other information related to negotiation status such as policy structure and fine-grained constraints are lost. In contrast, the security policy (e.g., a policy depicted in XACML) for real applications usually contains two types of useful constraints. One is the *static attributes constraint* for qualified credentials; the other is the *dynamic context constraint* for a negotiation context. Particularly, the function of *dynamic context constraint* can ensure the negotiation success ratio and performance, but be rarely considered in existing negotiation models. For instance, in order to ensure the negotiation quality and resist possible DoS attack, *dynamic context constraints* are usually required to restrict the number of interactions, received credentials, length of credential chain and so on. In this paper, we introduce a hybrid tree model which depicts both policy structure and negotiation context constraints.

Second, in a negotiation tree, a policy will be split into several credential nodes. There are three deficiencies: (1) The original structure of a policy may be destroyed; (2) The number of tree nodes will be increased; (3) The performance of tree searching and matching will be reduced. In particular, to decide whether a policy is satisfied, it must backtrack to its parent node to reconstruct the original policy according to the credentials relationships. Unlike it, HiTrust holds the policy as a whole node and searches it through policy index id or *hash value*, which will improve the efficiency of policy query and compliance verification.

Third, some existing ATN approaches generally assume that the credentials can be found locally, without adequate considerations on how to construct a credential chain. So far, RT [5] is an effective approach to discover credential chain during trust negotiation. In our former study, we have found that the synchronous construction of credential chain is always time-consuming and becomes a major bottleneck problem of negotiation service performance.

To address the above issues, we proposed a hybrid tree based dynamic trust negotiation service named HiTrust. We have made the following contributions.

- First, we designed a hybrid tree model composed of both security policy and credential set nodes to depict the status of negotiation process, and presented an efficient automated hybrid tree construction algorithm.
- Second, we presented an adaptive trust negotiation strategy based on the hybrid tree model, where a tree searching and information disclosure algorithm is given. In this strategy, a policy-stack mechanism is employed to improve efficiency of policy query, and an asynchronous credential chain propagation mechanism is employed to enhance the usability of HiTrust service.
- Last, we have successfully implemented the HiTrust prototype system in the CROWN Grid [21], and comprehensive experimental study shows our approach is scalable and efficient.

The remainder of this paper is organized as follows. We discuss related work in Sect. 2. Section 3 elaborates the concepts and algorithms of hybrid tree model. We introduce some key techniques related to the adaptive negotiation strategy in Sect. 4. The implementation experience of HiTrust is given in Sect. 5. Finally, we conclude the paper in Sect. 6.

2 Related work

In Internet-based pervasive computing environments, the security, privacy and trust of cross-organizational resources

sharing and collaboration have become an important research issue. At present, the protection of sensitive credentials based on access control policy and trust establishment through dynamic negotiation between principals have attracted many research efforts, such as TrustBuilder [6], PeerTrust [7], Trust-X [8], TTG [3], trust negotiation for MAS system [9] and trust negotiation formal model [4].

Winsborough [14] firstly proposed the concept of trust negotiation, describing trust negotiation process as with a credential disclosed sequence. In this sequence, two participated principals credentials are interactively disclosed the credentials and policies by participants. Ting Yu et al. [15] proposed the Disclosure Tree model, in which all nodes belong to credential type except for the root node. All the children nodes of one node compose of a satisfying credential set solution to this node. If there are multiple satisfying credential sets, they must be represented as multiple trees. In [15], authors proved that the disclosure sequence model and disclosure tree model are equivalent. The shortcoming of this model is that the number of trees is changeable, and there is redundant information between different trees. Yu and Bertino et al. [8, 16, 17] use the negotiation tree to express the security policy of both principals, and it is de facto a credential tree. Yu et al. [16] proposed an AND-OR Tree to get a credential disclosure sequence, and provided a negotiation strategy PRUNES to address the complexity problem induced by brute searching in the AND-OR tree. Bertino et al. [8] constructed an AND-OR negotiation tree based on X-NTL policy language to control the credential disclosure. Trust-Serv [18] is a model-driven framework using a state machine model for credential disclosure. Similarly, Chen et al. [19], based on negotiation tree model, provided an optimized credential sequence searching regarding to different sensitive cost. In [20], a meta-policy based information protection mechanism is presented, which also classifies credential sensitive cost in different levels.

Negotiation strategy is also a key component of trust negotiation approach, in which the rules for credentials and policies disclosure are defined to disclose credentials and policies. Winsborough et al. [14] also gave two kinds of negotiation strategies: *eager strategy* and *parsimonious strategy*. However, these strategies are used independently, and also have not concerned the credential chain construction issue.

Compared with the disclosure tree model, the features of the hybrid tree model are as follows: (1) The security policy node is added in the tree, and has the ability to describe more complex negotiation constraints. (2) The credential set node is a minimum credential solution to a security policy, which reduces the number of tree nodes. (3) The tree nodes are of OR logical relation compared with AND/OR tree.

Therefore, HiTrust is an extension to credential tree model, the hybrid tree consists of both policy node and credential set node. HiTrust improves the ability of policy expression, and the relation between credential set and policy becomes clearer, and HiTrust also simplifies the hybrid tree construction algorithm and negotiation strategy algorithm.

Moreover, many recent research efforts for production system are conducted. In [12], authors argue that trust negotiation is ready for a trial deployment in a real-world application. Some recent work has been introduced. TrustBuilder2 framework is for experimenting with trust negotiation runtime systems, the CLOUSEAU compliance checker which can quickly determine whether a set of credentials complies with a particular policy, and the Traust approach lets legacy applications take advantage of trust negotiation. Dragoni, et al. proposed two major research problems [10, 11] on trust negotiation: real-life and dependability problems. The first one concerns an adaptive negotiation strategy according to security risk; the second considers how to provide self-protection and self-healing functions against malicious attacks. Currently, our approach has been used to secure the remote hot deployment service in CROWN testbed.

3 HiTrust: a hybrid tree based trust negotiation service

3.1 Basic concepts

In HiTrust, we adopt an attribute-based authorization mechanism, and a principal can be a user, a system process and so on. *An attribute credential is the carrier of principal's attributes assertion including principal identity, attributes, etc. signed by an Attribute Authorities*. In an attribute credential, there are principal's public key, attribute items, signature and validation information and so on. In HiTrust, we use symbol C with or without subscripts to denote a credential and symbol N with or without subscripts to denote a credential set.

Generally speaking, an attribute credential is issued directly by an Attribute Authority. However Furthermore, some delegation and trust federation technologies are used to multiple organizations collaboration, a principal's attribute may be represented by with a credential chain (e.g., a typical certificate chain), and we uses $C_m \rightarrow C_n \rightarrow C$ to represent a credential issuing sequence. Thereby, a credential chain construction is a procedure that often is used to find some potential attribute assertions that a principal owns.

Security policy is used to enforce access control on resources (including service resources, sensitive resources such as credentials), and specifies which credentials should be shown to prove principal's attributes. Generally, the policy decision procedure takes attribute values and negotiation context information as input and returns a *boolean* value. In HiTrust, *a security policy* (*access control policy*) *on resources or credentials is denoted by symbol* $\mathcal{P}$. *In* $\mathcal{P}$, *it also*

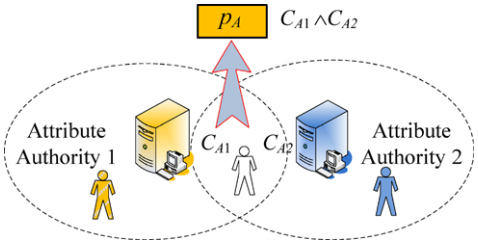

Fig. 2 Relation between attribute credential and the security policy

includes some static attributes constraints and dynamic context constraints. We use $\mathcal{P} \propto R$ to denote that a resource R is protected by a policy $\mathcal{P}$, and $\mathcal{P} \propto N$ to denote that a credential set N is protected by $\mathcal{P}$.

A credential often contains some sensitive attributes, such as *age*, *address* etc., so holders should make corresponding security policy to protect them. A security policy to protect a credential C is denoted by $p_1 \vee \cdots \vee p_n \propto C$, where "$\vee$" means these atomic policies are of logical disjunction (or) relation.

When making the security policy, we usually normalize a security policy $\mathcal{P}$ into a CNF (conjunctive normal form) formula. If a policy $\mathcal{P}$ cannot be further normalized into a DNF (Disjunctive Normal Form) formula, we call it *atomic security policy*, and every atomic policy p takes the following form:

$$p = (C_1 \wedge C_2 \wedge \cdots \wedge C_m) \wedge (\text{constrains on } C_1 \ldots C_m).$$

Therefore, a security policy $\mathcal{P}$ has the following form $\mathcal{P} = p_1 \vee p_2 \vee \cdots \vee p_n$. A credential often contains some sensitive attributes, such as *age*, *address* etc., so holders should make corresponding security policy to protect them. A security policy to protect a credential C is denoted by $p_1 \vee \cdots \vee p_n \propto C$, where "$\vee$" means these atomic policies are of logical disjunction (or) relation.

As shown in Fig. 2, a principal is issued two attribute credentials by two attribute authorities, and these credentials can satisfied the policy p. Sometimes, a principal's attribute credentials are often stored in the attribute authority servers or other servers.

According to this convention, the authorization decision can be made easily with a normalized security policy. For example, as for a security policy $p = (C_1 \wedge C_2) \wedge (C_1.x = \mathsf{vip}) \vee C_3$, this definition mainly contains two parts: (1) Specifying what credentials should be shown, where a credential is referred by a *DN*, such as $C_1 \wedge C_2, C_3$; (2) Providing attribute constraints expression on credentials, e.g., $C_1.x = \mathsf{vip}$ which denotes an equality constraint on attribute x of C_1. During a trust negotiation process, these security policies will be sent to the corresponding negotiator, and at the same time, some negotiation context constraints for this trust establishment session will also be sent, e.g., a constraint $\mathsf{creds.count} \leq 5$ which means the maximum number of credentials allowed to be received.

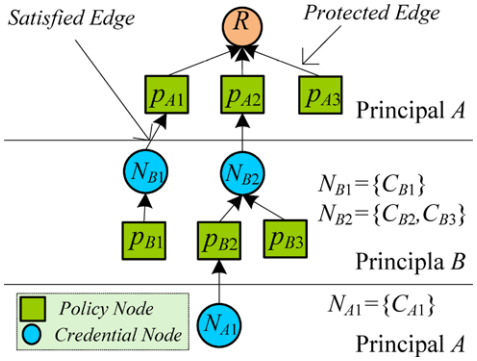

Fig. 3 A hybrid tree example

Definition 1 (Minimum Credential Set Solution of Policy) Let p be an atomic policy within a negotiation session, if there exists a non-empty credential set $N = \{C_1, C_2, \ldots, C_n\}$, taking attribute value in C_k and context information of negotiation session as an input and making security policy p always be true, we call N as a credential set solution of policy p, denoted by $N \rightarrow p$. if $\forall N_1 \subset N, N_1 \neq \varnothing$ there never holds $N_1 \rightarrow p$, then we call N as a minimum credential set solution of policy p, denoted by $N \Rightarrow p$.

3.2 Hybrid tree model

The disclosure tree model splits some policy information and thereby it is inefficient for compliance checking. Instead, HiTrust uses a hybrid tree model to represent the relations between security policies and credential sets in order to control information disclosure during trust establishment in a fine-grained manner.

Definition 2 (Hybrid Tree) A hybrid tree related to the resource $\mathcal{R}$ holds the following properties:

1. The root node represents the resource $\mathcal{R}$;
2. For a credential set node or root node $\mathcal{R}$, if no sensitive credentials are in this set, there is no child node, otherwise its child nodes represent security policies protecting this credential set.
3. For a policy node, each of its child nodes, if has, represents a minimum credential set satisfying this policy.

As shown in Fig. 3, a hybrid tree has three kinds of nodes and two kinds of edges (shown in Table 1). Thus the hybrid tree model can describe all the relations between credential sets and access control policies in one tree, and the relationship nodes to one parent node are of "or" logic relation.

In a hybrid tree, a leaf node may be a type of *Credential Set Node* or *Policy Node*. If it is a *Credential Set Node*, it means that there are no protected credentials in this *Credential Set*. If it is a *Policy Node*, it means that there are no credentials satisfying this security policy. As shown in

Table 1 Policies of principal A and B

Name	Note
Root Node	It represents the requested resource (e.g., a business service),
Credential Set Node	It represents the minimum credential set solution of a security policy
Policy Node	It represents the security policy for resources
Satisfied Edge	It is a relation of $N \Rightarrow p$
Protected Edge	It is a relation of $P \propto N$

Fig. 3, node N_{A1} means C_{A1} can be disclosed freely, while p_{B1}, p_{B3} and p_{A3} are policy leaf nodes, and it means that there is no minimum credential set solution for these three atomic policies.

Based on the requested resource in a session, the credentials (stored in a *credBase* set) and the security policies (stored in a *policyBase* set) of both negotiation participants, a hybrid tree labeled by root node can be built according the following algorithm (Algorithm 1).

Algorithm 1 The Hybrid Tree Building Algorithm
Input: *credsBase*, *policyBase*, requested resource $\mathcal{R}$
Output: A Hybrid Tree labeled by *root*

```
1.  buildHybridTree(credsBase, policyBase, R) {
2.    root = new PoliciesNode(R); //set the root node of the Hybrid Tree
3.    nodeSet.add(root);
4.    while(nodeSet ≠ ∅) { //process each node in a loop
5.      n = nodeSet.next(); //get an element from the node set
6.      if(n == R || n contains sensitive credentials) {
7.        P = getPolicy(policyBase, node); //get protected security policy
8.        for(∀p ∈ P) { //processing every atomic policy
9.          n.addChild(p); //add p as a policy subnode
10.         nodeSet.add(n); } }
11.     else if (n is an atomic security policy ) {
12.       N ← getPolicySolution(credsBase, n); //get the minimum credential set solution of the policy n
13.       n.addChild(N); //add N as a credential set subnode
14.       nodeSet.add(n); }
15.   } //end of while loop
16. return root; }
```

As shown in Table 2, principal A is the provider of resource R, and makes an access control policy as (1), principal B has credentials C_{B1}, C_{B2} and C_{B3}, and B has the

Table 2 Policies of principal A and B

Principal	Policies	Note
A	(1) $p_{A1} \vee p_{A2} \vee p_{A3} \propto R$	security policy for resource
B	(2) $N_{B1} = \{C_{B1}\} \Rightarrow p_{A1}$	satisfied credential sets
	(3) $N_{B2=}\{C_{B2}, C_{B3}\} \Rightarrow p_{A2}$	
	(4) $p_{B1} \propto C_{B1}$	security policy for sensitive credential
	(5) $p_{B2} \vee p_{B3} \propto C_{B2}$	
A	(6) $N_{A1} = \{C_{A1}\} \Rightarrow p_{B2}$	satisfied credential sets

satisfying relations (2)&(3), but no credentials satisfy p_{A3}. There is sensitive information in credentials C_{B1} and C_{B2}, so B makes two policies (4)&(5) to protect them. Principal A has a non-sensitive credential C_{A1} satisfies p_{B2} as (6). Based on these relations, we can build a hybrid tree as Fig. 3 using Algorithm 1.

Property 1 In a hybrid tree, let p be a policy node, it will at most has one child credential set node.

Explanation This property can be got through the steps of Algorithm 1. If the number of child credential set nodes connected to the parent policy node is larger than 1, then we get:

$$|\text{getPolicySolution}(credsBase, p)| \geq 2.$$

Without losing generality, we assume there exist two credential sets $N_1 \neq N_2$ s.t. $N_1 \Rightarrow p, N_2 \Rightarrow p$, it is an obvious contradiction with definition of Minimum Credential Set Solution of the policy. Therefore, a policy node in a hybrid tree will at most has one child credential set node.

Theorem 1 *Let $\mathcal{P}_A, \mathcal{P}_B$ be the policies of both participants during the trust negotiation respectively, and then the time complexity of the hybrid tree building algorithm is polynomial with the length of policies.*

Proof Here we define the length of a policy p as $|p|$, which is the number of atomic policies within a normalized p (in a CNF formula). For example, the length of $p = C_1 \wedge C_2$ is $|C_1 \wedge C_2| = 1$. We use L to denote the maximum value $\max(\text{len}(p_i))$ in all policies.

Considering the line 4–16 in the while loop of Algorithm 1, we can reach a conclusion that the maximum number of *Policies Node* is $L = |\mathcal{P}_A| + |\mathcal{P}_B|$. According to Property 1, the number of *Credential Set Node* at most is $|\Sigma N| \leq L$, so the times of loop operation are less than $2L$.

For the internal loop, the number of atomic policy is no more than L, so the time complexity of this algorithm is $2L^2$, that is $2(|\mathcal{P}_A| + |\mathcal{P}_B|)^2$. Therefore, the time complexity

of the hybrid tree building algorithm is polynomial with the length of polices. □

Theorem 2 *In a hybrid tree, if node N is both a leaf node and a type of Crendetial Set Node, we can obtain a successful trust negotiation procedure (i.e., getting the authorization on resource R) which discloses credentials from leaf node N to the root node.*

Proof On the path from the root node to a credential set leaf node, the resource (original service or credential set) and policy node appear alternatively and end with a non-sensitive credential set, so there are totally $2 \times n + 1$ (n is an integer) nodes. In accordance with the order from the leaf node to the root node, we arrange these nodes as a sequence: $N_0, p_0, \ldots, p_{n-1}, N_n, R$ $(0 \leq n)$, R is the resource firstly requested, N_0 $(i = 0)$ represents a credential set node which can be disclosed, the other node N_i $(1 \leq i \leq n)$ represents a credential set containing sensitive credentials, whose security policy is node p_{i-1} $(1 \leq i \leq n)$. Obviously node N_i $(1 \leq i \leq n)$ is the minimum credential set solution of node p_{i-1} $(1 \leq i \leq n)$.

When $n = 0$, since N_0 is a node can be disclosed freely and also a minimum credential set solution of R, security policy represented by N_1 also can be satisfied by the other side. Thus it makes resources represented by $N_{2\times i}$ $(i = 1)$ be disclosed.

When $n = k$ $(0 \leq k < n)$, assuming a credential set represented by node N_k can be disclosed, then it means security policy in node p_k can be satisfied, thus making credential set represented by N_{k+1} be disclosed. That means when $n = k + 1$ $(0 \leq k < n)$, a credential set represented by N_{k+1} can also be disclosed;

By combining the above induction results, we can get a successful trust negotiation path to R through disclosing credentials on this path. □

From the above proof process, we can see that, to obtain an access authorization on resource R, the credentials must be disclosed in order from a leaf node to the root node. Besides, it also shows that a special case of hybrid tree path is a credential disclosure sequence. If we delete all security policy nodes in the sequence of $N_0, p_0, \ldots, p_{n-1}, N_n, R(0 \leq n)$, it becomes a pure credential disclosure sequence. Based on this theorem, we can see the hybrid tree in Fig. 2 contains four leaf nodes, but only the path taking N_{A1} as its leaf node can find a successful negotiation process, while paths taking p_{B1}, p_{B3} and p_{A3} respectively as leaf nodes cannot be successful.

Definition 3 (Trust Negotiation Process) For principal A and B, let their credential sets and policy sets be $\mathcal{N}_A, \mathcal{P}_A$ and $\mathcal{N}_B, \mathcal{P}_B$ respectively. Supposing principal A owns resource $\mathcal{R}$, then a trust negotiation process for B to request $\mathcal{R}$ of A is defined as a message exchange sequence $\langle M_1, \ldots, M_n \rangle$, which satisfies the following conditions:

(1) $M_{2k+1} \subseteq \mathcal{N}_A \cup \mathcal{P}_A, M_{2k} \subseteq \mathcal{N}_B \cup \mathcal{P}_B, k \geq 1$;
(2) If $N \subseteq M_k$, and the protected policy of N is p, then there is $\{\Sigma C_i\} \rightarrow p, C_i \in M_1 \cup \cdots \cup M_{k-1}, i < k$.

Condition 1 gives the interactive information exchanging process during trust negotiation, and both sides need to disclose necessary information to the other side so as to promote trust establishment between them. Condition 2 shows that if one side discloses a credential set protected by p, this policy p can be satisfied by former credentials disclosed by the other side.

Proposition 1 *During a trust negotiation process based on a hybrid tree, the number of policy nodes and credential set nodes is even, on the path that the trust construction establishment depends on.*

Explanation This property can be got obviously from Theorem 1. Based on this property, we can determine whether trust can be established through calculating the height of a hybrid tree. This is an effective tree pruning condition.

Compared with disclosure sequence and disclosure tree model, the hybrid tree model has the ability to express more realistic trust establishment scenarios, which is mainly reflected on the three aspects: (1) The hybrid tree introduces the access control policy node, which can express fine-grained security constraints. (2) A credential set node corresponds to a minimum satisfying credential set, and a policy node only has a credential set node, thus the tree model is simplified and redundant nodes are reduced. (3) The hybrid tree is merely an OR relation tree which makes us easier to searched or processed the hybrid tree than an complex AND/OR tree.

4 Adaptive trust negotiation strategy

In HiTrust, we design an adaptive trust negotiation strategy to improve the performance of trust negotiation. This strategy mainly includes two key techniques: *the policy index mechanism* based on a policy stack, and *the asynchronous trust chain propagation mechanism.*

4.1 Adaptive trust negotiation strategy

To construct a whole hybrid tree, it requires two negotiators to disclose their credentials and security policies interactively. Taking the scenario in Fig. 3 as an example, principal A firstly constructs a SubTree 1 (shown in Fig. 4), which

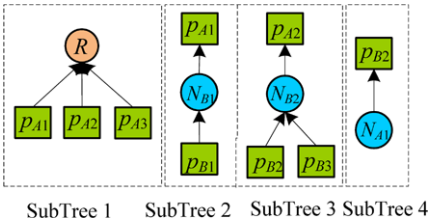

Fig. 4 SubTrees constructed during trust negotiation

indicates only if B satisfies one of atomic security policies p_{A1}, p_{A2} or p_{A3}, then it can access resource R. In order to prevent principal B from blindly disclosing information, principal A needs to disclose this policy. During the negotiation process, a principal can disclose its information according to its negotiation strategy.

Assuming A discloses three security policies p_{A1}, p_{A2} and p_{A3} based on SubTree 1. When B receives them, B will construct SubTree 2 and SubTree 3 with their respective credential sets, then B discloses policies p_{B1}, p_{B2} and p_{B3} to A at one time. A constructs SubTree 4 and in which the credential set N_{A1} can be freely disclosed. Based on the Theorem 1, there exists a trust chain from N_{A1} to resource R from N_{A1}.

During the negotiation process, if any side finds all the atomic policies cannot be satisfied, it will terminate the negotiation process and send a *terminate* signal enclosed in a *META* message so as to avoid unnecessary further interactions. All of the similar signals are stored in *META* message. Besides, if it is too strict to control the information disclosure, the rounds of negotiation interactions will be increased and extra trust establishment cost will be introduced. Therefore, the negotiation strategy should be adapted according to the security risk and network requirements.

The aim of negotiation strategy is to reduce the times of interactions, and improve the efficiency with low risk. Based on the hybrid tree, the negotiation strategy can adaptively guide the information disclosure. If there are more than one solutions of minimum credential set, a negotiation strategy needs to determine which set can be disclosed with in a higher priority. The aim of this strategy is to reduce the times of interactions, and improve the efficiency with low risk. The procedure of negotiation strategy in HiTrust is shown in Algorithm 2.

During the negotiation procedure, *META* information plays an important role. It mainly consists of two types of information: One is the negotiation state signal including negotiation *continue*, *failure*, *success* and so on. The other is associated meta-information with disclosed credentials or policies. For instance, when a credential set is sent, the policy *id* and *hash* will also be attached in *META*. This will help the receiver quickly query the exact policy, and avoid traversing all the nodes in the hybrid tree, thereby improving the negotiation efficiency. In this algorithm, a principal

Algorithm 2 Adaptive Negotiation Strategy Algorithm
Input: Message M_{recv} received from negotiation side
Output: Message M_{send} disclosed by the principal

```
1.  NegoStrategy(M_recv, hybridTree){
2.    if (∃P ∈ M_recv){
3.    for (every p ∈ P){ //processing every atomic
      security policy
4.      N ← getPolicySolution(N, p); } //N is a full
        credentials set of a principal
5.    for (∀C_i ∈ N){
6.      if (∃p_i s.t. ProtectedPol(C_i) = p_i){
7.        P_tmp = ∧p_i; }} //finding security policy for
          sensitive credentials, and combining every
          policy
8.    P_send = Normalization(P_tmp);} //normalize it as a
      CNF formula
9.    if (P_send == ∅){
10.     M_send.creds = N; }

11.   else{
12.     M_send.pol = NegoCfg.preprocess(P_send); }
        //sending the policy, which can implement various
        negotiation modes
13.   Update(hybridTree); //updating the state of hybrid
      tree
14.   }
15.   if ( ∃N ∈ M_recv){P = getSatisfiedPolicy(N);
16.     for (every p ∈ P ) {
17.       if (p.parenetNode() == N){
18.         M_send.creds = ∪N ; }}
19.     Update(hybridTree); //updating the state of
        Hybrid tree
20.     }
21.   if (M_send = ϕ || ∃META_msg ∈ M_recv){
22.     META_msg = NegoCfg(META_msg, M_send);
        //according to the meta information, continue or
        terminate the negotiation
23.     M_send.meta_msg = META_msgsend; }
24. }
```

can choose to disclose information based on previous policy configuration and context information. For example, when there are several security policies p with different sensitive levels for a credential set N, we can use a fine-grained negotiation strategy to choose an appropriate p. During the negotiation procedure, *META* information plays an important role. It mainly consists of two types of information: One is the negotiation state signal including negotiation *continue*, *failure*, *success* and so on. The other is associated meta-information with disclosed credentials or policies. For instance, when a credential set is sent, the policy *id* and *hash* will also be attached in *META*. This will help the receiver

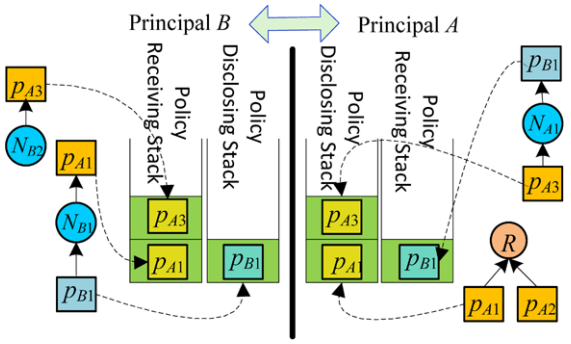

Fig. 5 A policy stack related the hybrid tree

quickly query the exact policy, and avoid traversing all the nodes in the hybrid tree, thereby improving the negotiation efficiency.

4.2 Policy Stack

Trust negotiation is a stateful process for participants, and all the key session information is stored with the hybrid tree structure. Because one in a complete negotiation process, more than one security policies of sensitive resources may be disclosed according to Theorem 2, lastly disclosed security policy must be satisfied firstly by credentials. Thus we put all the policies nodes in a stack (parallel policy nodes are stored as a list mode). In HiTrust, we use *Policy Disclosing Stack* and *Policy Receiving Stack* respectively to store the security policies. As illustrated in Fig. 5, the root node is the requested resource in a hybrid tree, the policy node in a protected edge is constructed by the policy in *Policy Disclosing Stack*, and the policy node in a satisfied edge is constructed by the policy in *Policy Receiving Stack*.

4.3 Asynchronous credential propagation mechanism

Since the collaboration over the Internet generally spans multiple security organizations, trust relationship may be evolved with different requirements. We classify the credential into two types. One is the principal's ending credential which is stored locally, and the other is delegation credential or federation credential among attribute authorities which is stored at the authority server. When constructing a trust chain, a principal needs to dynamically discover the delegation credentials or federation credentials from authority servers.

However, during the trust negotiation, it is a time-consuming process when a principal wants to discover a trust chain to prove its ultimate privilege. At the same time, the total time of trust chain construction is almost proportional to the length of trust chain. When a long trust chain is too long, the negotiation performance will be reduced and even unacceptable. In order to resolve this problem, we

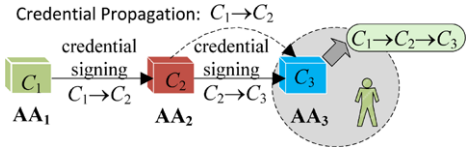

Fig. 6 Asynchronous credential propagation for trust chain

introduce an asynchronous credential propagation mechanism. When an authority issues a credential to another authority, it will propagate the credential to the next authority server. An example shown in Fig. 6, there are three security domains, where AA_1 and AA_2 have collaboration relation ($C_1 \rightarrow C_2$), and if AA_2 and AA_3 also have potential collaboration relation ($C_2 \rightarrow C_3$), the credential $C_1 \rightarrow C_2$ will be propagated from AA_2 to AA_3, so a user within AA_3 domain can easily construct a trust chain $C_1 \rightarrow C_2 \rightarrow C_3$.

In HiTrust, an attribute authority is implemented as a Web service, and a timestamp is embedded into a SOAP packet to prevent the replay attack. In HiTrust, The credential propagation is an asynchronous process, which is executed by an independent single process to guarantee the efficiency.

5 System implementation experience

5.1 HiTrust prototype

We have implemented HiTrust approaches in CROWN middleware. CROWN is the brief name for China Research and Development environment Over Wide-area Network, and it aims to promote the utilization of valuable resources and cooperation of researchers nationwide and world-wide. CROWN service grid middleware is the kernel to build an application service grid. CROWN adopts an OGSA/WSRF compatible architecture, and considers the application requirements and the limitation of security architecture of OGSA/WSRF, more focus is put on the Grid resource management and dynamic management mechanism. CROWN NodeServer is implemented based on Globus Toolkit Service Container, and some security specifications such as WS-Trust, WS-Security and WS-SecureConversation are used for secure communication.

The architecture of HiTrust Agent is shown as in Fig. 7, and it includes five key modules.

(1) **Message Agent**: It is responsible for the protocol message encapsulation and parsing during the negotiation procedure. It firstly query the message *session id*, then extracts the *credentials*, *policies* and *meta-information* enclosed in communication message, finally forwards the data to the corresponding modules.

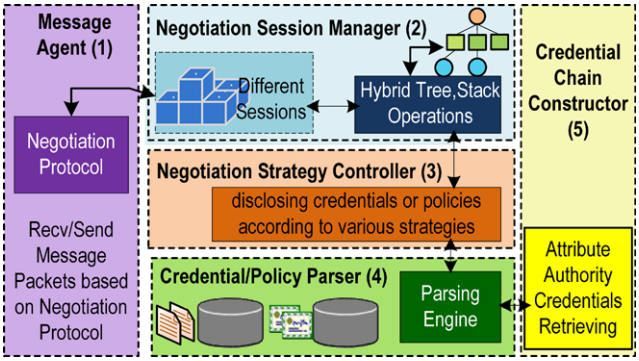

Fig. 7 Architecture of HiTrust agent

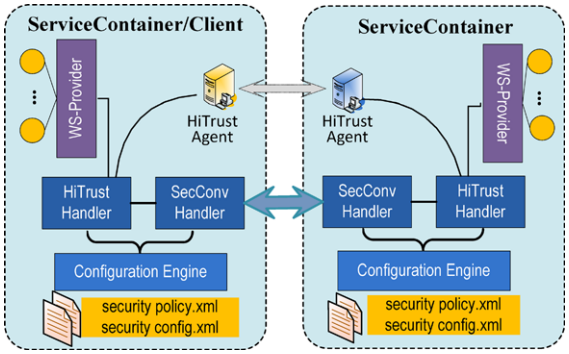

Fig. 8 Chain-based security policy configurable HiTrust

(2) **Negotiation Session Manager**: It is responsible for the negotiation strategy selection and negotiation state maintaining for multiple trust establishment sessions. For a negotiation session, the state is fully stored by the hybrid tree and policy stack. Different session uses different negotiation strategy according to its security risk or network condition.

(3) **Negotiation Strategy Controller**: It implements the functions of Algorithm 2, and controlling the disclosure of credentials and policies according to the negotiation strategy.

(4) **Credential/Policy Parser**: It is responsible for the credential and policy management, and providing a caching mechanism for frequently visited credentials. In HiTrust, credential is of X.509 v3 or SAML format, and security policy is of XACML format. All the functions such as credential verification, constraints verification are done by the Parsing Engine.

(5) **Credential Chain Constructor**: It is responsible for credentials retrieval and credential chain construction. During the negotiation, it not only verifies the validation of credential chains, but also constructs credential chains satisfied specific security policy. To reduce the burden of lots of credentials retrieval, an asynchronous credential propagation mechanism is employed in this module.

The HiTrust service is implemented as a Web service in CROWN NodeServer. As illustrated in Fig. 8, multiple procedures are involved in the trust negotiation. When a client principal requests a target service which is protected by trust negotiation service, it will firstly initialize a HiTrust Agent. Upon receiving the negotiation requests from client, the service provider will also create a HiTrust Agent. The state of negotiation will be stored in *hybrid tree* and *policy stack* respectively. Then, the two participants may disclose their credentials or policies for sensitive credentials according to the negotiation strategy. If the negotiation succeeds, *HiTrust* will return a success status, and the context will be updated accordingly. The requester can insert the session id into SOAP header and sign it before sending to the target service. The target service will verify the authenticity of session id through its *HiTrustHandler*, and allow the access if the verification succeeds.

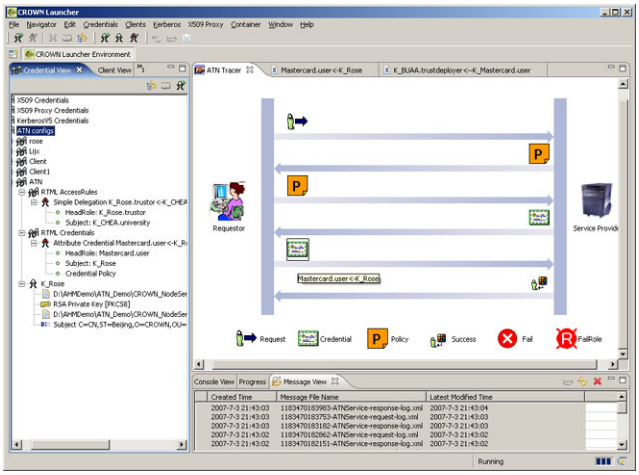

Fig. 9 Screenshots of ATNLauncher

We also developed an Eclipse Plugin-based client, named *ATNLauncher* (shown in Fig. 9), which provides a tool to manage credentials, access control policies and config files. In *ATNLauncher,* client command parameter string can be generated via a wizard, and the negotiation procedure between client and server is displayed on a special GUI view, and we can check detailed information by clicking the policy or credential icon in the view. Moreover, the SOAP messages during negotiation can also be monitored and logged.

5.2 Experimental evaluation

We have conducted some experiments in CROWN test bed. CROWN middleware is deployed on a cluster node with Intel Xeon 2.8 GHz CPU, 2 G RAM, Linux operating systems and 100 Mbps Internet connection. The client is an IBM T40 computer with Intel 1.6 GHz CPU, 1 G RAM, Windows XP operating system. In order to ensure the accuracy of evaluation, there are no other programs running on the computers.

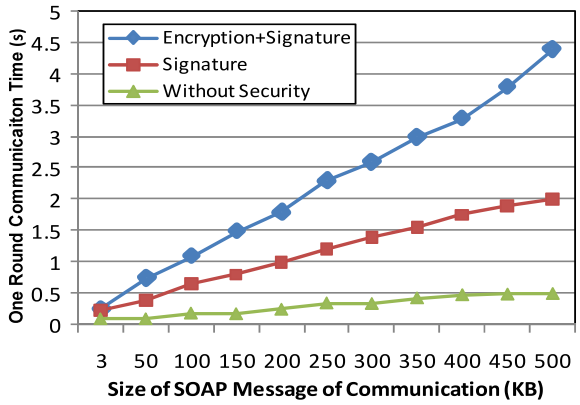

Fig. 10 Efficiency of HiTrust secure communication

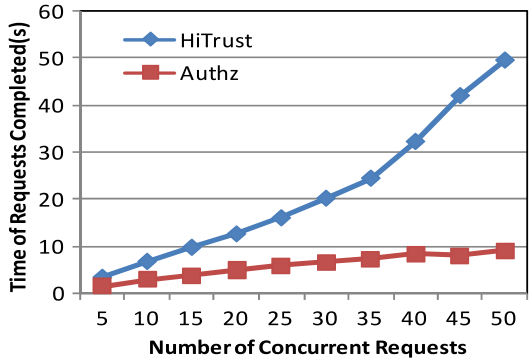

Fig. 11 Execution time of concurrent requests

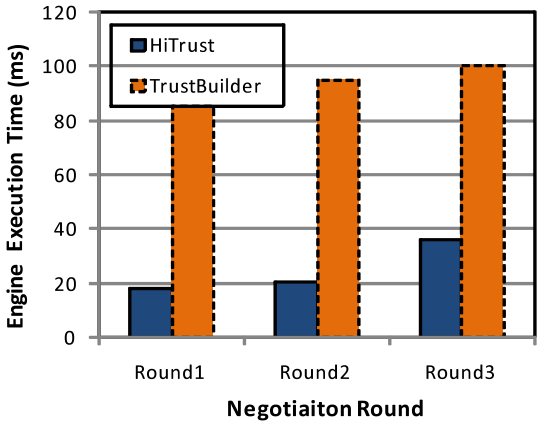

Fig. 12 The engine execution time of every negotiation round for HiTurst and TrustBuilder

Unless special statement, each experiment is executed five times and the average value is chosen.

Compared with general authorization mechanism without consideration to privacy and other sensitive information protection, trust negotiation primarily brings extra cost on secure communication and multiple interactions, thereby we want to evaluate the processing capability of HiTrust services. The performance is mainly impacted by three factors: traffic overhead (security communication), concurrent requests and credential queries. We generate four simulation groups to study our approach.

Experimental Group 1 *In this simulation, the credential uses Base* 64 *encoding, the size of every credential file in pem format is about* 1–2 *KB, and every security policy rule is about* 1–5 *KB. Through varying the size of SOAP message, we measure the request response time respectively.*

Figure 10 shows the one round communication time against the size of SOAP packet from 3–500 KB. When no security mechanism is employed, the request response time is below 500 ms (Without Security curve); When a RSA-SHA1 security signature mechanism is employed, the time is only about 650 ms for 100 KB SOAP message (Signature curve); When a 3DES-CBC encryption and RSA-SHA1 signature mechanism is employed, the time is about 1100 ms for 100 KB SOAP message (Encryption+Signature curve). As we can see, the one-round communication time increases linearly with the increasing size of SOAP message, and the security mechanism will impact the response time. But in a common trust negotiation scenario, the extra communication overhead is small when the number of credential and atomic policy is small (e.g., even if 10 credentials and 10 policies are sent, the total size of communication is no more than 100 KB).

Experimental Group 2 *We designed two scenarios, one is a traditional authorization scenario without privacy protection, and the other is a trust negotiation with a three-round interaction. The number of concurrent requesting clients is varied from* 5 *to* 50*, we measure the total time of requests with and without trust negotiation mechanism.*

The result is shown in Fig. 11. We can see the total execution time is about 10 s for 50 concurrent requests under the traditional authorization mechanism (Authz curve), and about 50 s under the trust negotiation mechanism (HiTrust curve). The result shows that the overall executing time almost increases linearly with the number of concurrent requests, and negotiation service performance is scalable.

Experimental Group 3 *We Launch the service deployed on cluster node, and a requester invokes the negotiation service to measure the performance of HiTrust Agent. We select a typical example with three rounds interaction, which is also used in TrustBuilder. We measure the time of every negotiation decision for HiTrust and TrustBuilder.*

Figure 12 shows the average negotiation response time following different rounds. The bar chart indicates that the engine execution time during every interaction round in this example. As a whole, the time is approximately 15–40 ms

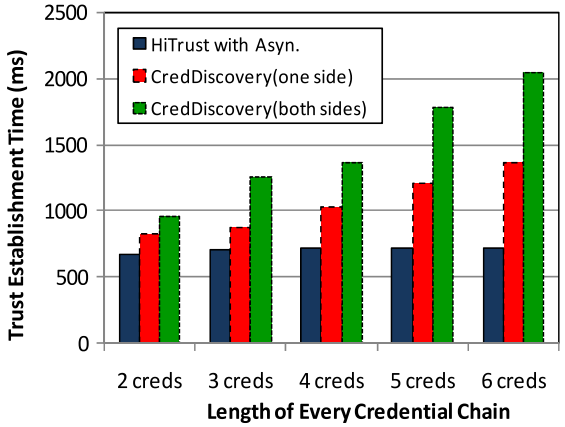

Fig. 13 HiTrust service with credential chain construction

in HiTrust. In Addition, we designed a similar scenario with Example 1 for TrustBuilder, and get the results. As shown in Fig. 12, the response time of TrustBuilder is higher than HiTrust. This is because that TE, on which TrustBuilder based, uses a negotiation tree model which need dynamically search and visit all credential nodes.

Experimental Group 4 *We still use a trust negotiation scenario with a three-round interaction, and generating different examples by varying the length of credential chain from* 2 *to* 6. *We mainly compare the time of trust establishment under three trust chain constructing methods. The first is the asynchronous credential propagation, and all credentials have been collected into local credential repository. In the second method, one negotiation side needs to get credentials from the remote AA servers. In the third method, both negotiation sides need to get credentials form the remote AA servers.*

As shown in Fig. 13, the average time of getting a credential from an AA server is about 100–200 ms. For instance, when there are four credentials in a credential chain, the trust establishment time is about 770 ms, 1100 ms and 1300 ms for three different examples respectively. We can see that dynamic trust chain construction overhead has a great impact on trust negotiation performance. Therefore, the asynchronous credential propagation mechanism is very efficient to a realistic application.

6 Conclusions and future work

With the widely adoption of pervasive computing environments over the Internet, trust establishment for mobile services sharing and collaboration has become an important issue.

We proposed a hybrid tree based dynamic cross-organizational trust establishment service. The hybrid tree contains both policy node and credential set node, and it enables fine-grained security policy constraints for attributes and negotiation context, thereby it not only enhances the capability of trust establishment, but also simplifies tree search algorithm. Furthermore, we presented an adaptive trust negotiation strategy based on the hybrid tree, where a policy-stack based mechanism is employed to improve efficiency of policy query, and an asynchronous trust chain propagation mechanism is used to enhance the usability of HiTrust service.

At present, we are also constructing a cloud computing environment iVIC [13],[1] which is a network software operating environment to provide the elastic, scalable, and transparent resource management mechanism. It leverages virtual machine (VM) or virtual network to launch network software, and delivers desired software on-demand through presentation streaming mode (based on VNC) to PC or mobile phone. Currently, iVIC has been used as a virtual lab infrastructure for our campus courses experiment. We are also trying to integrate the negotiation strategy to evaluate the risk of iVIC computing environment. In addition, trust establishment is also a key technology for data and application migration among multiple Clouds or Virtual Network. We will integrate our approach into iVIC to provide a transparent trust enabled collaboration infrastructure for mobile users.

Acknowledgements The authors gratefully acknowledge the anonymous reviewers for their helpful suggestions and comments. Some preliminary results of this work were presented in AINAW 2010. This work is partially supported by Program for New Century Excellent Talents in University 2010 and the Fundamental Research Funds for the Central Universities, National Nature Science Foundation of China (No. 60903149), China 863 Program (No. 2009AA01Z419), and China 973 Fundamental R&D Program (No. 2011CB302602, 2011CB302603).

References

1. Hansmann, U. (2003). *Pervasive computing: the mobile world*. Berlin: Springer. ISBN 3540002189.
2. Ahmed, A., & Zhang, N. (2009). Towards the realisation of context-risk-aware access control in pervasive computing. *Telecommunication Systems Journal*. doi:10.1007/s11235-009-9240-3.
3. Winsborough, W. H., & Li, N. (2002). Towards practical automated trust negotiation. In *Proceedings of the 3rd international workshop on policies for distributed systems and networks (POLICY'02)* Monterey, CA, USA.
4. Zou, D., Park, J. H., Yang, L. T., Liao, Z., & Kim, T. (2008). A formal framework for expressing trust negotiation in the ubiquitous computing environment. In *Proceedings of the 5th international*

[1] http://portal.ivic.org.cn.

conference on ubiquitous intelligence and computing, Oslo, Norway, June 23–25. *Lecture notes in computer science* (Vol. 5061, pp. 35–45). Berlin: Springer.
5. Li, N., Winsborough, W. H., & Mitchell, J. C. (2003). Distributed credential chain discovery in trust management. *Journal of Computer Security*, *11*, 35–86.
6. Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., & Yu, L. (2002). The TrustBuilder architecture for trust negotiation. *IEEE Internet Computing*, *6*(6), 30–37.
7. Constandache, I., Olmedilla, D., & Nejdl, W. (2005). Policy based dynamic negotiation for grid services authorization. In *Semantic web policy workshop in conjunction with 4th international semantic web conference*, Galway, Ireland.
8. Bertino, E., Ferrari, E., & Squicciarini, A. C. (2004). Trust-X:A peer to peer Framework for trust negotiations. *IEEE Transactions on Knowledge and Data Engineering*, *16*, 827–841.
9. Yamaki, H., Fujii, M., & Nakatsuka, K. (2005). A dynamic programming approach to automated trust negotiation for multiagent systems. In *1st international workshop on rational, robust, and secure negotiations in multi-agent systems (RRS2005)*.
10. Dragoni, N., Massacci, F., & Saidane, A. (2009). A self-protecting and self-healing framework for negotiating services and trust in autonomic communication systems. *Computer Networks*, *53*, 1628–1648.
11. Dragoni, N., & Saidane, A. (2008). A framework for dependable trust negotiation in open environments. In *Fifth IEEE workshop on engineering of autonomic and autonomous systems*, Belfast.
12. Winslett, M., Adam, J. L., & Kenneth, J. P. (2009). Trust negotiation: authorization for virtual organizations. In *Proceedings of the 5th annual workshop on cyber security and information intelligence research: cyber security and information intelligence challenges and Strategies*. Oak Ridge: ACM.
13. Li, J., Li, B., Wo, T., Hu, C., Huai, J., Liu, L., & Lam, K. P. (2011). CyberGuarder: A virtualization security assurance architecture for green cloud computing. *Future Generation Computer Systems*. doi:10.1016/j.future.2011.04.012.
14. Winsborough, W. H., Seamons, K. E., & Jones, V. E. (2000). Automated trust negotiation. In *DARPA information survivability conference and exposition*.
15. Winslett, T. Y. M., & Seamons, K. (2003). Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security 6*, 1–42.
16. Yu, T., & Winslett, X. M. M. (2000). PRUNES: an efficient and complete strategy for automated trust negotiation over the Internet. In *Conference on computer and communications security (CCS00)*, Athens, Greece.
17. Yu, T., & Winslett, M. (2003). A unified scheme for resource protection in automated trust negotiation. In *IEEE symposium on security and privacy*, Berkeley, California.
18. Skogsrud, H., Benatallah, B., & Casati, F. (2004). Trust-Serv: model-driven lifecycle management of trust negotiation policies for web services. In *Proceeding of 13th world wide web conference (WWW2004)*, New York, NY, USA.
19. Chen, W., Clarke, L., Kurose, J., & Towsley, D. (2005). Optimizing cost-sensitive trust-negotiation protocols. In *Proceedings of the 24th conference of the IEEE communications society (Infocom 2005)*, Miami, FL.
20. Bonatti, P., & Olmedilla, D. (2005). Driving and monitoring provisional trust negotiation with metapolicies. In *Sixth IEEE international workshop on policies for distributed systems and networks (POLICY'05)*, Stockholm, Sweden.
21. Huai, J., Hu, C., Li, J., et al. (2007). CROWN: A service grid middleware with trust management mechanism. *Science in China Series F: Information Sciences*, *49*, 731–758.

Jianxin Li is an associate professor in the School of Computer Science and Engineering, Beihang University, Beijing, China. He received the Ph.D. degree in Jan. 2007. He has authored over 30 papers in IEEE T. on Industry Electronic, Information Sciences, SRDS, HASE and eScience etc. His research interests include trust management, information security and distributed system.

Xudong Liu is a professor and dean of the School of Computer Science and Engineering, Beihang University, Beijing, China. Has have leaded several China 863 key projects and e-government projects. He has published more over 30 papers, more than 10 patents. His research interests include software middleware technology, software development methods and tools, large-scale information technology projects and application of research and teaching.

Lu Liu is Senior Lecturer in School of Computing and Mathematics, University of Derby (UK). Before joining University of Derby, he was Lecturer in School of Engineering and Information Sciences at Middlesex University (UK). Prior to his academic career, he was Research Fellow in the School of Computing at the University of Leeds (UK), working on NECTISE Project which was an UK EPSRC/BAE Systems funded research project involving ten UK Universities and CoLaB Project which was funded by UK EPSRC and China 863 Program. He received a Ph.D. degree (funded by UK DIF DTC) from the University of Surrey (UK) and M.Sc. degree from Brunel University (UK). His research interests are in areas of service-oriented computing, software engineering, Grid computing and peer-to-peer computing. Dr Liu has over 50 scientific publications in reputable journals, academic books and international conferences. He won the Best Paper Award at the Realizing Network Enabled Capability Conference in 2008. He is member of IEEE.

Dazhi Sun received the M.E. degree in control science and engineering from Nanchang University, Jiangxi, in 2002, and the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, Shanghai, China, in 2006. He is currently a lecturer at the School of Computer Science and Technology, Tianjin University, Tianjin, China. His current research interests include applied number theory, applied cryptography, and information security.

Bo Li received his Bachelor and Master degrees from Dalian University of Technology, China. Now he is a Ph.D. student at the Department of Computer Science, Beihang University, China. His research interests include a broad range of topics related to computer security including virtualization security, operating system security, and trusted computing.