A Complete and Efficient Strategy Based on Petri Net in Automated Trust Negotiation

Yan He
College of Computer Science, Zhejiang University
Zhejiang Financial Profession College
Hangzhou, China
heheyan@126.com

Miaoliang Zhu
College of Computer Science, Zhejiang University
Hangzhou, China
zhum@zju.edu.cn

ABSTRACT
Traditional security model, where the identity of all possible requesting subjects must be pre-registered in advance, is not suitable for the distributed applications with strong real-time requirements, especially recently popular P2P networks and Grid computing. A promising approach is represented by automated trust negotiation, which establishes trust between strangers through the exchange of digital credentials and the use of access control policies. An automated trust negotiation strategy needs to be adopted to establish trust between two parties based on their disclosure policies. Previously proposed negotiation strategies may fail when in fact success is possible, disclose irrelevant credentials, or have high communication or computational complexity. In this paper, we model the policies participating trust negotiation as Negotiation Petri Net and propose a trust negotiation Strategy based on Negotiation Petri Net (SNPN) by combining the characteristics of Negotiation Petri Net architecture with the behaviors of auto trust negotiation. We prove that SNPN is efficient with O(n) communication complexity and O(nm) computational complexity including Negotiation Petri Net building process and the negotiation process in the worst case, where n is the number of credentials and m is the size of the credential disclosure policies. Meanwhile SNPN is complete and makes sure that no irrelevant credentials will be disclosed during negotiations.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.4.6 [**Operating Systems**]: Security and Protection - Access Control

General Terms
Security.

Keywords
automated trust negotiation; negotiation strategy; Negotiation Petri Net.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
***Conference name:** Infoscale 2007, June 6-8, 2007, Suzhou, China
***Copyright number (LaTeX \crdata{}):** 978-1-59593-757-5

1. INTRODUCTION
Variety of distributed applications with strong real-time requirements, especially recently popular P2P networks[1] and Grid computing[2], has challenged the traditional security model. In distributed systems, different parties may make connections and do business without being previously known to each other, different resources are shared across organizational boundaries and a potentially unbounded number of users and resources exist with few guarantees regarding pre-existing trust relationships. However, the traditional security model, where the identity of all possible requesting subjects must be pre-registered in advance, is not suitable for the distributed environments. A promising approach is represented by automated trust negotiation[3,4,5,6], which establishes trust between strangers through the exchange of digital credentials and the use of access control policies that specify what combinations of credentials a stranger must disclose in order to gain access to each local service or credential.

In automated trust negotiation, access control decisions are made based on the attributes of requester rather than his identity. A credential is a digitally signed assertion by a credential issuer about the credential owner regarding one or more attributes about the owner, each consisting of an attribute name/value pair and describing some property of the owner asserted by the issuer[7,8]. Since credentials may contain sensitive and private information, the disclosure of credentials also must be protected through the use of policies that specify which credentials must be received before the requested credential can be disclosed[9]. A trust negotiation is triggered when one party requests access to a resource owned by another party. Since each party may have access control policies that the other needs to satisfy, trust is established incrementally through the exchange of digital credentials. Negotiation strategy controls the exact content of the messages that a party sends to others, i.e., which credentials to disclose, when to disclose them, and when to terminate a negotiation. Successful trust negotiation is not always possible, as the parties may not possess needed credentials, or subjects may govern their credentials by policies that, together, impose cyclic dependencies[3]. A complete strategy[10] should be able to find a successful credential exchange sequence whenever such a sequence exists. It is high desirable that the negotiation strategy be complete, reasonably efficient, and avoidable disclosing any credentials that are not needed for the successful negotiation.

Some negotiation strategies have been proposed, with different defects that they may fail when in fact success is possible, disclose irrelevant credentials, or have high communication or computational complexity. The earliest negotiation strategies, the *eager* strategy and the *parsimonious* strategy are proposed in [3].

The *eager* strategy is complete and efficient, but with an important disadvantage that some irrelevant credentials may be disclosed unnecessarily. Conversely, the *parsimonious* strategy discloses credentials only after exchanging sufficient policy content to ensure that a successful outcome is ensured, but it is not complete and introduces the difficulty of deciding when the negotiation should fail and stop. [10,11] proposed *Prudent Negotiation Strategy* (PRUNES). PRUNES is based on backtracking the AND/OR tree formed by the credentials and policies. It can finish negotiation efficiently, but ignoring the complexity of building the AND/OR tree, which requires exponential time and space. Yu et al.[12,13] developed families of strategies called *disclosure tree strategy* (DTS) where two parties can negotiate trust if they choose different strategies within the same family. Each disclosure tree is a branch of a set of policies. Although some strategies in the DTS family are efficient, like the *TrustBuilder-Simple* and *TrustBuilder-Relevant* shown in the articles, it would require exponential time and space during a negotiation. A credential or a policy is assigned a weighted cost in [14]. The objective of the paper is to minimize the total sensitivity costs of the credentials and policies disclosed by a trust negotiation protocol, which proposes a new direction of search in trust negotiation strategy. Ye et al.[15] introduces a third party trusted by both peers to act as a mediator and disclose their credentials and policy rules to each other when appropriate, thus breaking the cyclic dependency and allowing trust negotiation to succeed in peer-to-peer systems, but in contravention of not involving third party in auto trust negotiation.

In this paper, we introduce the definition of Petri Net to the trust negotiation and model the policies participating negotiation as Negotiation Petri Net. Petri Net[16] is an excellent formalism to model a large class of discrete state systems that exhibit a large amount of asynchronous behavior, yet have the capability to occasionally synchronize some of their activities. In auto trust negotiation, the policies of both parties compose a collection for a possible negotiation progress, and the credentials in the policies is asynchronous or synchronize as some credentials must be disclosed based on others. It is reasonable and efficient to model the policies participating trust negotiation as a Negotiation Petri Net. Based on the model, a trust negotiation *Strategy based on Negotiation Petri Net* (SNPN) is proposed. SNPN guarantees to succeed whenever trust establishment is possible between two parties, meanwhile it makes sure that no credential will be disclosed if the negotiation fails, and no irrelevant credentials will be disclosed if the negotiation succeeds. We prove that SNPN is efficient, in the worst case, the communication complexity is $O(n)$ and the computational complexity is $O(nm)$ including Negotiation Petri Net building process and the negotiation process in the worst case, where n is the number of credentials involved in the trust establishment and m is the total size of the credential disclosure policies for these credentials.

The paper is organized as follows. Section 2 defines credential disclosure policies. Section 3 proposes Negotiation Petri Net model and its Reverse Negotiation Petri Net. In section 4 the negotiation process of SNPN is discussed in detail. Section 5 analyzes communication and computational complexity of SNPN. Section 6 draws conclusions and describes directions of possible future work.

2. CREDENTIAL DISCLOSURE POLICIES

In auto trust negotiation, trust is established incrementally through a sequence of bilateral credential disclosures without involving third parties. A negotiation process is initiated by a party (typically, a Client) requesting services from another party (a Server). If the initially requested services are granted, trust is established and the client can visit the service. In the simplest case where credentials do not contain sensitive information, they can be shown to anybody whenever requested, and only the service itself needs to be protected from unauthorized access. The client will be willing to provide any credentials requested by the server in order to get the service. In this case, the trust negotiation can be finished in a single round. But the common cases are that credentials contain sensitive information and need to be protected from unauthorized access, a single-round trust negotiation is no longer sufficient, and a certain level of trust must be established before a party is willing to disclose a credential[17].

In this paper, we formalize the trust negotiation process using prepositional symbols as in [10]. A credential C is *disclosed* if it has been sent to the other party in the negotiation. A credential access control policy for a resource R is defined as a credential expression:

$$C_R \leftarrow F_R(C_1, C_2, \ldots, C_k)$$

where F_R is a logical form with credentials from the other party $C_1, C_2, \ldots, C_k$ and the boolean operators $\wedge$ and $\vee$. C_i ($1 \le i \le k$) is satisfied if and only if the other party reveals credential C_i. The credential C_R of resource R can be access by the other party if $F_R(C_1, C_2, \ldots, C_k)$ is evaluated to *TRUE*. If credential C_R is disclosed without requiring the disclosure of any other credential, which means this credential can be freely disclosed whenever requested, then credential C_R is said to be *unprotected*. The policy for C_R is written as $C_R \leftarrow \mathcal{E}$. Also, when an agent does not have credential C_R, or when it does not disclose C_R in any cases, the policy for C_R is written as $C_R \leftarrow$ *false* or omitted generally, and is called as denial policy. Obviously a party has denial policies for credentials it does not possess. An intuitive observation is that two parties cannot establish trust unless there is at least one unprotected credential on either side.

Given sequence $G = (C_1, \ldots, C_{|G|})$ of disclosures of protected resources, $|G|$ is the number of the credentials in the sequence. If each C_i is unlocked at the time it is disclosed, which means G is applied and satisfies resource R's access control policy, then we say G is a *safe disclosure sequence* for R. The purpose of trust negotiation is to find a safe disclosure sequence where $C_{|G|} = R$, the resource to which access was originally requested.

Figure 1 shows a successful trust negotiation process initiated by a client requesting service S from a server. The client's access control policies are shown at the left, and the server's access control policies are shown at the right. The client begins by revealing credential C_5, since no previously received server credentials are needed in order for the client to disclose it. The server then discloses S_3, which requires the earlier receipt of client credential C_4 or C_5. The credential exchange process continues as shown in the center of the figure and finishes by S is disclosed. At each round, all policies for disclosed credentials are satisfied. The safe disclosure sequence in the figure is G=(C_5, S_3, C_1, S_2, C_2, C_4, S). There exists other safe disclosure sequences, for example, G'=(C_4, S_3, C_1, S_2, C_2, C_4, S).

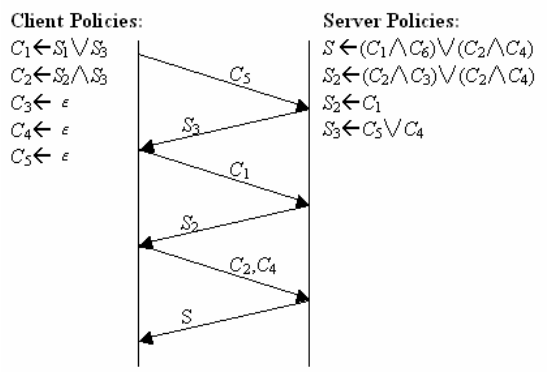


Figure 1. An Exmple of Disclosure Policies

The sequence of exchanged credentials is decided by a trust negotiation strategy based on local credentials, local policies, requests for local credential from the other party, and credentials received from the other party. The strategy starts when the client's security agent sends a request for service *S* to the server, and then the server checks *S*'s policy. If *S* is an unprotected service, the request for access to it is granted immediately, which means the security agent informs the client that credential *S* can be visited immediately. If the server does not possess *S*, the request is denied. Otherwise, the security agent of server tries to find solutions to the policy, for example, $S \leftarrow (C_1 \wedge C_6) \vee (C_2 \wedge C_4)$ in figure 1. The server sends the requests for the credentials in the policy to the client. The security agent of client check the credentials based on the local policies and if one of the disjunction of credential for *S* can be fulfilled, a safe disclosure sequence is got and credentials are actually disclosed according to the order of the sequence. Otherwise the client sends requests to the server for the credentials needed in its policies. The check goes on until *S* can be granted or denied. If the request for service *S* is denied, the process halts and claims that a successful negotiation is impossible between the two parties. If *S* can be granted, a safe disclosure sequence is returned. Then both parties in the negotiation begin to disclose credentials, using grant and deny information accumulated during previous negotiation rounds.

3. NEGOTIATION PETRI NET

In this section, we introduce the definition of Petri Net to the trust negotiation and model the policies of both parties as a Negotiation Petri Net. For the strategy running, which is analyzed in the next section, a Reverse Negotiation Petri Net is built base on Negotiation Petri Net.

Petri Net offers a powerful formalism for analysis of the concurrency or the interaction of events. A Petri Net structure is defined as a triplet $N = (P, T; F)$, where $P = \{P_1, P_2, \ldots, P_n\}$ is a finite set of *places*, $T = \{T_1, T_2, \ldots, T_k\}$ is a finite set of *transitions* with $P \cap T = \phi, P \cup T \neq \phi$, F is a *flow relation* with $F \subseteq P \times T \cup T \times P$, (× is a Cartesian product). Each place in P represents a state of resource, transitions in T generate the flow of the states in P which is expressed by F. Places contain tokens which is the number of the resources. Generally, a place is drawn as circle, a transition is drawn as a bar, a flow relation is drawn with an arrowhead on their destination and tokens are drawn as dots in the circles.

We model the trust negotiation progress based on the series of negotiation policies of the server and the client. The definition of Negotiation Petri Net is as follows.

Definition 1. *Negotiation Petri Net* is a triplet $NP = \{P, T; F\}$, where $P = \{C_i$, which is the credentials of the client and the server$\}$. $T = \{t_i$, which is the operations of the credentials in P except for the policies for the unprotected credentials$\}$. F = {the flow relation between nodes in T and P corresponding to the trust negotiation policies$\}$.

In Negotiation Petri Net, all the credentials of client and server displayed in the trust negotiation policies form the places set P, while the nodes in T correspond to the operations of credentials. No direct flow relations between two places or two transitions. The flow relations flow from a transition denote the ∧ operation, while the flow relations flow from a place denote the ∨ operation. Negotiation Petri Net starts from the credential for the service *S* originally requested by the client. If a trust negotiation policy is $S_1 \leftarrow C_1 \wedge C_2 \wedge \ldots \wedge C_n$, then the place node for S_1 in Negotiation Petri Net has one transition node t_1 as its child, n place nodes as the children of t_1, and F is $\{(S_1,t_1),(t_1,C_1),(t_1,C_2),\ldots,(t_1,C_n)\}$. If a trust negotiation policy is $S_1 \leftarrow C_1 \vee C_2 \vee \ldots \vee C_n$, the place node for S_1 has n transition nodes $t_1,t_2,\ldots,t_n$ as its children, while each transition node has one child, and F is $\{(S_1,t_1), (S_1,t_2),\ldots, (S_1,t_n), (t_1,C_1),(t_2,C_2),\ldots,(t_n,C_n)\}$. For the simplest trust negotiation policy $S_1 \leftarrow C_1$, S_1 has one transition node and the transition node takes the place node C_1 as its child, $F = \{(S_1,t_1),(t_1,C_1)\}$. Tokens are set in the place nodes corresponding to the unprotected credentials, which are called *TRUE* credentials. As the unprotected credentials can be visited at any time, the number of tokens is set as infinite. One dot is set in an unprotected credential place. Once token flows from the place along a flow relation in F to another place, one dot is added in the destination place and the dot in the source place will not change.

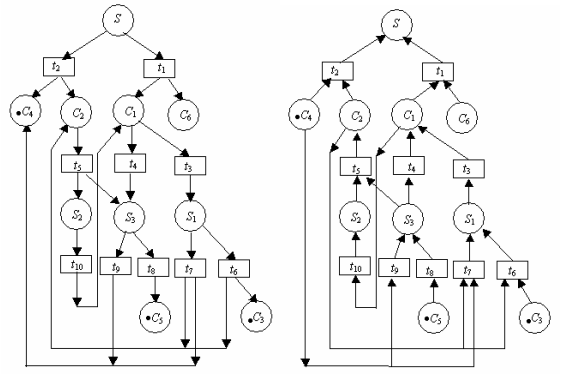
Figure 2(a) gives the Negotiation Petri Net of the policies and credential exchange sequence of the Figure 1. Negotiation Petri Net starts from the credential for the service *S* originally requested by the client and *S* needs the client show its credential C_1 together with C_6 or C_2 together with C_4, so place *S* can either flow through transition t_1 to places C_1 and C_6, or flow through transition t_2 to C_2 and C_4. As C_3, C_4 and C_5 are unprotected credentials, tokens are set in the corresponding places.

Definition 2. Triplet $NP' = \{P, T; F^{-1}\}$ is called *Reverse Negotiation Petri Net* of $NP = \{P, T; F\}$, where $F^{-1} = \{(x, y) \mid (y, x) \in F\}$.

Reverse Negotiation Petri Net has the same P and T with Negotiation Petri Net, but only change the arrow directions of the flow relations in F. In Negotiation Petri Net, a child transition node of a place represents a clause of the credential' policy, so each transition node has only one parent place node. Therefore in Reverse Negotiation Petri Net, each transition node has only one child place node. Negotiation Petri Net starts from *S*, there are flow relations starting from *S* but no flow relations ending by *S*. Then in Reverse Negotiation Petri Net, no flow relations flow from *S*. Tokens firstly flow from the places corresponding to the *TRUE* credentials to other places through the flow relation in F^{-1}.

Once a token flows from a source place to a destination place, a new token is added in the destination place.

Figure 2(b) shows the Reverse Negotiation Petri Net of Figure 1. Reverse Negotiation Petri Net changes the arrow directions of Negotiation Petri Net. C_3, C_4, C_5 and C_6 become the starting point and S becomes the end point.



(a) Negotiation Pet net (b) Reverse Negotiation Petri Net

Figure 2. Negotiation Petri Net and Reverse Negotiation Petri Net for the Disclosure Policies of Figure 1

```
input: Policies= the server policies and the client policies;
output: Negotiation Petri Net NP=(P,T,F);
Construct Negotiation Petri Net (Policies){
    P= Φ; T= Φ; F= Φ;
    WHILE(Polices≠ Φ){
        Pick a policy from Policies, p_i;
        BuildNegotiationPetriNet(P,T,F,p_i);
        Policies = Policies – p_i;   }
    RETURN(P,T,F);
}

BuildNegotiationPetriNet(P,T,F,p_i){
    IF (target(p_i)∉ P){
        P = P+ target(p_i);
        create a Place node for target(p_i);   }
    IF (condition(p_i)=TRUE)
        add a token in target(p_i) and mark it;
    FOR EACH disjunction D of condition(p_i) {
        create a Transition node t_k;
        T=T+ t_k;
        F=F+{( target(p_i), t_k)};
        FOR EACH element e of D{
            IF (e ∉ P)   P=P+{e};
            F=F+{(t_k,e)};   }   }
}
```

```
input: Negotiation Petri Net NP=(P,T,F);
output: Reverse Negotiation Petri Net NP'=(P,T,F');
Construct Reverse Negotiation Petri Net (P, T, F){
    F' = Φ;
    FOR EACH element (x,y) in F
        F'=F'+{(y,x)};
    RETURN(P,T,F');
}
```

Figure 3. Pseudo of Building Negotiation Petri Net and Reverse Negotiation Petri Net

Figure 3 presents the pseudo code the algorithm constructing Negotiation Petri Net and its Reverse Negotiation Petri Net. Given the input of the server policies and the client policies, Negotiation Petri Net $NP = \{P, T; F\}$ can be conformed. Each credential has at most one corresponding place node, and a unique transition node for each operator exists in NP. When a credential appears multiple times in the policies, its corresponding place node has multiple flow relations with other credentials. Consequently, there may exist cycles in Negotiation Petri Net, as the path in Figure 2(a) between C_1 and C_2 along the places C_1, t_3, S_1, t_7, C_2, t_5, S_2, t_{10}, C_1. Based on Negotiation Petri Net, its Reverse Negotiation Petri Net NP' is built by reverse the flow relations in NP.

4. STRATEGY BASED ON NEGOTIATION PETRI NET

The purpose of trust negotiation is to find a safe disclosure sequence $G = (C_1, C_2, \ldots, C_k, S)$, where S is the resource to which access was originally requested, such that when a credential C_i is disclosed, its access control policy has been satisfied by credentials disclosed earlier in the sequence or to determine that no such credential disclosure sequence exists. In Negotiation Petri Net, the place corresponding to the service S is regarded as the root of the net. Based on the definition of the place and the transition node of Negotiation Petri Net, a safe disclosure sequence is in a solution path of Negotiation Petri Net which (1) contains the root S; (2) if node $n \in P$ then it contains one of transition nodes in $\{t_i | (n,t_i) \in F\}$; (3) if node $n \in T$ then it contains all the place nodes in $\{C_i, S_i | (n,C_i) \in F, (n,S_i) \in F\}$; (4) ends with one of the place node with token. The safe disclosure sequence for S is the credentials in the solution path of Negotiation Petri Net starts from the place node with token and ends with S. No safe disclosure sequences exist if one of the conditions does not be satisfied. Although the solution path is defined in Negotiation Petri Net, it is searched in the Reverse Negotiation Petri Net in practice. A reverse solution path is got if there is and the credentials ordered by the reverse solution path compose a safe disclosure sequence.

Theorem 1. In Reverse Negotiation Petri Net $NP' = \{P,T,F'\}$, the transition t_x ($t_x \in T$) will be fulfilled if and only if all the place in $\{C_i | C_i \in P, (C_i,t_x) \in F'\}$ have got the tokens. The place C_y ($C_y \in P$) will get token if one of transitions in $\{t_i | t_i \in T, (t_i,C_y) \in F'\}$ has been fulfilled.

Proof. By the definition of Negotiation Petri Net, the flow relations flow from a transition denote the $\wedge$ operation, which means that the transition can be fulfilled only when all the children places are satisfied. The flow relations flow from a place denote the $\vee$ operation, which means that the place can get token if one of its children transition is fulfilled. The Reverse Negotiation Petri Net NP' has the reversed flow relation directions to Negotiation Petri Net, but the same meaning of the places and transitions. Therefore, in Reverse Negotiation Petri Net, a transition can be fulfilled only when all the parent places of it have got tokens, and a place will get a token if one of its parent transitions has been fulfilled. □

For example, in Figure 2(b), C_4 is a *TRUE* credential and a token is set originally in the place node C_4. Transition t_9 will be fulfilled as there is only one flow relation in F' flows into t_9. However, transition t_2 and t_7 cannot be fulfilled because they must wait until C_2 has got the token. Suppose at a stage, C_2 gets token, and is followed by transition t_2 being fulfilled, then S will get the token as $(t_2, S) \in F$'. Once S gets the token, a reverse solution path of Negotiation Petri Net is set and a safe disclosure sequence for S is obtained.

Some places, transitions or flow relations in F' can be deleted safely during the reverse solution path search process in the Reverse Negotiation Petri Net, for example, the *FALSE* credentials with denial policies, such as the credential C_6 in Figure 1, as these credentials have no contributions to the reverse solution path. Details are shown in Theorem 2.

Theorem 2. In Reverse Negotiation Petri Net $NP' = \{P, T, F'\}$, let C_x be a place, t_i is a transition in $\{t_i \mid (C_x,t_i) \in F'\}$, t_j is a transition node in $\{t_j \mid (t_j,C_x) \in F'\}$.

If C_x is a place corresponding to a *FALSE* credential, then it is safe to delete t_i and all the flow relations that flow into t_i and flow from t_i.

If C_x is a place with token, then it is safe to delete all the flow relations ended by C_x, all the transitions in $\{t_j \mid (t_j,C_x) \in F'\}$ and the flow relations which flow into t_j.

If there is no flow relations flow from C_x ($C_x \neq S$), then it is safe to delete all the flow relations ended by C_x, all the transition nodes in $\{t_j \mid (t_j,C_x) \in F'\}$ and the flow relations which flow into t_j.

Proof. From the assumption, we know that in Reverse Negotiation Petri Net, all the transitions in $\{t_i \mid (C_x,t_i) \in F'\}$ are end nodes of flow relations started from C_x, and all the transitions in $\{t_j \mid (t_j,C_x) \in F'\}$ are start nodes of flow relations ended by C_x.

(1) Since credential C_x is *FALSE*, none solution path will pass through C_x, then the place C_x can be erased from Reverse Negotiation Petri Net. By Theorem 1, all the transitions t_i in $\{t_i \mid (C_x,t_i) \in F'\}$ will not happen, therefore these transitions can be deleted from T. Since transition t_i will never exist in Reverse Negotiation Petri Net, all the flow relations that flow into t_i and flow from t_i must be deleted.

(2) Since credential node C_x has a token, C_x is a granted credential. Once the other party has granted a credential, the credential can be visited at anytime. In Reverse Negotiation Petri Net, it is unnecessary to keep the flow relations ended by C_x, then all the flow relations $\{(t_j,C_x) \in F'\}$ can be deleted. By the characteristic of Reverse Negotiation Petri Net, each transition has only one child place. Since the flow relation started from the transition node t_j has been erased, the transition t_j can be pruned and all the flow relations that flow into t_j must be deleted.

(3) The destination of the solution path is the place S and there are at least one flow relation which flows into it but none flow relation flowing from S in Reverse Negotiation Petri Net. If a place node C_x is not S but with no flow relations flow from it, the place C_x is not a contributive place to achieve S and it can be pruned. Since C_x has been deleted, all the flow relations ended by C_x: $\{(t_j,C_x) \in F'\}$ must be deleted. As the explained in (2), the transition node t_j can be pruned and all the flow relations that flow into t_j must be deleted. □

For example, in Figure 2(b), C_6 is a place corresponding to a *FALSE* credential, then t_1 can be erased from the figure, followed by erasing the flow relations (C_6, t_1), (C_1, t_1), (t_1, S).

Theorem 3. In Reverse Negotiation Petri Net NP', if none of transition t_i in T satisfy: all the place nodes in $\{C_j \mid (C_j, t_i) \in F'\}$ have tokens, the reverse solution paths do not exist.

Proof. Tokens flow from a place to another place through the transition between them. By Theorem 1, the transition t_x will happen if and only if all the place in $\{C_i \mid (C_i,t_x) \in F\}$ have got the tokens, which means that a transition can not be fulfilled if one of its parent places does not have token. Here we call the transition is in the Waiting Position. If all the transitions in T are in the Waiting Position, no tokens will flow in the net. Therefore no tokens will reach S, the destination of the reverse solution path, and safe disclosure sequence do not exist. □

```
input: Reverse Negotiation Petri Net NP'={P, T, F'};
       service S originally requested by the client;
output: the safe disclosure sequence for S
Try-Search(NP', S){
    TokenSet={C_i| C_i has token};
    FlowSet= Φ;
    FOR EACH (place C_i= FALSE in P)
        DeleteFalsePlace(C_i);   // according to the first part of Theorem 2
    SeachSafeDisSeq(TokenSet, FlowSet);
    RETURN(credentials in FlowSet);
}

SeachSafeDisSeq(TokenSet, FlowSet){
    SatTranSet = Φ;
    FOR EACH (t_i in T)
        IF (Fulfill(t_i)) SatTranSet = SatTranSet + {t_i};
    WHILE ( S ∉ TokenSet  and  SatTranSet ≠ φ){
        t = SelectTran(SatTranSet);
        FlowSet = FlowSet + {flow relations ended by t};
        FOR EACH (place C_j in P)
            IF((t,C_j) ∈ F'){
                add tokens to places C_j;
                TokenSet = TokenSet + {C_j};
                FlowSet = FlowSet + {(t, C_j)};
                DeleteTran(t);
                DeleteFlowEndbyPlace(C_j); // according to the second part of Theorem 2
                IF(C_j = S)  RETURN(FlowSet);  }
        LoneSet = {the credentials which have no flow relations flow from them};
        FOR EACH (place C_i in LoneSet)
            DeleteFlowEndbyPlace(C_i); // according to the third part of Theorem 2
        FOR EACH (t_i in deleted T)
            IF (Fulfill(t_i)) SatTranSet = SatTranSet + {t_i};
    }// end while
    IF(SatTranSet = Φ){// according to Theorem 3
        FlowSet = Φ;
        RETURN(FlowSet); }
}
```

Figure 4. Pseudo of Key Functions of SNPN

The purpose of SNPN is to find a reverse solution path and return a safe disclosure sequence. Based on the theorems analyzed before, SNPN works as follows. Firstly, the FALSE credentials and their related transitions and flow relations are deleted

according to Theorem 2(1) as they will never appear in the reverse solution path. Then the search process begins. Based on the currently disclosed credentials (the disclosed credentials have tokens in Reverse Negotiation Petri Net), the transitions, which can be fulfilled, are collected and form a fulfilled-transition set. If the fulfilled-transition set is empty, according to Theorem 3, the search exits and returns that no solution exists. Otherwise, a transition in the fulfilled-transition set is selected and tokens flow from places to other places across the transition, followed by deleting the places, transitions and flow relations according to second and third part of the Theorem 2. The process goes on until none solution has been found, or S has got the token and a safe disclosure sequence has been returned. Figure 4 gives the pseudo code of two key function of SNPN.

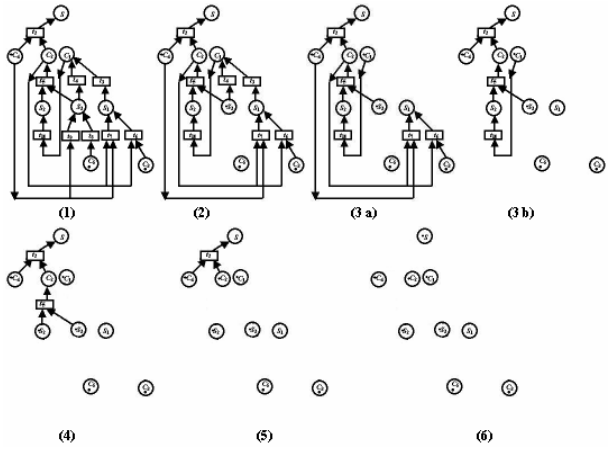


Figure 5. Each Stage of Reverse Negotiation Petri Net of Figure 1 during SNPN Works

Figure 5 gives each stage of Reverse Negotiation Petri Net of figure 1 during SNPN works. In stage (1), the *FALSE* credentials and its related transitions and flow relations are deleted from Reverse Negotiation Petri Net according to Theorem 2(1). Entering the search process, as C_3, C_4 and C_5 have tokens, transition t_8, t_9 can be fulfilled and they are added into the fulfilled-transition set. According to the order of transitions in set, t_8 is selected and a token is added to S_3 across t_8 from C_5. Then by Theorem 2(2), transitions t_8, t_9, and flow relations (C_4, t_9), (t_9, S_3), (C_5, t_8), (t_8, S_3) are deleted, as shown in stage (2). Now, C_3, C_4, C_5 and S_3 have tokens, transition t_4 can be fulfilled and a token flows from S_3 to C_1 through t_4. After C_1 has got a token and finished the delete process (stage (3a)), S_1 is a credential which has no flow relations flow from it, then according to Theorem 2(3), its related transitions and flow relations are deleted, as shown in stage (3b). The search process goes on until S has got a token and return the safe disclosure sequence (C_3, S_3, C_1, S_2, C_2, C_4, S).

5. ANALYSIS OF SNPN

As discussed before, an auto trust negotiation strategy must be complete, reasonably efficient, and avoidable disclosing any credentials that are not needed for the successful negotiation. In this section, we will analyze SNPN from the three aspects.

SNPN is a complete strategy. Firstly, from the construction of Negotiation Petri Net and Reverse Negotiation Petri Net, all credentials and policies are mirrored in the net by the places, transition and flow relations. Secondly, although some places, transitions and flow relations are deleted during the search process, Theorem 2 assures the safety and the deletions are unaffected to finding a safe disclosure sequence. Furthermore, the scan of transition in the remained transitions based on Theorem 3 ensures that any chance of granting credentials will not be missed. So SNPN can find a safe disclosure sequence whenever such a sequence exists, the strategy is a complete strategy.

The efficiency of a strategy includes two aspects: the computational cost and the communication cost.

Computation cost is computed in building model process, negotiation process and credential exchange process. Negotiation Petri Net is built based on the policies of both parties with total n credentials and m policies size. Each policy contains n credentials at most, and the computation cost of building Negotiation Petri Net is $O(nm)$ in the worst case. Reverse Negotiation Petri Net is constructed by reversing the flow relations in Negotiation Petri Net and the computation cost is linear to Negotiation Petri Net. In the negotiation process, each transition is visited at most one time, as it will be deleted after the vision. Each credential is disclosed once, as other flow relations flow into it are deleted, which effectively avoids cyclical request for the same credential. So the computational cost in this phase is at most $O(nm)$. The credentials exchange ordered by the safe disclosure sequence found in the negotiation phase, and the computational cost of credential exchange phase is at most $O(n)$. Therefore the total computational complexity is $O(nm)$.

The communication cost includes both the total size of the messages and the total number of messages sent between the two security agents. Negotiation Petri Net is built on the received messages and only the requested messages are sent between the client and server. SNPN works so far as a safe disclosure sequence is found or not and the total number of request messages is $O(n)$ in the worst case. Since each request message only contains a credential name, the total size of request messages is also $O(n)$ in the worst. Credential exchange starts after SNPN has found a safe disclosure sequence. During this phase, there are at most n messages and the size of each message depends on the size of the credential (which we assume is bounded by a constant). In all, the worst case communication cost is $O(n)$ in the sense of both the total number of messages and the total size of messages.

In SNPN, credential exchanges begin until both parties know there exists a successful negotiation. Only the credentials in the safe disclosure sequence are disclosed, which means SNPN will avoid disclosing any credentials that are not needed for the successful negotiation.

6. CONCLUSION AND FUTURE WORK

In this paper we model the policies participating negotiation as Negotiation Petri Net and propose a trust negotiation *Strategy based on Negotiation Petri Net* (SNPN). Negotiation Petri Net is built by combining the characteristic of Petri Net architecture with the behavior of policies in auto trust negotiation, and the modeling complexity is $O(nm)$ which is sharp contrast with previous proposed modeling methods with exponential time, where n is the total number of credentials requested and m is the size of credential disclosure policies. Based on Negotiation Petri Net, SNPN is a complete and efficient automated trust negotiation strategy. SNPN guarantees to find a successful credentials

disclosure sequence whenever the credential policies of the service requester and provider allow. We also proved that, in the worst case, the communication cost and computational complexity of SNPN are $O(n)$ and $O(nm)$. Meanwhile, SNPN ensures that no irrelevant credentials are disclosed in the negotiation process.

We assume all the credentials are of equal importance in this paper. However, this assumption is not true in many situations. For example, ones credit card number or social security number is often much more sensitive than her home phone number. It is desirable to establish trust by exchanging the least sensitive credentials possible. Therefore, besides completeness and minimal credential disclosure, minimum total sensitivity of disclosed credentials might also be a desired feature of a negotiation strategy.

7. REFERENCES

[1] Milojicic D et al. Peer-to-peer (P2P) technology. HP Labs Technical Report, HPL-2002-57, http:□www.hpl.hp.com/techreports/2002/HPL-2002-57.html, 2002. Submitted to ACM Computing Surveys.

[2] . Foster, C. Kesselman. The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2004.

[3] Winsborough WH, Seamons KE, Jones VE. Automated trust negotiation. DARPA Information Survivability Conf. and Exposition. New York: IEEE Press, 2000. 88-102. http://isrl.cs.byu.edu/pubs/ discex2000.pdf.

[4] William H. Winsborough, Ninghui Li. Towards practical automated trust negotiation. In Proc. of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), pages 92– 103, June 2002.

[5] E. Bertino, E. Ferrari, A. Squicciarini. Trust Negotiations: Concepts, Systems, and Languages. IEEE Computing in Science and Eng., vol. 6, no. 4, 27-34, 2004.

[6] Huraisingham, Bhavani. Secure Knowledge Management: Confidentiality, Trust, and Privacy. IEEE. Transactions on Systems, Man & Cybernetics, Vol.36, Issue 3. May2006, 429-438.

[7] J. Li, N. Li, W. H. Winsborough. Automated trust negotiation using cryptographic credentials. In: Proc. of the 12th ACM conf. on Computer and communications security. New York: ACM Press, 2005. 46 – 57.

[8] Anna Cinzia Squicciarini, Elisa Bertino, Elena Ferrari, Indrakshi Ray. Achieving Privacy in Trust Negotiations with an Ontology-Based Approach. IEEE Transaction Dependable Section Computer, 3(1): 13-30 (2006).

[9] Bertino E,Ferrari E,Squicciarini A. Trust-X: A Peer-to-Peer framework for trust establishment. IEEE Transaction on Knoledge and Data Engineering, 2004, 16(7).

[10] Yu T, Ma X, Winslett M. PRUNES: An efficient and complete strategy for trust negotiation over the Internet. In: Proc. of the 7th ACM Conf. on Computer and communications Security. New York: ACM Press, 2000. 210-219. http://www4.ncsu.edu:8030/~tyu/pubs/ccs2000.pdf.

[11] Yu T. Automated trust establishment in open systems [Ph.D. Thesis]. Illinois: University of Illinois, 2003.

[12] Yu T, Winslett M, Seamons KE. Interoperable strategies in automated trust negotiation. In: Proc. of the 8th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2001. 146-155. http://isrl.cs.byu.edu/pubs/ccs2001.pdf.

[13] Yu T, Winslett M, Seamons KE. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Trans. on Information and System Security, 2003,1(6): 1-42.

[14] Weifeng Chen, L. Clarke, James F. Kurose, Donald F. Towsley. Optimizing cost-sensitive trust-negotiation protocols. In: Proc. of the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies. Miami, FL, USA. 13-17 March 2005. 1431-1442.

[15] Song Ye, Fillia Makedon, James Ford. Collaborative Automated Trust Negotiation in Peer-to-Peer Systems. In: Proc. of the Fourth International Conf. on Peer-to-Peer Computing (P2P'04), Aug. 2004.

[16] T. Murata. Petri Nets: properties, analysis and applications. In: Proc. of the IEEE, 77(4):541–579, Apr. 1989.

[17] W.Winsborough, N. Li. Protecting Sensitive Attributes in Automated Trust Negotiation. ACM Workshop on Privacy in the Electronic Society, Washington, DC, Nov. 2002.