A Dynamic Programming Approach to Automated Trust Negotiation for Multiagent Systems

Hirofumi Yamaki, Masao Fujii, Kousuke Nakatsuka, and Toru Ishida

Kyoto University, Dept. of Social Informatics,
Yoshida-Hommachi, Kyoto, 606-8501 JAPAN
{yamaki, mfujii, nakatuka, ishida}@kuis.kyoto-u.ac.jp

Abstract. We consider a framework where agents perform trust formation on behalf of human users in open environment like pervasive computing or the Internet. In this paper, an automated trust negotiation framework is proposed to support such trust agents, and a protocol is presented. In the framework, the preference for the disclosure of credentials and policies are explicitly included in contrast to ordinary ATN frameworks studied in network authentication domain. The proposed protocol is based on dynamic programming and successfully performs the exchange of credentials and policies without unnecessary disclosure of them.

1. Introduction

In this paper, we discuss on trust formation among agents in open environments, e.g., pervasive computing or the Internet, where users, service providers, or agents are often mutually strangers. Before interacting with each other, they must somehow decide whether to trust the counterpart.

Suppose a case where a person is to buy some good in a net shop. Before buying it, he/she will read information in the Web site, such as the identity of the shop, its payment or returning policy etc. to decide whether to trust the shop. This is because there is a risk in giving money or personal information to a completely unknown party. By getting information beforehand, he/she tries to minimize such a risk. Also, the net shop may require his/her information, such as credit card number or telephone number, not only to deliver the good but also to avoid troubles. Generally, a trust formation process includes the bilateral exchanges of information as shown in the above example.

Techniques like authentication and cipher, which are studied in security domain, provide various ways to connect mutually trusting parties. Trust formation is required before applying them. For instance, before authenticate a person, one must decide whether he/she is trusted. Also, ciphers assume the owner of the key is trusted. Trust formation precedes the application of these security techniques.

Currently, trust formation is usually performed by humans, and the decision is often static, e.g., password files for online systems are given by human administrator, and are static for authentication modules. This approach is simple and easily implemented in relatively small and static environments.
However, this is not enough in large scale, open environments. In the net shop example given above, various users can visit various shop with various identities and policies. And, in each case, one must decide whether to trust the counterpart. It is almost impossible to know which user visits which shop, thus trust formation cannot be given in static form beforehand. Since trust formation is relatively costly decision process, users tend to repeatedly visit the same shop just to avoid it even if it is not the best choice. This may be preferable for the owners of the shop chosen in the first place, but not efficient from the economic point of view.

In open pervasive/ubiquitous computing environment, this problem is more serious. Consider a case where many services are provided by different service providers in open space. Without some common authentication platform, users have to perform trust formation for each of the services, every time they move to different places. This is almost unbearable cost for human users.

There are several approaches to this problem. One of the simplest approaches is, obviously, to have a single, large-scale, common platform to authenticate all of the users and services. But, this may not scale and is often impossible to build.

Instead, we propose a trust agent approach, where each agent automatically performs trust formation on behalf of its human owner. Each trust agent knows the information about its owner, exchanges it with other trust agents, and automatically decide whether to trust the counterpart. By augmenting pervasive computing environments with trust agents, users can make maximum use of services around without suffering for the cost of manual trust formation.

Here, we have to consider the security aspect of personal information that trust agents handle. Human owners want their agents to resolve the trust formation problems, but, simultaneously, do not want them to tell delicate personal information to every agent they interact with in open environment, where malicious agents may be around. They may want to tell the agents some policies that is used for what information may be disclosed at what condition.

In literature of authentication techniques in networks, the process where automated entities exchange information conforming disclosure policies is called automated trust negotiation (ATN)[1], where strangers exchange digital credentials each other to establish mutual trust conforming their disclosure policies. Though a number of protocols are proposed, they lack the ability to reflect agent-owner’s preference, such as “I would disclose my ‘customer ID’ rather than my name.” This is usually unimportant in the original context of ATN, but is sometimes critical to be used in multiagent systems.

Below, we propose an ATN protocol for multiagent systems that reflect user preference , where a variation of distributed dynamic programming is applied to avoid the disclosure of unnecessary information by exchanging minimal number of credentials.
2. Automated Trust Negotiation

As discussed above, automated trust negotiation (ATN) is a method to establish trust among mutual strangers, by exchanging credentials, a set of data with proof that it is truly of the owner, and policies, that describe the conditions to disclose each credential.

2.1. Framework of ATN

A negotiation is a bilateral, iterative process where the disclosure of credentials and policies is performed in turn among agents. Disclosure of a credential or a policy means to explicitly send it to the counterpart. The players of a negotiation are two agents: Client and Server. Below, we call the owner of Client as User, and Server as Service Provider, respectively. Each agent has its own credentials and policies. Server has services and service policies, which describe the condition to provide the services. Each agent does not know what credentials and policies the other agent has before starting a negotiation.

A negotiation is initiated when Client requests service $R$ to Server. A negotiation is successful and service $R$ is provided to Client if the service policy for service $R$ is finally fulfilled after iteration of the discloser of credentials and policies by Client and Server. In contrast, if the service policy for service $R$ is not fulfilled, the negotiation is failed and service $R$ is not provided to Client.

A credential of an agent is information that proves a feature of the owner. Each credential is digitally signed with a certificated secret key of the issuer, and is authenticated with the corresponding public key.

Table 1 gives an example of credentials and services of Client and Server.

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (of User)</td>
<td>Ticket Reservation Service</td>
</tr>
<tr>
<td>Telephone Number (of User)</td>
<td>Address (of Service Provider)</td>
</tr>
<tr>
<td>Credit Card Number (of User)</td>
<td>Certificate of Card Company</td>
</tr>
</tbody>
</table>

A policy describes the set of credentials that are required to disclose a credential or to provide a service. A policy for credential $C$ is formalized as $C \leftarrow F_c(S_1,\ldots,S_k)$, where $F_c(S_1,\ldots,S_k)$ is a logical form with credentials $S_1,\ldots,S_k$ and operators $\cap$ and $\cup$. When the counterpart discloses credential $S_i$, the value of the variable $S_i$ in the logical expression becomes true. When logical value of expression $F_c(S_1,\ldots,S_k)$ becomes true, the policy for $C$ is fulfilled, $C$ becomes discloseable. For example, when the policy for credential $C$ is $C \leftarrow (S_1 \cap S_2) \cup S_3$, it is fulfilled and the credential becomes discloseable when the counterpart discloses both $S_1$ and $S_2$, or discloses $S_3$. 
When a credential $C$ is disclosable without requiring the disclosure of any other credential, the policy for $C$ is written as $C \leftarrow true$. In this case, credential $C$ is said to be unprotected. Also, when an agent does not have credential $C$, or when it does not disclose $C$ in any cases, the policy for $C$ is written as $C \leftarrow false$, and is called as denial policy. An agent tacitly has denial policies for credentials that it does not have.

Table 2 gives an example for policies of Client and Server for credentials and services in Table 1.

### Table 2. An Example of Policies

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name $\leftarrow true$</td>
<td>Ticket Reservation Service</td>
</tr>
<tr>
<td>Telephone Number $\leftarrow Address$</td>
<td>$\leftarrow (Name \bigcap \text{Telephone Number})$</td>
</tr>
<tr>
<td>Card Number $\leftarrow \text{Certificate of Credit Corporation}$</td>
<td>$\bigcup \text{Credit Card Number}$</td>
</tr>
<tr>
<td></td>
<td>Address $\leftarrow true$</td>
</tr>
<tr>
<td></td>
<td>Certificate of Credit Corporation $\leftarrow true$</td>
</tr>
</tbody>
</table>

#### 2.2. Example of Negotiation

Table 3 gives an example of negotiation when Client and Server has credentials, services, and policies in Tables 1 and 2. In Table 3, the process of negotiation proceeds from the top to the bottom. First, when Client requests Server for ticket reservation service, Server tells the Client that it needs either Name and Telephone Number or Credit Card Number, by disclosing the policy for the service. Client then disclose the policy for Credit Card Number to tell Server that it needs Certificate of Credit Corporation. Since Certificate of Credit Corporation is unprotected, Server discloses it to Client. Here, it is obvious to Client that this credential is unprotected, the policy is also regarded as disclosed. Since Certificate of Credit Corporation is disclosed, the policy for Credit Card Number is fulfilled, and thus Client discloses Credit Card Number to Server. Then, the policy for ticket reservation service is fulfilled, the trust between Client and Server is established, and the service is provided to Client. The negotiation finishes at this point.

### Table 3. Example of Negotiation

<table>
<thead>
<tr>
<th>Agent</th>
<th>Disclosed credentials/policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Ticket Reservation Service $\leftarrow (Name \bigcap \text{Telephone Number}) \bigcup \text{Credit Card Number}$</td>
</tr>
<tr>
<td>Client</td>
<td>Credit Card Number $\leftarrow \text{Certificate of Credit Corporation}$</td>
</tr>
<tr>
<td>Server</td>
<td>Certificate of Credit Corporation $\bigcap \text{Certificate of Credit Corporation} \leftarrow true$</td>
</tr>
<tr>
<td>Client</td>
<td>Card Number</td>
</tr>
<tr>
<td>Server</td>
<td>Ticket Reservation Service</td>
</tr>
</tbody>
</table>
2.3. Unnecessary Disclosure of Credentials and Policies

Credentials disclosed during the process of negotiation include information whose unnecessary disclosure may be harmful. Policies also can include such dangerous information.

Thus, from the viewpoint of security and privacy, the minimal disclosure of credentials and policies is preferred. In this paper, we assume that agents cooperatively minimize the total sum of the disclosure performed by both of them. This assumption is reasonable in some cases, but, in other cases, we may have to analyze the cases where agents are selfish and does not behave cooperatively.

3. Extending ATN Framework

3.1. Avoiding Unnecessary Disclosure of Policies Concatenated by OR

In ATN, unnecessary disclosure of policies is as undesirable as that of credentials. For example, the policy for providing ticket reservation service in Table 2 includes two conditions, i.e., “Name and Telephone Number” and “Credit Card Number” which are concatenated by OR. In the negotiation given in Table 3, both conditions are disclosed, though the condition “Name and Telephone Number” is unnecessary in the following process. This is mainly because the original definition of ATN allows only one policy and connects multiple conditions using OR to include them in one policy.

Instead, in this paper, we allow multiple policies for one credential and only one condition for one policy. Formally, a credential \( C \) has \( n \) policies, \( P_1, \ldots, P_n \), where \( P_i \) is written as \( C \leftarrow F_i(S_1, \ldots, S_k) \). \( F_i(S_1, \ldots, S_k) \) is a logical expression that consists of \( S_1, \ldots, S_k \) and operator \( \bigcap \). In this definition, the ticket reservation service has two policies, “Ticket Reservation Service \( \leftarrow \) Name \( \bigcap \) Telephone Number” and “Ticket Reservation Service \( \leftarrow \) Credit Card Number,” which can be disclosed separately.

Below, to simplify the problem, we assume that there is no loop in the policy set that agents have, such as “\( C_1 \leftarrow C_2 \)” and “\( C_2 \leftarrow C_1 \).” This type of loop is easily detected if necessary and does not affect the result of this work.

Also, we assume that there is no confluence, such as “\( C_1 \leftarrow S_1 \bigcap S_2 \),” “\( S_1 \leftarrow C_2 \),” and “\( S_2 \leftarrow C_2 \).” In contrast to the above, this assumption is strong, and should be removed in the future works.
3.2. Introducing Preference of Humans to ATN

Disclosure of credentials and policies by agents should reflect the preference of User and Service Provider, such as “I would disclose Credit Card Number rather than Telephone Number (if any other condition does not change).” In the original definition of ATN, such preference is not included and the value of disclosure is the same.

In this work, we introduce human preference by assuming that the disclosure of each credential or policy requires some cost. Cost is a real number larger than zero, and larger cost means larger (mental) barrier to disclose the corresponding credential or policy. For instance, if the cost for disclosing Telephone Number is 3 and that for Credit Card Number is 1, it is more preferable for User to disclose the latter. If credential $C$ has denial policy $C \leftarrow false$, the cost to disclose $C$ is $+\infty$. When multiple credentials and policies are disclosed, the total cost is the sum of the disclosure of each credential and policy. Below, we call the sum of the cost to disclose credentials and policies in a negotiation as just “the cost of negotiation.”

<table>
<thead>
<tr>
<th>Table 4. Cost for Disclosing Credentials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
</tr>
<tr>
<td>C1</td>
</tr>
<tr>
<td>C2</td>
</tr>
<tr>
<td>C3</td>
</tr>
<tr>
<td>C4</td>
</tr>
<tr>
<td>C5</td>
</tr>
<tr>
<td>C6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 5. Cost for Disclosing Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy</td>
</tr>
<tr>
<td>C1 $\leftarrow$ S1 $\cap$ S2</td>
</tr>
<tr>
<td>C2 $\leftarrow$ S2</td>
</tr>
<tr>
<td>C2 $\leftarrow$ S3 $\cap$ S4</td>
</tr>
<tr>
<td>C3 $\leftarrow$ true</td>
</tr>
<tr>
<td>C4 $\leftarrow$ true</td>
</tr>
<tr>
<td>C5 $\leftarrow$ false</td>
</tr>
<tr>
<td>C6 $\leftarrow$ true</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
In this framework, the objective, avoiding unnecessary disclosure of credentials and policies, is modeled as the minimization of the cost of negotiation. Table 4 gives an example of costs for disclosing credentials, and Table 5 gives that for disclosing policies.

4. Negotiation Graphs

Here, we introduce the notion of negotiation graph, which a special type of AND/OR graph, and each instance of negotiation graph corresponds to a problem of ATN. In this framework, each negotiation is modeled as a process of distributed search of negotiation graph performed by Client and Server agents.

A negotiation graph is an AND/OR graph, where each node corresponds to the disclosure of a credential. The service provided by Server corresponds to the start node, and unprotected credentials correspond to the terminal node. For each of credentials that appear in the right side of a policy, directed edge is given from the node that corresponds to the credential (or the service) that appears in the left side to the node that corresponds to this credential. By doing this, a policy corresponds to a set of directed edges, which is called the connector for the policy.

The cost for disclosing credentials and policies are given as the cost of corresponding nodes and edges. The cost of a connector is defined as the sum of the cost for disclosing corresponding policy $P$ and the cost for disclosing all the credentials that appear in the right side of $P$.

Figure 1 gives the negotiation graph that corresponds to the example given in Tables 4 and 5.

![Negotiation Graph for Example in Tables 4, 5](image)

4.1. Distributed Search in Negotiation Graph

As mentioned above, a negotiation is modeled as a process of distributed search in a negotiation graph. Specifically, negotiation corresponds to finding a solution tree in the negotiation graph. The root of the
solution graph has to be the starting node (service) in the negotiation graph, and all the leaf nodes have to be the terminal node (unprotected credential). If a solution tree is found, the negotiation is successful.

The minimization of the cost for negotiation is achieved by finding the minimal solution tree without disclosing credentials and policies that is not included in the solution tree, or determining that there is no solution tree without disclosing any credentials or policies.

The characteristic of this problem is in that none of Client and Server knows the whole image of the negotiation graph. For example, Client does not know what policies Server has, thus it does not know whether there are edges that correspond to them. Because of this, no agent can reconstruct a complete negotiation graph like that depicted in Figure 1. Figure 2 shows the part Client can observe, and Figure 3 shows the part Server can observe.

Without any communication between Server and Client, it is just impossible to find a solution tree. Below, we present a protocol that achieves this without disclosing unnecessary credentials and policies.
5. ATN Protocol Based on Dynamic Programming

The basic idea is to have agents exchange the estimated cost of current partial solution instead of directly disclosing credentials and policies, which is not always necessary in finding a solution tree in negotiation graph.

Before describing the protocol, we define several notations and show theorems according to them.

Definition 1.

\( h(v) \) is the cost of the minimal tree with root node \( v \).

Definition 2.

Disclosure of node \( v \) means to send the value of \( h(v) \) to the counterpart of the negotiation.

Definition 3.

OPEN set is a set of already disclosed nodes. Each of Client and Server has its own OPEN set, and neither can observe the content of OPEN set of the other.

Definition 4.

The following messages are exchanged between agents.

- **Success**: Message to tell that a minimal solution tree has successfully determined.
- **Failure**: Message to tell that there is no solution tree in the given negotiation graph.
- **Null**: Message to tell the OPEN set is empty.
- **NodeDisclosure**: Message to disclose a node, written as \( \{v, h(v)\} \).
- **Solved**: Message to send the nodes in the minimal solution tree. For nodes in the minimal solution tree \( v_1, \ldots, v_n \), this message is written as \( \{v_1, \ldots, v_n\}_{Solved} \).
- **Terminal**: Message to disclose a terminal node. For terminal node \( t \), this message is written as \( \{t, g(t)\}_{Terminal} \).

Theorem 1.

When a node \( v \) is a non-terminal node and has connectors \( \theta_1, \ldots, \theta_m \) as its children, the cost for the minimal solution tree that has \( v \) is determined by

\[
h(v) = \min_{1 \leq i \leq m} \left\{ d_i + \sum_{k=1}^{N_i} h(v_{ik}) \right\},
\]

where \( d_i \) is the cost for connector \( \theta_i \) and \( v_{i1}, \ldots, v_{iN_i} \) are the children of \( \theta_i \).

Theorem 2.

When a node \( v \) is a non-terminal node and has connectors \( \theta_1, \ldots, \theta_m \) as its children, if \( v \) is a node in the minimal solution tree, nodes \( v_{i1}, \ldots, v_{iN_i} \) that satisfies

\[
h(v) = d_i + \sum_{k=1}^{N_i} h(v_{ik})
\]

are also
in the minimal solution tree, where \( d_i \) is the cost for connector \( \theta_i \) and \( v_{i1}, \ldots, v_{iN_i} \) are the children of \( \theta_i \).

The agents can find a solution graph by the following protocol, which applies to both of Client and Server.

1. In the case where it is the agent’s turn to disclose a node,
   i. If OPEN set is empty, it sends \( \text{Null} \) to the counterpart.
   ii. If OPEN set is not empty, it chooses a node \( v \) from OPEN set, send \( \{v, h(v)\} \) to the counterpart, and removes \( v \) from the OPEN set.

2. In the case where the agent receive Null message,
   i. If OPEN set is empty,
      A) Client sends Null to Server.
      B) Server sends \( \{v_1, \ldots, v_n\}_{\text{Solved}} \) to client after determining \( v_1, \ldots, v_n \) using Theorem 2, if the service \( R \) has already been disclosed. Otherwise, it sends Failure to Client and terminates the search process, without finding any solution tree.
   ii. If OPEN set is not empty, it chooses a node \( v \) from OPEN set, send \( \{v, h(v)\} \) to the counterpart, and removes \( v \) from the OPEN set.

3. In the case where the agent receives Success message, it terminates the search process. A solution tree has found.

4. In the case where the agent receives Failure message, it terminates the search process. No solution tree has found.

5. In the case where the agent receives \( \{V_1, \ldots, V_m\}_{\text{Solved}} \) message,
   i. If all the nodes in \( V_1, \ldots, V_m \) are terminal nodes, it sends Success to the counterpart and terminates the search process. A solution tree has found.
   ii. Otherwise, it extracts non-terminal node \( v_1, \ldots, v_n \) from \( V_1, \ldots, V_m \). For each \( v_j \), the minimal solution tree that has \( v_j \) as its root node and the nodes \( v_{i1}, \ldots, v_{iN_j} \) in the solution tree are determined. Then, the agent sends \( \{v_{i1}, \ldots, v_{iN_j}\}_{\text{Solved}} \) to the counterpart.

6. In the case where the agent receives \( \{R, h(R)\} \) message, all the nodes that have \( h \) larger than \( h(R) \) are removed from OPEN set. The turn to disclose a nodes is changed to the counterpart.
7. In the case where the agent receives \(\{v, h(v)\}\), OPEN set is updated as follows. Suppose \(\theta_1, \ldots, \theta_m\) to be the connectors that has \(v\) as its child node, \(V_i\) to be the parent node of \(\theta_i\), and \(v_{i1}, \ldots, v_{iN_i}\) are the child nodes of \(\theta_i\) other than \(v\). For each \(V_i\),

i. If there is a node that has not yet been disclosed in \(v_{i1}, \ldots, v_{iN_i}\), the agent does nothing.

ii. Otherwise, the agent calculate the value of \(h(V_i)\) based on Theorem 1, then

A) If \(V_i\) is a member of OPEN set, \(h(V_i)\) is updated

B) If \(V_i\) has not yet been disclosed, \(V_i\) is added to OPEN set.

C) If \(V_i\) has been disclosed and the new value \(h(V_i)\) is different from the older one, \(V_i\) is added to OPEN set.

iii. If \(R\) is a member of OPEN set, all the nodes that have \(h\) larger than \(h(R)\) are removed from OPEN set.

iv. If \(R\) has already been disclosed, all the nodes that have \(h\) larger than \(h(R)\) are removed from OPEN set.

After updating OPEN set, the turn to disclose a node is given to the counterpart.

Although this protocol looks fairly complicated, the process is basically a variation of dynamic programming. It basically computes the cost of the minimal solution tree by updating the cost from root node to each node.

This particular version starts the calculation from terminal nodes and updates the value of cost upward. We can define another version that starts from the service node and updates the value downward, though we do not show it because of the limitation of space.

The choice of a node from OPEN set, performed in Steps 1 or 2, has large effect on the performance of the negotiation process. Though we have not yet thorough evaluation, it is clear that the order of choice determines the number of exchanged messages among agents. Basically, choosing the node with the smallest \(h\) gives the smallest number of messages. However, we have also found that it becomes easy to infer the hidden policies based on the exchanged values if the order of choice is fixed. On the contrary, random choice from OPEN set gives the largest anonymity, at the expense of large number of messages.

We have to perform more experiments and analysis to clarify the nature of this trade-off.

6. Conclusion

In this paper, we proposed a framework where trust agent performs trust formation on behalf of human users.
By extending ordinary ATN framework to explicitly support user preference on the disclosure of credentials and policies, it becomes possible to apply methods proposed in ATN studies to the trust formation in multiagent systems.

Each negotiation in ATN framework can be modeled as a characteristic subset of distributed cooperative search on AND/OR graph called negotiation graph, where each agent can observe only a part of the graph and they have to interact with each other to accomplish the search. By modeling ATN as multiagent search, we can apply many techniques developed in multiagent studies to this field. The security restrictions of the problem require the information exchange between agents have to be minimal. This feature makes this problem very interesting from the view point of multiagent negotiation.

The protocol presented in this paper is based on dynamic programming, and it is shown that it achieves minimal disclosure of credentials and policies.

However, in some cases, one can infer other information using disclosed credentials and policies. Because of this, although the explicit disclosure is minimal, we cannot say there is no problem on information leaks in our protocol. Actually, the set of policies of an agent can be inferred from the result of negotiation.

This work is a preliminary work for realizing trust agents for supporting humans in open and dynamic computing environments. Our future work includes the estimation of such information leaks, and the evaluation of the performance of the proposed protocol. Also, we plan to build some agent system that support human users based on our framework.

References