Optimizing Cost-sensitive Trust-negotiation Protocols *

Weifeng Chen, Lori Clarke, Jim Kurose, Don Towsley

Department of Computer Science
University of Massachusetts, Amherst
{*chenwf, clarke, kurose, towsley*}*@cs.umass.edu*

Technical Report 2004-29

Abstract

Trust negotiation is a process that establishes mutual trust by the exchange of digital credentials and/or guiding policies among entities who may have no pre-existing knowledge about each other. Motivated by the desire to disclose as little sensitive information as possible in practice, this paper investigates the problem of minimizing the "cost" of the credentials exchanged during a trust-negotiation protocol. A credential or a policy is assigned a weighted cost, referred to as its *sensitivity cost*. We formalize an optimization problem, namely the *Minimum Sensitivity Cost* problem, whose objective is to minimize the total sensitivity costs of the credentials and policies disclosed by a trust-negotiation protocol. We study the complexity of the Minimal Sensitivity Cost problem and propose algorithms to solve the problem efficiently, in both cases when policies are cost-sensitive and cost-insensitive. A simple Finite State Machine model of trust-negotiation protocols is presented to model various trust-negotiation protocols, and used to provide a quantitative evaluation of the number of exchange rounds needed to achieve a successful negotiation, and the probability of achieving a successful negotiation under various credential disclosure strategies.

keywords: Trust-negotiation protocols, sensitivity cost

*This research has been supported in part by the NSF under grant awards UF-EIES-0205003-UMA and EIA-0080119. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

In an electronic environment (e.g., the Internet, electronic commerce, and digital government) in which entities may have no pre-existing knowledge about each other, trustworthiness is of critical concern. The concept of trust has been addressed within many disciplines. It is complex and multidimensional [5]. A general definition of trust from [8] is that "trust is a legal arrangement in which an individual (the trustor) gives fiduciary control of property to a person or institution (the trustee) for the benefit of beneficiaries." Here, we focus on trust in an electronic environment [5, 7, 10]. We will use the definition in [3] that "trust is usually considered a belief or cognitive stance that could eventually be quantified by a subjective probability." This subjective probability is built upon evidence. In real life, people establish a trust relationship based on paper *credentials*, e.g., an employment ID or a Social Security Number (SSN). These paper credentials act as the foundation upon which a person builds trust with others. Trust relationships in an electronic environment is typically established by exchanging *digital* credentials [16, 22, 25], the analogue of paper credentials. Digital credentials are digitally signed assertions by the *credential issuer* about the *credential owner* [1, 20]. For instance, an X.509 certificate, which contains a digital signature of the issuer, the identity and the public key of the owner, and an expiration date etc., is a common digital credential.

Entities can establish one-direction trust or mutual trust. Most current Internet applications establish one-direction trust, e.g., a client provides information to a server in order for the server to trust the client; it is implicitly or explicitly assumed that the client trusts the server. To make this concrete, consider a customer registering at amazon.com. The customer needs to provide a mailing address, phone number, etc. When the customer purchases a book, he/she also needs to provide credit card information. However, Amazon does not need to authenticate itself to the customers. In one-direction trust, one of the negotiators, generally the server, is assumed to be trusted and only the other negotiator (the client) needs to provide authentication information. However, establishing mutual trust is often desirable in electronic environment. This is typically achieved through the process of *trust negotiation* [21]. For example, in the DIAMETER protocol [4], a client seeking access to network resources in dial-up PPP, wireless AP, or Mobile IP environment, exchanges Capabilities-Exchange-Request and Capabilities-Exchange-Answer messages with a network access server in order to negotiate a "mutually acceptable service" based on their capabilities.

During trust negotiation, an entity may not want to disclose credentials freely, since credentials can be sensitive. An *access control policy* (*policy*, for short) for a credential consequently specifies the prerequisite conditions that must be satisfied in order for that credential to be disclosed. For example, a customer may have a policy for disclosing his/her SSN that specifies that the SSN will only be disclosed when an

authorization certificate issued by the Social Security Administration is received.

In real life, people treat their paper credentials with different levels of sensitivity. For example, an SSN will be more sensitive than a telephone number. Given multiple credential-exchange sequences achieving a same result, it is desirable to pick the sequence that discloses a set of less sensitive credentials. For example, when a customer is asked to disclose either a telephone number or an SSN, the customer would likely choose the former. Policies themselves may also be considered sensitive [2, 14, 24]. We can thus associate a cost or weight with each credential or policy. A credential with a high cost is more sensitive. We can then define the *sensitivity cost* to be the total cost of the disclosed credentials and policies in a particular exchange sequence.

In this report, we formulate and study the Minimal Sensitivity Cost problem of minimizing the total sensitivity cost of credentials and policies disclosed during a trust-negotiation protocol's execution. When policies have no sensitivity cost (i.e., they can be freely disclosed), the Minimal Sensitivity Cost problem is shown to be **NP**-complete. Fortunately, we find that heuristic algorithms based on Dijkstra's algorithm perform quite well, achieving around 95% of optimal for the cases considered. When policies themselves have a sensitivity cost, solving the Minimal Sensitivity Cost problem becomes even more computationally complex. Thus, we consider a greedy algorithm to solve this problem approximately. We also describe a Finite State Machine model that provides a simple framework for analyzing the number of exchange rounds needed to achieve a successful negotiation, and the probability of achieving a successful negotiation under various credential-disclosure strategies.

The rest of the report is organized as follows. In Section 2, we briefly overview the literature of trust negotiation. We formulate the Minimal Sensitivity Cost (MSC) problem in Section 3. Section 4 attacks the MSC problem when policies have no disclosure cost. Section 5 is devoted to solving the MSC problem when policies are themselves sensitive. We discuss related optimization problems in Section 6. A FSM model is described in Section 7. Section 8 presents related work. Finally, we conclude the paper in Section 9.

2 Background

During trust negotiation, the disclosure of a credential s is guided by an access control policy p_s that specifies the prerequisite conditions that must be satisfied in order for credential s to be disclosed. Typically, the prerequisite conditions are a set of credentials $\mathcal{C}' \subseteq \mathcal{C}$, where $\mathcal{C}$ is the set of all credentials.

In this paper, policies are modelled using propositional formulas. Specifically, for each credential $c_i \in \mathcal{C}$, we introduce a boolean variable x_i. Every policy p_s has the form: $p_s : s \leftarrow \phi_s(x_1, \ldots, x_k)$ where

$\phi_s(x_1, \ldots, x_k)$ is a normal formula consisting only of literals x_i, the Boolean operators $\vee$ and $\wedge$, and parentheses as needed[1]. s is referred to as the *target* of p_s, and $\phi_s(x_1, \ldots, x_k)$ the *condition* of p_s.

Given a set of credentials $\mathcal{C}' \subseteq \mathcal{C}$, we denote $g_{\phi_s}(\mathcal{C}')$ as the value of $\phi_s(x_1, \ldots, x_k)$ given $x_i = 1 \Leftrightarrow c_i \in \mathcal{C}'$. For example, if $\phi_s = (x_1 \wedge x_3) \vee x_2$, then $g_{\phi_s}(\{c_1, c_2, c_4\}) = 1$ and $g_{\phi_s}(\{c_1, c_4\}) = 0$. Policy p_s is *satisfied* by a set of credentials $\mathcal{C}' \subseteq \mathcal{C}$ iff $g_{\phi_s}(\mathcal{C}') = 1$. During trust negotiation, a negotiator can disclose credential s if $g_{\phi_s}(\mathcal{C}') = 1$ where $\mathcal{C}'$ is the set of credentials that the negotiator has received from the opposing negotiator. In the rest of paper, for notational simplicity, we will replace x_i with c_i in policies, following the notation used in [16, 22, 23, 25]. The ϕ_s in the example above is thus represented as $\phi_s = (c_1 \wedge c_3) \vee c_2$.

A trust-negotiation protocol is normally initiated by a negotiator (typically, a client) requesting particular services from another negotiator (a server). Trust is established if the initially requested services are granted and all policies for disclosed credentials are satisfied [22]. In this case, the credential-exchange sequence is a *successful* negotiation. Otherwise, it is a *failed* negotiation.

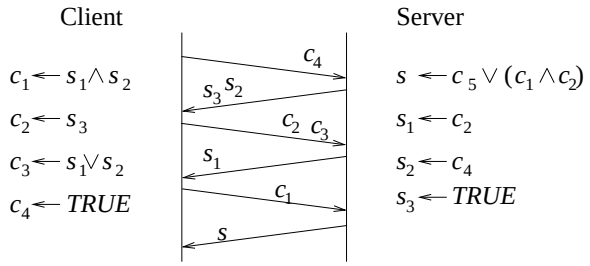


Figure 1: An example of exchange sequence of credentials.

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_4, since no previously-received server credentials are needed in order for the client to disclose c_4. The server then discloses s_3 (which has no precondition) and s_2 (which requires the earlier receipt of client credential c_4). The credential-exchange process continues as shown in the center of the figure. Note that at each round, all policies for disclosed credentials are satisfied.

The sequence of exchanged credentials depends on the decisions of each negotiator, referred to as a *strategy*. A strategy is based on local credentials, local policies, requests for local credentials from the opposing negotiator, and credentials received from the opposing negotiator. A strategy controls which credentials are

[1]Usually, monotonicity of policy languages [15, 25] is also assumed such that no negative operators ($\neg$) appear in policies.

disclosed and when, and when to terminate a negotiation [25].

Two negotiation strategies: an *eager* strategy and a *parsimonious strategy* are proposed in [19]. With eager strategies, two negotiators take turns disclosing a credential to the other side as soon as access control policy restrictions for that credential are satisfied. For example, the negotiation process in Figure 1 is achieved using an eager strategy. Conversely, with a parsimonious strategy, neither negotiator will disclose a credential until both of them know there exists a successful negotiation via an initial exchange of policies only. As a consequence, only credentials are exchanged using eager strategies, while under parsimonious strategies, both policies and credentials may be exchanged. Figure 2 shows the corresponding negotiation process of Figure 1 with a parsimonious strategy.

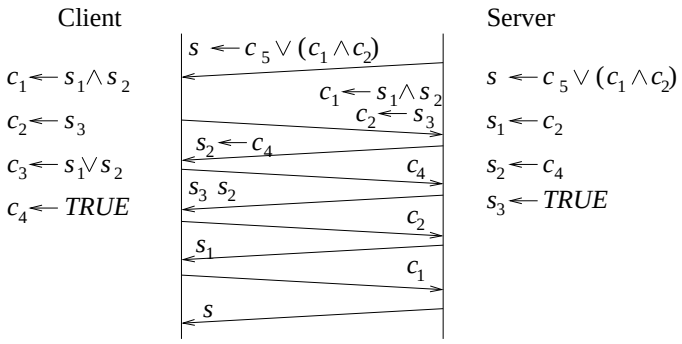


Figure 2: An exchange sequence of credentials for the negotiation in Figure 1 using a parsimonious strategy.

A strategy is *safe* if, under the strategy, all policies for credential disclosure are satisfied whenever a credential is disclosed [14]. Consider $p_s : s \leftarrow c_5 \vee (c_1 \wedge c_2)$ in Figure 1. $\phi_s = c_5 \vee (c_1 \wedge c_2)$. When either c_5, or both c_1 and c_2, are received, ϕ_s is satisfied and s can be disclosed safely. When a credential c can be disclosed without the receipt of any credentials from the opposing negotiator, we use the policy $p_c : c \leftarrow TRUE$. Instead, if a negotiator does not have a credential c, we have $p_c : c \leftarrow FALSE$, which is generally omitted.

When negotiators applying a particular strategy are able to find a successful credential-exchange sequence, whenever such a sequence exists, the strategy is a *complete* strategy [22].

There is a large body of work in the literature on trust negotiation [14, 19, 22, 25]. In this previous work, however, credentials are treated without preference (i.e., all credentials are assumed to be of equal value to a negotiator, in that a negotiator does not have a preference about whether to disclose credential c_i or c_j when given a choice) in the work. In this report, we are concerned with the preference of credentials.

3 Problem formulations

Given the problem setting in Section 2, we formulate the trust negotiation problem in this section.

Definition 1: *Given a set of credentials $\mathcal{C}_S$ and policies $\mathcal{P}_S$ processed by a negotiating server, and $\mathcal{C}_C$ and $\mathcal{P}_C$ by a client, the* general trust negotiation problem *initiated by a request for $s \in \mathcal{C}_S$ from the client, is to find an exchange sequence of credentials and policies $M_1 \circ M_2 \circ \cdots M_n$, such that*

(1) $s \in M_n$;

(2) $M_k \subseteq \mathcal{C}_S \bigcup \mathcal{P}_S$ or $M_k \subseteq \mathcal{C}_C \bigcup \mathcal{P}_C$, for $1 \leq k \leq n$; and

(3) $\forall m \in M_k, m \in \mathcal{C}_S \Rightarrow g_{\phi_m}(\mathcal{C}_C \bigcap(\bigcup_{j<k} M_j)) = 1$ and $m \in \mathcal{C}_C \Rightarrow g_{\phi_m}(\mathcal{C}_S \bigcap(\bigcup_{j<k} M_j)) = 1$, for all $1 \leq k \leq n$.

Condition (1) expresses the requirement that the sequence should achieve a successful negotiation, i.e., that the initially requested service s is granted. Condition (2) indicates that, at each exchange round, both parties exchange credentials or requests for credentials. These requests for credentials are in the form of policies that have credentials in the conditions of the policies. Condition (3) requires that every disclosure of a credential is safe, i.e., that the corresponding policy is satisfied when the credential is disclosed. In the above, n is called the number of exchange rounds.

Definition 1 is based on existing work [14, 16, 22], although no explicit formal definition is provided there. Similar to Definition 1, we have the following definition incorporating the sensitivity costs of credentials and policies.

Definition 2: *Given a set of credentials $\mathcal{C}_S$ and policies $\mathcal{P}_S$ processed by a negotiating server, $\mathcal{C}_C$ and $\mathcal{P}_C$ by a client, and a sensitivity cost w_c for any credential or policy $c \in \mathcal{C}_S \bigcup \mathcal{P}_S \bigcup \mathcal{C}_C \bigcup \mathcal{P}_C$, the* Minimum Sensitivity Cost (MSC) problem *initiated by a request for $s \in \mathcal{C}_S$ from the client, is to find an exchange sequence of credentials and policies $M_1 \circ M_2 \circ \cdots M_n$, such that*

(1) $s \in M_n$;

(2) $M_k \subseteq \mathcal{C}_S \bigcup \mathcal{P}_S$ or $M_k \subseteq \mathcal{C}_C \bigcup \mathcal{P}_C$, for $1 \leq k \leq n$;

(3) $\forall m \in M_k, m \in \mathcal{C}_S \Rightarrow g_{\phi_m}(\mathcal{C}_C \bigcap(\bigcup_{j<k} M_j)) = 1$ and $m \in \mathcal{C}_C \Rightarrow g_{\phi_m}(\mathcal{C}_S \bigcap(\bigcup_{j<k} M_j)) = 1$, for all $1 \leq k \leq n$; and

(4) $\Sigma_{c \in (\bigcup_{1 \leq k \leq n} M_k)} w_c$ is minimum.

For now, we assume that only credentials are protected by policies. Note, however, the formulations can

be extended to the case that policy disclosures themselves are protected by other policies [14].

In the remainder of this report, we assume that there is no cost to disclose the *names* or *ID*s of credentials to the opposing negotiator. In other words, only the disclosure of the *contents* of credentials incurs sensitivity costs. In most cases, possessing a credential is not sensitive, e.g., everyone is known to have an SSN although the SSN number is sensitive to disclose. By doing this, we exclude possession-sensitive credentials discussed in [16, 18, 23], which we leave as a direction for future work.

4 Solving the MSC problem

We will investigate the complexity of solving the MSC problem defined in Section 3 under two scenarios. In the first scenario, policies have no sensitivity costs and can be freely disclosed. This section is devoted to this first case. In the second scenario, which is discussed in Section 5, policies themselves are sensitive and disclosing a policy incurs a positive cost.

4.1 Policy-graph-based Strategy

When policies have no sensitivity costs, we propose a straightforward strategy, namely the *policy-graph-based* strategy, to solve the MSC problem. The strategy consists of four steps:

(1) Both negotiating parties first disclose all policies and the costs of local credentials to the other side;

(2) A *policy graph* based on the exchanged policies is constructed;

(3) The negotiators then apply algorithms to find, if it exists, a solution with minimum sensitivity cost;

(4) Both negotiators conduct the actual exchange sequence of credentials based on the resulting solution.

In this subsection, we briefly describe these four steps; the following subsection focuses on a complexity analysis of this strategy.

A policy graph consists of two kinds of nodes: circle nodes corresponding to credentials and rectangle nodes corresponding to "$\wedge$" operators in the policies. Consider a formula ϕ_s in policy $p_s : s \leftarrow \phi_s$. We assume that all ϕ_s are represented in a disjunctive normal form, a disjunction (sequence of ORs) consisting of one or more disjuncts, each of which is a conjunction (AND) of one or more credentials. If a disjunct consists of a single credential c, there is a direct edge from node c to node s. If a disjunct consists of several

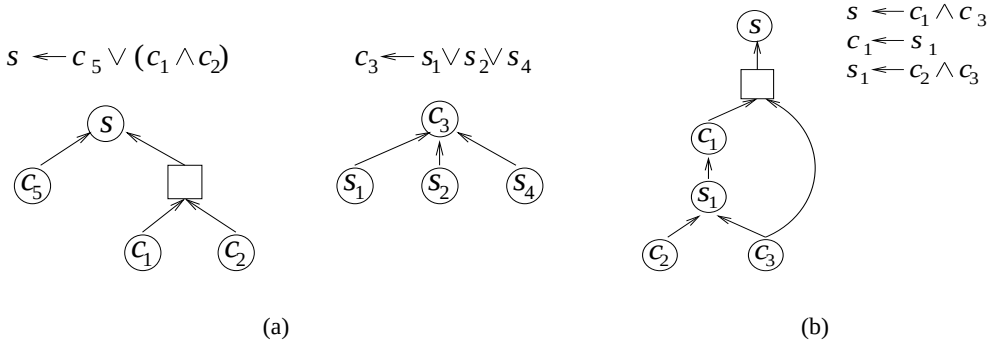


Figure 3: Examples of policies and corresponding policy graphs.

credentials, each credential has a directed edge to a rectangle node r, which further has a directed edge to c. Figure 3 shows three policy graphs and their corresponding policies.

```
//Assuming conditions of all policies are represented in a disjunctive normal form
Construct-Negotiation-Graph(P_S, P_C) {
(1)   E = {credentials in the condition of p_s}; // p_s is the policy for originally requested service s;
(2)   G = ∅;
(3)   BuildGraph(G, target of p_s, condition of p_s);
(4)   WHILE (E is not empty) {
(5)       Pick e from E;
(6)       IF (there is a policy p_e ∈ P_S ⋃ P_C s.t. e is the target of p_e) {
(7)           BuildGraph(G, target of p_e, condition of p_e);
(8)           E = (E ⋃ credentials in the condition of p_e) \ {e}; }
(9)       ELSE {
(10)          BuildGraph(G, e, FALSE);
(11)          E = E \ {e}; } }
(12)  RETURN G; }

BuildGraph(G, target, condition) {
(13)  IF (target ∉ G) Create a node for target;
(14)  IF (condition==FALSE) {
(15)      Prune(target, G);
(16)      return; }
(17)  FOR (each disjunct D of condition) {
(18)      IF (D is a conjunct consisting of more than one literals) {
(19)          Create a rectangle node R in G;
(20)          Link R to the node of target;
(21)          FOR (each element e of D) {
(22)              IF (e ∈ G) Link a direct edge from e to R;
(23)              ELSE Create a circle node for e and link a direct edge from e to R; } }
(24)      ELSE { //D is a single credential
(25)          IF (D ∈ G) Link a direct edge from D to the node of target;
(26)          ELSE Create a circle node for D and link a direct edge from D to the node of target;
(27)      } } }
```

Figure 4: Pseudo-code for constructing a policy graph.

Figure 4 presents the pseudo code of the algorithm constructing a policy graph, G, given the input of the server policies ($\mathcal{P}_S$) and the client policies ($\mathcal{P}_C$). Note that each credential has at most one corresponding circle node; however, there exists a unique rectangle node for each "$\wedge$" operator. Every credential that

can be disclosed without cost has an incoming edge from a node T that corresponds to "TRUE." When a credential appears multiple times in the policies, its corresponding node has multiple outgoing edges. Consequently, there may exist cycles in a policy graph, as shown in Figure 5(b), which presents the policy graph constructed in step (2) for the server and client policies shown in Figure 5(a).

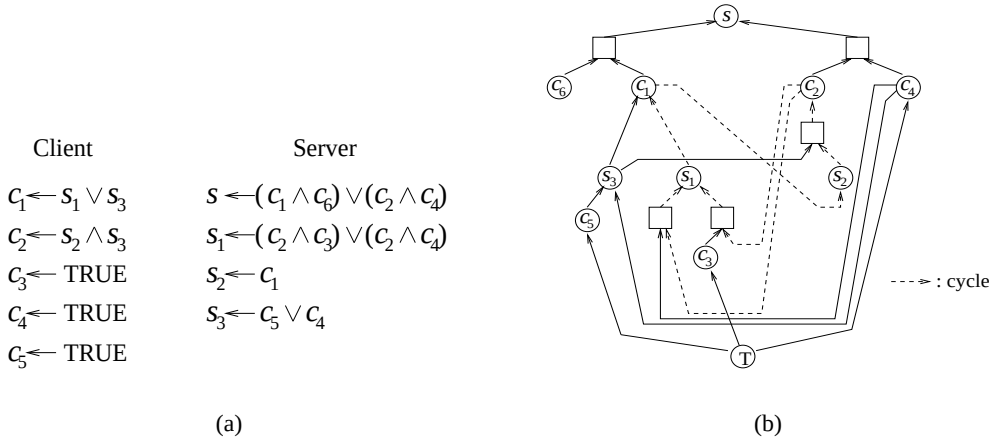


Figure 5: A policy graph (b) constructed based on the policies (a).

Once both negotiators have constructed a policy graph based on the exchanged policies, they conduct searching algorithms to find a successful solution with minimum cost in the graph. The MSC problem can be translated into the following *Minimum Directed-Acyclic-Graph* problem.

Definition 3: *Given a directed graph $G =< V, A >$, a node u' is* reachable *from a node u if there exists a sequence $\langle v_0, v_1, \ldots, v_k \rangle$ such that $v_0 = u$, $v_k = u'$ and $(v_{i-1}, v_i) \in A$ for $i = 1, 2, \ldots, k$.*

Definition 4: *Given a directed graph $G =< V, A >$, where the node set V consists of circle nodes U and rectangle nodes R, i.e., $V = U \cup R$, a* directed acyclic graph (DAG) *starting from a node u and ending at a node u' is a subgraph $G' \subseteq G$ such that*

(1) G' is acyclic and $u, u' \in G'$;

(2) There are no incoming edges to u and no outgoing edges from u' in G';

(3) For all $v \in G'$, v is reachable from u; and

(4) If a rectangle node v is in G', then $(v', v) \in A \Rightarrow v' \in G'$ for all $v' \in G$.

Note that condition (4) in Definition 4 ensures that, if a rectangle node, v, is in a DAG, all its *child nodes*, nodes that have outgoing edges to v, must also be in the DAG.

Definition 5: *Given a directed graph $G =< V, A >$, $V = U \cup R$, a source node u, a destination node u', and a cost map $w : V \rightarrow \mathbb{Z}^*$, the* Minimal DAG problem *is to find a DAG, denoted as $G_u^{u'}$, starting from u and ending at u' with minimum cost, i.e.,*

$$\min \sum_{v \in G_u^{u'}} w(v) \tag{1}$$

To solve the MSC problem, we need to find a minimum G_T^s in the constructed policy graph, where T is the node corresponding to "TRUE" and s represents the initially requested service. As we shall describe in the next subsection, the complexity of finding a minimum DAG depends on the policy graph. More specifically, if all nodes in the graph are circle nodes, it is polynomial solvable; however, the problem is **NP**-hard if the graph includes rectangle nodes.

Once a G_T^s is found in a policy graph, there exists a successful negotiation. Both negotiators exchange sequences of credentials according to this G_T^s, achieving a successful outcome. The exchange can be initiated by a negotiator who has freely-disclosed credentials, i.e., one of the parent nodes of node T in the policy graph. If both negotiators have such freely-disclosed credentials, either can initiate the exchange sequence. A credential c in G_T^s can be disclosed by a negotiator if the negotiator has received all credentials appearing as the predecessors of c in G_T^s from the opposing negotiator.

Proposition 1. *The policy-graph-based strategy is* safe *and* complete.

Proof: Condition (4) in Definition 4 guarantees that any node c in a G_T^s has at least one of its child nodes, which corresponds to a disjunct in ϕ_c, in the G_T^s. Credential c can be disclosed only when credentials corresponding to c's child nodes in the G_T^s are received, which means ϕ_c is satisfied. If a successful solution exists, there exists a G_T^s in the policy graph. Thus the strategy is complete. ■

4.1.1 Pruning the policy-graph

We observe that for a rectangle node in a policy graph to be in a G_T^s, all its child nodes must be in the G_T^s. If a negotiator does not have credential c, i.e., $p_c : c \leftarrow FALSE$, node c can not be in any G_T^s. Consequently we can prune node c and all its child edges. Furthermore, if such node c is a child node of a rectangle node r, r can also not appear in a G_T^s. Thus during the procedure of constructing a policy graph, when node c with policy $p_c : c \leftarrow FALSE$ is encountered (step (15) in Figure 4), a procedure **Prune**(c, G) is called to prune the graph, but the pruned graph still contains all valid G_T^s. Figure 6 show the pseudo-code of the

pruning procedure.

```
Prune(target, G) {
(1)    F = {target};
(2)    WHILE (F is not empty) {
(3)        Pick v from F;
(4)        FOR (each parent u of v) {
(5)            IF (u is a rectangle node)
(6)                F = F ⋃ {u};
(7)        }
(8)        F = F \ {v};
(9)        Delete v and all outgoing edges from v from G;
(10)   } }
```

Figure 6: Pseudo-code for pruning the policy graph during construction.

With this pruning, the pruned policy graph of the one in Figure 5(b) is shown in Figure 7.

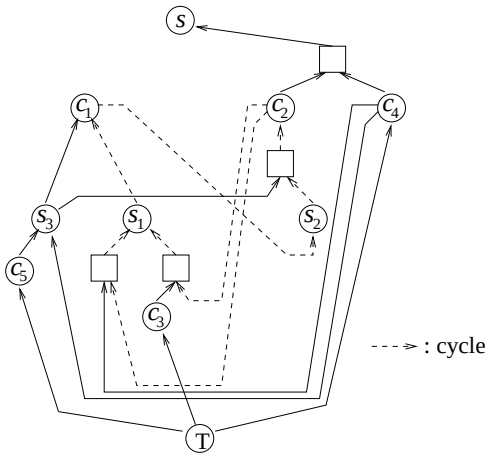


Figure 7: The pruned policy graph for policies shown in Figure 5(a)

4.2 Complexity Analysis

In this subsection, we analyze the complexity of the policy-graph-based strategy. In particular, we focus on the algorithm for constructing the policy graph, and algorithms for finding the minimum G_T^s in the resultant graph.

Proposition 2. *The running time of the algorithm for constructing a policy graph is polynomial in the number and the length of policies $\mathcal{P}_S$ and $\mathcal{P}_C$.*

Proof: We define the length, l_{p_s}, of a policy p_s, as the number of credentials in the condition of the policy. For example, policy $p_s \leftarrow c_5 \vee (c_1 \wedge c_2)$ has length of 3 since there are total 3 credentials in the

condition of p_s. Let L be the maximum length of all policies, i.e., $L = \max_{p_s \in \mathcal{P}_S \cup \mathcal{P}_C} l_{p_s}$.

Consider steps (4) to (11) in the **WHILE** loop. Since set E only includes credentials appearing in policies, $|E| \leq (L+1)(|\mathcal{P}_S| + |\mathcal{P}_C|)$. During each execution of the loop, step (8) or (11) deletes one element from E, thus the loop is at most executed $(L+1)(|\mathcal{P}_S| + |\mathcal{P}_C|)$ times.

Next consider procedure **BuildGraph**(). The **FOR** loop from step (17) to (25) has at most L executions since each execution deals with one disjunction in the *condition*. Assume that the policy-graph is stored in an $n \times n$ adjacency matrix, where n is the number of nodes in the output graph. Deciding whether a node has been in G (step (13), (21) and (24)) requires complexity of $O(n)$. Since at least two credentials are required to incur an $\wedge$ operator, a policy with length of l generates at most $\lfloor l/2 \rfloor$ rectangle nodes. Thus the resulting graph has at most $\lfloor L/2 \rfloor(|\mathcal{P}_S| + |\mathcal{P}_C|)$ rectangle nodes and $(L+1)(|\mathcal{P}_S| + |\mathcal{P}_C|)$ circle nodes (corresponding to credentials), i.e., $n \leq \lfloor L/2 \rfloor(|\mathcal{P}_S| + |\mathcal{P}_C|) + (L+1)(|\mathcal{P}_S| + |\mathcal{P}_C|)$. Consequently, the running time of procedure **BuildGraph**() is at most $L(\lfloor L/2 \rfloor + L + 1)(|\mathcal{P}_S| + |\mathcal{P}_C|)$.

As a result, the algorithm in Figure 4 will have a total running time that is at most $L(L+1)(\lfloor L/2 \rfloor + L + 1)(|\mathcal{P}_S| + |\mathcal{P}_C|)^2$. ■

Although a policy graph can be constructed in polynomial time, finding the minimum G_T^s in the graph is more complicated.

```
Compute-Cost(G, W) {
//W is the cost matrix for all credentials/services
//L[v] and W[v] store the LABEL and the cost of node v ∈ G respectively
//A is the set of nodes that are parent nodes of nodes in S
(1)   L[T] = ∅; c[T] = 0;
(2)   L[v] = ∅; c[v] = ∞; //for all other nodes except T nodes
(3)   S = {T};
(4)   For (each parent u of node T) {
(5)       Compute L[u] and W[u];
(6)       If (W[u] is finite) A = A ∪ {u}; }
(7)   WHILE (s ∉ S) or (A is not empty) {
(8)       Pick v ∈ A with minimum cost;
(9)       S = S ⋃ {v}; A = A \ {v};
(10)      For (each parent u of node v) {
(11)          Compute L[u] and W[u];
(12)          If (W[u] is finite) A = A ∪ {u}; }
(13)  }
(14)  If (W[s] is finite) return L[s] and W[s];
(15) }
```

Figure 8: Pseudo-code of variational Dijkstra's algorithm for finding a minimum G_T^s in a policy-graph G.

Proposition 3. *If a policy graph does not include rectangle nodes, the Minimum DAG problem can be solved using a variation of Dijkstra's algorithm, shown in Figure 8.*

Proof: When Dijkstra's algorithm in [6] is used, costs are associated with edges in the graph. When a

variation of Dijkstra's algorithm shown in Figure 8 is used, costs are associated with nodes rather than edges. But in both algorithms, costs are non-negative and the variation of Dijkstra's algorithm shown in Figure 8 operates exactly the same as Dijkstra's algorithm in [6]. The correctness of Dijkstra's algorithm in [6] guarantees that the variation of Dijkstra's algorithm returns a DAG $(L[s])$ with a minimum cost $(W[s])$. ■

When rectangle nodes exist in a policy graph, however, the MSC problem turns to be a **NP**-complete problem.

Proposition 4: *The general Minimum DAG problem is* **NP**-*complete*.

Proof: It is easy to show that the Minimum DAG problem ∈ **NP**. Given a G_T^s, validating condition (1) and computing the cost of G_T^s can be done in polynomial time. To show that it is **NP**-hard, we prove that 3-SAT is polynomially reducible to the Minimum DAG problem.

Given an instance of 3-SAT with clause set $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ and variable set $\mathcal{V} = \{x_1, x_2, \ldots, x_n\}$, we can formulate an instance of the Minimum DAG problem in Figure 9:

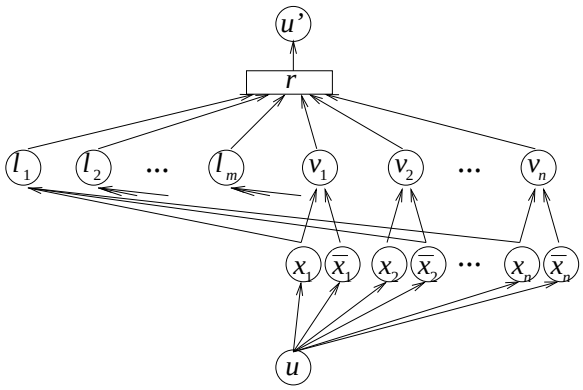


Figure 9: An instance of the Minimum DAG problem formulated from an instance of the 3-SAT problem

- Construct $2n$ circle nodes $\{x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots, x_n, \bar{x}_n\}$ for literals, n circle nodes $\{v_1, v_2, \ldots, v_n\}$ for variables, and m circles nodes for clauses. Construct a rectangle nodes r, a source node u and a destination node u'.

- Connect directed edges as shown in Figure 9. Note that a clause circle node, l_i, has three incoming edges from three literal nodes appearing in the clause.

- All literal nodes $\{x_i, \bar{x}_i\}$ have cost 1. Other nodes have cost 0.

The constructed Minimum DAG problem is to find a $G_u^{u'}$ with cost $w_{G_u^{u'}} \leq n$ subject to condition (1).

By solving this Minimum DAG problem in Figure 9, we can solve the given 3-SAT problem. If there is a $G_u^{u'}$ with cost $w_{G_u^{u'}} \leq n$, we answer "YES" for the given 3-SAT problem and the literal nodes in the $G_u^{u'}$ are assigned TRUE. Otherwise we answer "NO" for the 3-SAT problem.

Now, we want to show that there is a $G_u^{u'}$ with cost $w_{G_u^{u'}} \leq n$ in Figure 9 if and only if there is a truth assignment satisfying the 3-SAT instance.

"$\Rightarrow$". If a $G_u^{u'}$ with cost n exists, since r is a rectangle node, all clause and variable nodes must be in the $G_u^{u'}$. Consequently, one and exact one node from each literal-node pair $\{x_i, \bar{x}_i\}$ must be in the $G_u^{u'}$. We then assign those literal nodes in the $G_u^{u'}$ with TRUE. Since every clause node is in the $G_u^{u'}$, at least one of the three literal nodes appearing in a clause is also in the $G_u^{u'}$, which means the assignment above satisfies all clauses of the 3-CNF sentence.

"$\Leftarrow$". If there exists a truth assignment satisfying the 3-CNF sentence, we can put all nodes in Figure 9 in a $G_u^{u'}$ except those literal nodes that are NOT assigned TRUE. Such a $G_u^{u'}$ is a valid DAG starting from u and ending at u' since each variable node has exact one child node in the $G_u^{u'}$ and each clause node has at least one component literal node in the $G_u^{u'}$. The $G_u^{u'}$ also has a cost equal to n. ■

4.3 Heuristic Algorithms

Given the **NP**-completeness of the Minimum Sensitivity Cost problem even when policies can be disclosed freely, in this subsection, we describe two heuristic algorithms, followed by a performance study for these heuristics via simulation.

As we described in Section 4.2, a variation of Dijkstra's algorithm in Figure 8 can be used to solve the Minimum DAG problem when there are no rectangle nodes in the policy graph. So we consider that algorithm as the first heuristic, referred to as *Dijkstra's heuristic*, for the general Minimum DAG problem. Notice that, applying Dijkstra's heuristic, the cumulative cost of a rectangle node u is the sensitivity cost of u plus the cumulative costs of all the child nodes of u, i.e., if u is a rectangle node, step (11) in Figure 8 is replaced by $cost = W[u] + \sum_{v'} C[v']$, where v' is a child node of u.

To evaluate the performance of Dijkstra's heuristic, we randomly generate a set of client policies, server policies and credential costs with the following assumptions:

- The server has N_S credentials and a service s that is initially requested by the client; the client has N_C credentials.

- Formulas of all policies are in a disjunctive normal form; a formula ϕ_c for a protected credential c has k $(0 \leq k \leq K)$ disjuncts and each disjunct consists of m $(0 \leq m \leq M)$ credentials from the opposing negotiator. When $k = 0$, c can be disclosed freely, i.e., $c \leftarrow TRUE$.
- For the server, each of N_S credentials has the same probability of appearing in a disjunct of a policy formula of the client. Similar assumptions are made for the client credentials.
- A credential c has an integer cost w_c $(0 \leq w_c \leq W)$ if c is not freely-disclosed.
- k, m and w_c are all uniformly distributed in $[0, K]$, $[0, M]$ and $[0, W]$, respectively.

100 sets of policies are randomly generated and for each set, we create 100 different cost assignments for credentials. Among these 10^4 experiments, 8600 experiments have successful solutions[2]. We define the error percentage, $(C_{Approx}/C_{Optimal}) - 1$, to denote the performance of the heuristic algorithms, where C_{Approx} is the cost of the solution returned by Dijkstra's heuristic and $C_{Optimal}$ is the optimal solution[3].

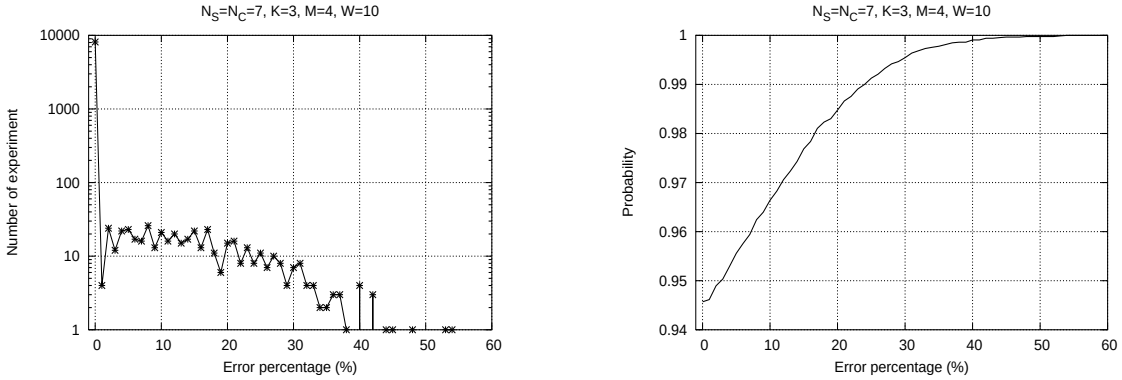
Figure 10 shows the performance of Dijkstra's heuristic with input parameters $N_S = N_C = 7, K = 3, M = 4$ and $W = 10$. Dijkstra's heuristic performs quite well in that it finds the solution with minimum cost in 8133 of the 8600 experiments. Figure 10(a) shows the number of experiments that had a given error percentage. Roughly, as the error percentage increases, the corresponding number of experiments decreases. Figure 10(b) shows the corresponding cumulative distribution of the number of experiments with a given error percentage, from which, one can see that all solutions returned by Dijkstra's heuristic have an error percentage that is less than 54%. We also simulated the eager strategy for the same 8600 experiments. The average cost achieved by the eager strategies, denoted as $\bar{C}_{Eager}$, is more than two times the cost returned by Dijkstra's heuristic, denoted as $\bar{C}_{Approx}$ (e.g.$\bar{C}_{Eager} = 38.7$ and $\bar{C}_{Approx} = 17.6$). Note that in the 1400 experiments that have no successful solutions, Dijkstra's heuristic does not disclose any credential but eager strategies will disclose a set of credentials till the negotiation is found to be failed.

To further improve the performance of the first heuristic, consider the policy graph in Figure 11(a) with the costs shown in the center. Dijkstra's heuristic chooses node c_2 rather than c_3 in the resultant G_T^s since $w_{c_2} < w_{c_3}$. However, c_3 must be in the G_T^s s because of the rectangle node. Consequently choosing c_3 as the predecessor of s_1 in the G_T^s has less cost than choosing c_2.

To remedy this situation, a node c with n outgoing edges to n rectangle nodes has a remedied cost $w'_c = w_c/(n+1)$. This remedied cost w'_c, is then used when searching for a G_T^s using Dijkstra's heuristic.

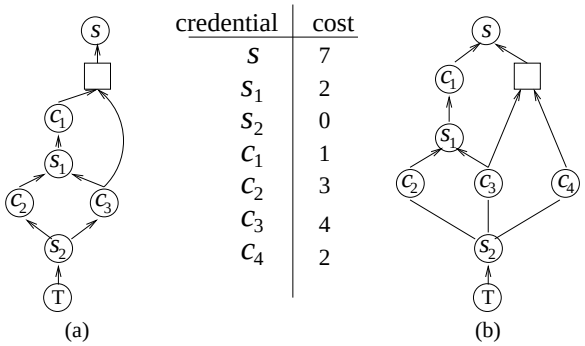
[2]The existence of successful solutions is determined by the polices, disregarding cost assignments. This means that roughly 86 out of 100 sets of policies were found to have successful solutions.

[3]The optimal solution is achieved by enumerating all possible DAGs, which requires exponentially computational complexity.



(a) (b)

Figure 10: Simulated performance of Dijkstra's heuristic

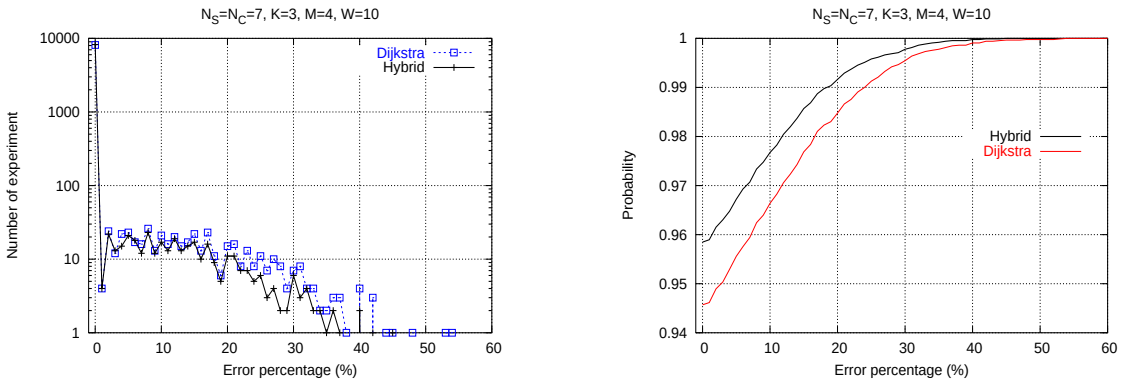


(a) (b)

Figure 11: Two policy-graphs.

The cost of the resultant G_T^s is still calculated using w_c. However, cost remediation may result a worse G_T^s than the one without remediation. For example, consider the policy graph in Figure 11(b). Here, the algorithm chooses c_3 in the resultant G_T^s with cost remediation even though the optimal G_T^s includes c_2 instead of c_3. Thus the second heuristic algorithm, referred to as the *hybrid* Dijkstra's heuristic, is to run the algorithm in Figure 8 twice: once with cost remediation and once without, and return the G_T^s with the smaller cost.

Simulation results in Figure 12 show that the hybrid Dijkstra's heuristic provides some improvement over the first heuristic algorithm. For example, it finds the solution with minimum cost in 8243 of the 8600 experiments with input parameters: $N_S = N_C = 7, K = 3, M = 4$ and $W = 10$.



(a) (b)

Figure 12: Compared performance of two heuristic algorithms

5 The MSC problem with policy-disclosure costs

In Section 4 we considered the solution of the MSC problem when policies are free to disclose. This section discusses the case when policies are themselves sensitive, i.e., there is a cost in policy disclosure.

As described earlier, with eager strategies, both negotiators immediately disclose a credential once that credential's conditions are satisfied. As a result, no policies are exchanged during the negotiation process. On the other hand, using policy-graph-based strategies, both negotiators exchange all policies and find a successful G_T^s with minimum sensitivity cost and thus only disclose a smaller set of credentials. It is possible that the minimum cost achievable using policy-graph-based strategies (including the cost of exchange policies) is higher than the cost under eager strategies. However, without knowledge of all policies, it is impossible to find a successful solution with minimum cost. For this reason, we propose a *greedy* strategy to approximately solve the MSC problem when policies are themselves sensitive.

The greedy strategy is based on eager strategies and consists of two steps. In the first step, both negotiators exchange the *names* and corresponding *accumulative* costs of credentials using eager strategies, i.e., a negotiator will disclose the name of a credential if the credential is satisfied by the names of credentials received from the opposing negotiator[4]. At the end of the first step, the negotiation is determined to have a successful solution or not. If a successful solution exists, the negotiation process evolves to the second step in which both negotiators construct the solution and exchange the *contents* of the credentials. These two steps are detailed in the following.

[4]We assume that both negotiators have consistent names of credentials.

5.1 Step One

In the first step, the negotiation process begins by one negotiator disclosing the names and cumulative costs of free credentials. The cumulative cost for a freely-disclosed credential is zero. After this initial step, both negotiators use eager strategies. More specifically, when a credential (except the initially request service s, for reasons that will be described shortly), is satisfied by the names of the credentials received from the opposing negotiator, its name and cumulative cost are disclosed. The cumulative cost of a satisfied credential c is the sum of the cost of the credential, w_c, and the cumulative costs of all credentials appearing in a satisfied disjunct of ϕ_c. If multiple disjuncts are satisfied, the one currently with minimal cumulative cost is chosen. The exchange process of names and cumulative costs continues till both negotiators have no more credentials to disclose. If the initially requested service s is not satisfied, it is a failed negotiation; otherwise, it is a successful negotiation. The reason for not disclosing the name of s immediately when s is satisfied is to find other possible solutions satisfying s with a smaller cost.

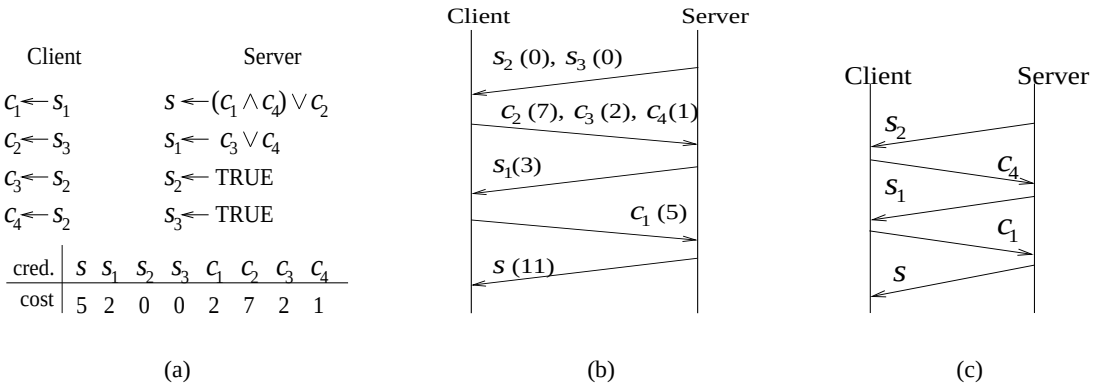


Figure 13: An exchange sequence using greedy strategies.

Figure 13 shows a simple two-step exchange using greedy strategies. The policies and cost are shown in Figure 13(a). Figure 13(b) shows the information exchanged during step one of the greedy strategies. The actual exchange of credentials conducted in step two is shown in Figure 13(c).

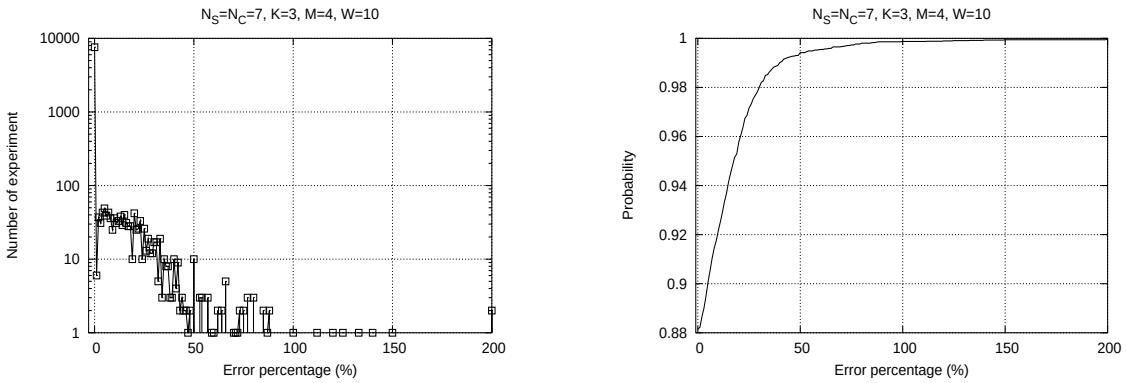
In Figure 13(b), the server begins the negotiation by disclosing $\mathcal{N}_1 = \{s_2(0), s_3(0)\}$, where the names of the credentials, s_2 and s_3, are followed by their cumulative costs, which are both 0. After the client receives $\mathcal{N}_1$, it discloses in $\mathcal{N}_2$ the names and cumulative costs of all credentials satisfied by $\mathcal{N}_1$, i.e., $\mathcal{N}_2 = \{c_2(7), c_3(2), c_4(1)\}$. Then the server discloses $\mathcal{N}_3 = \{s_1(3)\}$. Consider $p_{s_1} : s_1 \leftarrow c_3 \vee c_4$. The cumulative cost of s_1 is $w_{s_1} + w_{c_3}$ if c_3 is satisfied, or $w_{s_1} + w_{c_4}$ if c_4 is satisfied. If p_{s_1} is satisfied by both c_3 and c_4, the one with a smaller cumulative cost is chosen. So when the server receives $\mathcal{N}_2$, s_1 is satisfied by either c_3

or c_4, and c_4 is chosen due to its smaller cumulative cost. Also note that s is satisfied by c_2 in $\mathcal{N}_2$, but the server did not disclose the name of s immediately in $\mathcal{N}_3$. Indeed, another solution $(c_1 \wedge c_4)$ was found in a later disclosure with a smaller cost. When both negotiators have no more names of credentials to disclose and s is satisfied, the negotiation has a successful solution and the negotiation process enters the second step.

5.2 Step Two

In the second step of the greedy strategy, both negotiators construct a successful exchange sequence and conduct the actual exchange of credentials. To construct the sequence, the server sends the client the names of credentials that are chosen to satisfy p_s, which are referred to as a *counter-request*, denoted as $\mathcal{Q}_2$, e.g., $\mathcal{Q}_2 = \{c_1, c_4\}$ in the example shown in Figure 13. When a negotiator receives $\mathcal{Q}_i$, the negotiator replies with $\mathcal{Q}_{i+1}$ that includes the names of credentials chosen to satisfy the credentials in $\mathcal{Q}_i$. The process stops when $\mathcal{Q}_n$ consists of only freely-disclosed credentials (recall that n is the number of exchange rounds). $\mathcal{Q}_n, \ldots, \mathcal{Q}_2, \mathcal{Q}_1 = \{s\}$ consequently form a successful exchange sequence. In the example shown in Figure 13, $\mathcal{Q}_5 = \{s_2\}, \mathcal{Q}_4 = \{c_4\}, \mathcal{Q}_3 = \{s_1, s_2\}, \mathcal{Q}_2 = \{c_1, c_4\}$ and $\mathcal{Q}_1 = \{s\}$. The consequent exchange of credentials is conducted according to this sequence except that all credentials are disclosed at most once, i.e., if the name of a credential appears in $\mathcal{Q}_i$ and $\mathcal{Q}_j$ $(i > j)$, and the credential is disclosed in $\mathcal{Q}_i$, the credential will not be disclosed again in $\mathcal{Q}_j$. For example, as shown in Figure 13(c), s_2 appears in both $\mathcal{Q}_5$ and $\mathcal{Q}_3$. s_2 is disclosed at the first round (when $\mathcal{Q}_5$ is disclosed) and is not disclosed again at the third round (when $\mathcal{Q}_3$ is disclosed). Note that symbols for credentials in Figure 13(b) represent the names of the credentials whereas symbols in Figure 13(c) represent the contents of the credentials.

Clearly, it is not guaranteed that the solution achieved under greedy strategies be the optimal solution. Our simulation results in Figure 14 show that the greedy strategy finds the solution with minimum cost in 7611 of the 8600 experiments with input parameters: $N_S = N_C = 7, K = 3, M = 4$ and $W = 10$. The greedy strategy generates a higher error percentage (more than 200%) compared to the performance of the two heuristics described in Section 4. This is because the negotiators have no knowledge about the policies. It should be noted that, since no information of the G_T^s is included in $\mathcal{N}$ in step one, a credential may incur multiple copies of sensitivity cost in the cumulative costs if the credential appears multiple times in a solution. For example, c_4 incurs cost for s_1 and incurs cost again for s in Figure 13. However, only one copy of the cost is calculated in computing the cost of a solution in the simulation. The average cost achieved by the eager strategies ($\bar{C}_{Eager} = 38.7$) is around two times the average cost returned by the greedy strategy



(a) (b)

Figure 14: Simulated performance of greedy strategies

($\bar{C}_{Greedy} = 20.18$).

Policy Inference Under greedy strategies, although no policies are explicitly disclosed, partial information about the policies can be inferred based on the behaviors of the negotiators. For instance, in the example shown in Figure 13, the client may infer that $c_1 \wedge c_4$ is one disjunct of ϕ_s based on $\mathcal{Q}_2$. This form of policy inference can be concealed if the server adds other *mask* credentials into $\mathcal{Q}_2$. However, as pointed out in [23], the question of how to prevent policy inference deserves more consideration in the future work.

We conclude this section with the following proposition.

Proposition 5: *The greedy strategy is* safe *and* complete.

Proof: In step one of the greedy strategy, both negotiators exchange the names of credentials using the eager strategy. Because of the completeness of the eager strategy [19], a successful sequence is guaranteed to be found if it exists. Thus the greedy strategy is complete. A counter-request $\mathcal{Q}_{i+1}$ includes a set of credentials that satisfy credentials in $\mathcal{Q}_i$. In step two of the greedy strategy, since credentials in $\mathcal{Q}_{i+1}$ are disclosed earlier than the credentials in $\mathcal{Q}_i$, the strategy is safe. ■

6 Discussion

Thus far, we have focused on the MSC problem minimizing the total cost of credentials and policies disclosed by two negotiators during trust negotiation. In this section, we discuss related formulations of this problem.

6.1 The MSC Problem for One Negotiator

Consider an online-drug purchase during which a customer and an online store establish mutual trust followed by a transaction. A successful transaction is most important to the store and the store may not care about the negotiation cost. However, the customer may still prefer the solution that minimizes the customer's cost. In this case, the goal of the optimization problem is to minimize the cost of only one negotiator, which can be defined as the *Minimum Sensitivity Cost problem for One Negotiator (MSCON)*.

Definition 6: *Given a set of credentials $\mathcal{C}_S$ and policies $\mathcal{P}_S$ processed by a negotiating server, $\mathcal{C}_C$ and $\mathcal{P}_C$ by a client, and a sensitivity cost w_c for any credential or policy $c \in \mathcal{C}_S \bigcup \mathcal{P}_S \bigcup \mathcal{C}_C \bigcup \mathcal{P}_C$, the* Minimum Sensitivity Cost problem for One Negotiator (MSCON) *initiated by a request for $s \in \mathcal{C}_S$ from the client, is to find an exchange sequence of credentials and policies $M_1 \circ M_2 \circ \cdots M_n$, such that*

(1) $s \in M_n$;

(2) $M_k \subseteq \mathcal{C}_S \bigcup \mathcal{P}_S$ or $M_k \subseteq \mathcal{C}_C \bigcup \mathcal{P}_C$, for $1 \leq k \leq n$;

(3) $\forall m \in M_k, m \in \mathcal{C}_S \Rightarrow g_{\phi_m}(\mathcal{C}_C \bigcap(\bigcup_{j<k} M_j)) = 1$ and $m \in \mathcal{C}_C \Rightarrow g_{\phi_m}(\mathcal{C}_S \bigcap(\bigcup_{j<k} M_j)) = 1$, for all $1 \leq k \leq n$; and

(4) $\Sigma_{c \in (\cup_{1 \leq k \leq n} M_k) \bigcap (\mathcal{C}_C \cup \mathcal{P}_C)} w_c$ is minimum or $\Sigma_{c \in (\cup_{1 \leq k \leq n} M_k) \bigcap (\mathcal{C}_S \cup \mathcal{P}_S)} w_c$ is minimum.

In the proof of Proposition 4, we showed the **NP**-completeness of the MSC problem by reducing a general 3-SAT problem to a MSCON problem, which is a special case of the MSC problem. Thus the MSCON problem is also **NP**-hard even when policies are freely-disclosed. The policy-graph-based strategy and the greedy strategy proposed earlier can be used to approximately solve the MSCON problem when policies are free and with cost, respectively, by having the cost of all credentials and policies of the opposing negotiator to be 0.

6.2 The MSC Problem with Selfi sh Negotiators

The MSC problem minimizes the total cost of credentials and policies disclosed by both negotiators during trust negotiation. However, an optimal solution for the MSC problem may not give the minimal cost for a single negotiator. When negotiators are selfish, a negotiator's goal is to minimize his/her own cost disregarding the cost of the opposing negotiator. These considerations result in the following problem:

Definition 7: *Given a set of credentials $\mathcal{C}_S$ and policies $\mathcal{P}_S$ processed by a negotiating server, $\mathcal{C}_C$ and $\mathcal{P}_C$ by a client, and a sensitivity cost w_c for any credential or policy $c \in \mathcal{C}_S \bigcup \mathcal{P}_S \bigcup \mathcal{C}_C \bigcup \mathcal{P}_C$, the* Selfish-

Minimum Sensitivity Cost (SMSC) problem *initiated by a request for* $s \in \mathcal{C}_S$ *from the client, is to find an exchange sequence of credentials and policies* $M_1 \circ M_2 \circ \cdots M_n$*, such that*

(1) $s \in M_n$*;*

(2) $M_k \subseteq \mathcal{C}_S \bigcup \mathcal{P}_S$ *or* $M_k \subseteq \mathcal{C}_C \bigcup \mathcal{P}_C$*, for* $1 \leq k \leq n$*;*

(3) $\forall m \in M_k, m \in \mathcal{C}_S \Rightarrow g_{\phi_m}(\mathcal{C}_C \bigcap(\bigcup_{j<k} M_j)) = 1$ *and* $m \in \mathcal{C}_C \Rightarrow g_{\phi_m}(\mathcal{C}_S \bigcap(\bigcup_{j<k} M_j)) = 1$*, for all* $1 \leq k \leq n$*; and*

(4) $\Sigma_{c \in (\cup_{1 \leq k \leq n} M_k) \bigcap (\mathcal{C}_C \cup \mathcal{P}_C)} w_c$ *is minimum and* $\Sigma_{c \in (\cup_{1 \leq k \leq n} M_k) \bigcap (\mathcal{C}_S \cup \mathcal{P}_S)} w_c$ *is minimum.*

Since a successful sequence minimizing one negotiator's cost may not minimize the cost of the other negotiator, the SMSC problem may have no solution even though there exists a successful negotiation. Thus we revise the problem such that either negotiator safely discloses a set of credentials and policies that minimize the *remaining cost* at each step.

Definition 8: *The remaining cost,* C_M^R*, of a disclosure,* M*, of credentials and policies for a negotiator is defined as the minimum cost of credentials and policies that the negotiator needs to disclose after disclosing* M *to achieve a successful outcome if there exists. If there is no successful negotiation,* $C_M^R = \infty$ *for any* M*.*

Definition 9: *Given a set of credentials* $\mathcal{C}_S$ *and policies* $\mathcal{P}_S$ *processed by a negotiating server, and* $\mathcal{C}_C$ *and* $\mathcal{P}_C$ *by a client, and a sensitivity cost* w_c *for any credential or policy* $c \in \mathcal{C}_S \bigcup \mathcal{P}_S \bigcup \mathcal{C}_C \bigcup \mathcal{P}_C$*, the* Minimum Remaining Sensitivity Cost (MRSC) problem *initiated by a request for* $s \in \mathcal{C}_S$ *from the client, is to find an exchange sequence of credentials and policies* $M_1 \circ M_2 \circ \cdots M_n$*, such that*

(1) $s \in M_n$*;*

(2) $M_k \subseteq \mathcal{C}_S \bigcup \mathcal{P}_S$ *or* $M_k \subseteq \mathcal{C}_C \bigcup \mathcal{P}_C$*, for* $1 \leq k \leq n$*;*

(3) $\forall m \in M_k, m \in \mathcal{C}_S \Rightarrow g_{\phi_m}(\mathcal{C}_C \bigcap(\bigcup_{j<k} M_j)) = 1$ *and* $m \in \mathcal{C}_C \Rightarrow g_{\phi_m}(\mathcal{C}_S \bigcap(\bigcup_{j<k} M_j)) = 1$*, for all* $1 \leq k \leq n$*; and*

(4) $(\sum_{c \in M_k} w_c + C_{M_k}^R)$ *is minimum for each* $k = 1, \ldots, n$*.*

Before a negotiator discloses any credentials and policies, the remaining cost is exactly the optimal solution of the MSCON problem. Thus finding the remaining cost of a particular disclosure even when policies have no disclosure cost is **NP**-hard. Consequently, the MRSC problem is also **NP**-hard. We can first apply heuristic evaluation functions to approximate the remaining cost, followed by a *minimini* algorithm similar to the *minimax* algorithm in [13] to find a approximation solution for the MRSC problem.

6.3 Minimal Exchange Round Problem

When negotiators are concerned with the *speed* of trust negotiation rather than the cost, it is desirable to minimize the number of exchange rounds. This problem can be easily defined and the eager strategies proposed in [19] achieve the minimum number of exchange rounds. In the following section, a Finite State Machine is proposed to analyze the performance of a particular negotiation process in terms of the number of exchange rounds.

7 Modeling trust negotiation

In previous sections, we considered the computational complexity of solving cost optimization problems associated with credential disclosure in trust negotiation protocols. In this section, we turn our attention from credential disclosure costs to the time needed to successfully complete a negotiation protocol. Our goal will be to quantify the number of rounds needed to complete a negotiation under various credential disclosure policies.

7.1 Varied Strategies without Policy Knowledge

Let us begin by considering a trust negotiation process in which the server has a requested service s, and N_S credentials, among which, f_S credentials are freely-disclosed. The client has N_C credentials, f_C of which are freely-disclosed. We further assume that neither negotiator has knowledge of its opponent's policies, i.e., that policies are hidden [14] or sensitive [2].

Recall that under an *eager strategy*, both negotiators immediately disclose a credential once that credential's conditions are satisfied. Eager strategies thus achieve the minimum number of exchange rounds. The price paid for this minimum number of exchange rounds, however, is that a negotiator may disclose many more credentials than what is minimally required to successfully complete the negotiation. In contrast, under a *prudent strategy*, a negotiator discloses only one satisfied credential (specifically, the credential with minimum sensitivity cost) at each round. The goal of the prudent strategy is to minimize the number of credentials disclosed. Note that a prudent strategy differs from the parsimonious strategies [19] we considered earlier in that a prudent strategy has no knowledge of the opponent's policies.

Prudent strategies disclose as few credentials as possible. However, in the absence of policy knowledge, the credential disclosed by a negotiator may not satisfy any of the opponent's credential's policies. In

other words, the disclosure of a single credential under a prudent strategy may not be sufficient to allow a negotiation continue if the opposing negotiator has no satisfied credentials to disclose at the next round. As a compromise, at each round, a negotiator may disclose as few credentials as possible, but enough to advance the negotiation. A natural way to achieve this is to define a threshold $0 \leq \Gamma \leq 1$. A negotiator will calculate the corresponding probability, $p(x)$, that the opponent is able to continue the negotiation at the next round, given that the negotiator will disclose x credentials. Clearly, $p(x)$ increases as x increases. The negotiator consequently discloses a minimum number, m, of credentials such that $p(m) \geq \Gamma$. If a negotiator has A satisfied credentials and $p(A) < \Gamma$, the negotiator will disclose all A credentials. We will refer to this strategy as a *threshold* strategy with threshold Γ. Note that when $\Gamma = 0$, the threshold strategy is the same as the prudent strategy; when $\Gamma = 1$, the threshold strategy is the same as the eager strategy.

Prudent strategies and threshold strategies can reduce the number of credentials disclosed, but have a higher number of exchange rounds. Thus, there exists a trade-off between the number of exchange rounds and the number of credentials disclosed. To quantitatively study this tradeoff, we describe a simple Finite State Machine to model the rounds of credential exchange in the trust negotiation process.

7.2 A Non-deterministic Finite State Model

Rather than consider a specific set of client and server credential disclosure policies, we seek here a more general model in which the disclosure of credentials by one negotiator can lead to several different states for the other negotiator. Roughly speaking, this non-determinism models the fact that the server's policies are unknown to the client, and consequently (depending on the specific disclosure policies implemented by the server) a client's disclosure of a set of credentials can result in any number of server credentials becoming satisfied as a result of the client's disclosure. Thus, a non-deterministic Finite State Model similar to the probabilistic verification technique in [12] will be appropriate to model a trust-negotiation process in absence of policy knowledge.

The Finite State Machine is a 5-tuple $(\mathcal{S}, s_0, T, P, \mathcal{A})$ where $\mathcal{S}$ is a finite set of states; $s_0 \in \mathcal{S}$ is the initial state; $T : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ is a non-deterministic state transition function; $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ is a probability function; and $\mathcal{A} \subset \mathcal{S}$ is a set of accepting states.

A state $s_i \in \mathcal{S}$ consists of 5 elements: (r, d_S, a_S, d_C, a_C), where r is the number of exchange rounds to reach this state; d_S (d_C) is the number of credentials disclosed by the server (client) to the client (server); and a_S (a_C) is the number of server's (client's) credentials available, i.e., satisfied, but not disclosed to

the client (server). Note that $d_S, a_S \leq N_S$ and $d_C, a_C \leq N_C$. The initial state s_0 is $(1, 0, f_S, 0, f_C)$. A *successful* state is one in which the initially requested service is satisfied. A *failed* state is a state that is not a successful state and, in which both a_S and a_C are zero (meaning that none of the negotiators can continue the negotiation by disclosing new credentials.) The set of accepting states, $\mathcal{A}$, consists of all successful states. States that are neither successful states nor failed states are referred to as *transient* states.

When the FSM is in a transient state $s_i = (r, d_S, a_S, d_C, a_C)$, it may be the server's or the client's turn to continue the negotiation by disclosing satisfied (but previously undisclosed) credentials. If it is the server's turn, the FSM may transition into state $s_{i+1} = (r + 1, d_S + x, a_S - x, d_C, a_C + y)$, which indicates that the server discloses x $(1 \leq x \leq a_S)$ satisfied credentials and the client has y $(1 \leq y \leq N_C - a_C - d_C)$ credentials that are not satisfied at the rth round, but are satisfied at the $(r + 1)$th round. Similarly, if it is the client's turn, the FSM may transition into state $s_{i+1} = (r + 1, d_S, a_S + x, d_C + y, a_C - y)$, where $1 \leq y \leq a_C$ and $1 \leq x \leq N_S - a_S - d_S$.

Consider the transition: $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r + 1, d_S + x, a_S - x, d_C, a_C + y)$ with the disclosure of y credentials by the client to the server. The number, x, of the server's credentials that are satisfied by the client's disclosure is determined by the server's policies. Without knowledge of policies, the transition is non-deterministic, depending on the value of x. Let $P(s_i, s_{i+1})$ be the probability associated with the transition $s_i \rightarrow s_{i+1}$. If $P(s_i, s_{i+1})$ is 1, the transition is a deterministic transition, i.e., there is only one outgoing transition from state s_i. If $0 < P(s_i, s_{i+1}) < 1$, the transition is a non-deterministic transition. Correspondingly, s_i has multiple outgoing transitions. There are no transitions from a state to itself, i.e., $P(s_i, s_i) = 0$. Note that, for any $s_i \in \mathcal{S}$, $\sum_j P(s_i, s_j) = 1$.

Corresponding to the three strategies described in Section 7.1, we have the following transition functions T.

- With an eager strategy, at the $(r+1)$th round, a negotiator discloses all credentials that are satisfied at the rth round. Thus all transitions are: $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r+1, d_S+a_S, 0, d_C, a_C+y)$ when it is the server's turn, and $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r+1, d_S, a_S+x, d_C+a_C, 0)$ when it is the client's turn, where $0 \leq x \leq N_S - a_S - d_S$ and $0 \leq y \leq N_C - a_C - d_C$.

- With a prudent strategy, only one credential is disclosed at each round. Consequently, we have the following transition: $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r + 1, d_S + 1, a_S - 1, d_C, a_C + y)$ when it is the server's turn, or $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r + 1, d_S, a_S + x, d_C + 1, a_C - 1)$ when it is the client's turn, where $0 \leq x \leq N_S - d_S - a_S$ and $0 \leq y \leq N_C - d_C - a_C$.

- For threshold strategies, we need to determine the associated probability $P(s_i, s_{i+1})$ for the transition from s_i to s_{i+1}, which depends on the server's policies $\mathcal{P}_S$ and the client's $\mathcal{P}_C$. With specific assumptions on the set of all possible policies, we can determine $P(s_i, s_{i+1})$, as describe this the following subsection. Once we compute $P(s_i, s_{i+1})$, the transitions under threshold strategies are $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r+1, d_S + t_S, a_S - t_S, d_C, a_C + y)$ when it is the server's turn, or $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r+1, d_S, a_S + x, d_C + t_C, a_C - t_C)$ when it is the client's turn, where $1 \leq t_S \leq a_S$ ($1 \leq t_C \leq a_C$) is the number of credentials that the server (client) needs to disclose such that, with probability $P(s_i, s_{i+1}) \geq \Gamma$, the negotiation process can continue, i.e., $a_C + y > 0$ or $a_S + x > 0$, respectively.

7.3 State Transition Probabilities

To compute the transition probabilities $P(s_i, s_{i+1})$, we make the following assumptions for tractability.

(1) All the server's policies have the same disjunctive normal form. Each formula for the policy for satisfying a server credential has k_S disjuncts and each disjunct independently has m_S ($m_S \leq N_C$) distinct client credentials. Similar assumptions are made for client credentials, i.e., each formula for the policy for satisfying a client credential has k_C disjuncts and each disjunct independently has m_C ($m_C \leq N_S$) distinct server credentials.

(2) Each of the N_S server credentials independently has the same probability of appearing in a disjunct of a client formula. Similar assumptions are made for the client's credentials.

Throughout the following analysis, we only consider the server, i.e., we assume that the transition: $s_i \rightarrow s_{i+1}$ corresponds to the client's turn. The results for the client are similar.

Since all policies are in disjunctive normal form, a policy ϕ is satisfied if at least one disjunct is satisfied. Consider a disjunct consisting of m_S credentials chosen from a total of N_C credentials of the client. If the client has disclosed d_C credentials to the server, because of assumptions (1) and (2) above, the probability that the d_C credentials satisfy this disjunct is,

$$p_D = \begin{cases} \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}} & \text{if } d_C \geq m_S \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The probability that a policy is satisfied by the d_C credentials is thus

$$p_S = 1 - (1 - p_D)^{k_S} \tag{3}$$

Consider a transition from $s_i = (r, d_S, a_S, d_C, a_C)$ to $s_{i+1} = (r+1, d_S, a_S + x, d_C + y, a_C - y)$, i.e., the client has disclosed a total of d_C credentials to the server as of the $(r-1)$th round and will disclose y more credentials to the server at the $(r+1)$th round. The $d_C + y$ credentials may satisfy a server credential that is not satisfied by the d_C credentials at the $(r-1)$th round, with probability

$$\begin{aligned}
p_T &= Prob(\text{ } c \text{ is satisfied by } d_C + y \text{ credentials} \mid c \text{ is not satisfied by } d_C \text{ credentials}) \\
&= \frac{Prob(c \text{ is satisfied by } d_C + y \text{ credentials AND } c \text{ is not satisfied by } d_C \text{ credentials})}{Prob(c \text{ is not satisfied by } d_C \text{ credentials})} \\
&= \frac{Prob(c \text{ is satisfied by } d_C + y \text{ credentials}) - Prob(c \text{ is satisfied by } d_C \text{ credentials})}{Prob(c \text{ is not satisfied by } d_C \text{ credentials})} \\
&= \frac{[1 - (1 - \frac{\binom{d_C+y}{m_S}}{\binom{N_C}{m_S}})^{k_S}] - [1 - (1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}})^{k_S}]}{(1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}})^{k_S}} = 1 - [\frac{1 - \frac{\binom{d_C+y}{m_S}}{\binom{N_C}{m_S}}}{1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}}}]^{k_S}
\end{aligned} \tag{4}$$

Since all the server's policies have the same disjunctive normal form (assumption (1)), p_T is also the probability that the initially requested service is satisfied in s_{i+1}, i.e., s_{i+1} is an accepting state. Let p_A be the probability that s_{i+1} is an accepting state. We have $p_A = p_T$.

If s_{i+1} is not an accepting state, i.e., the initially requested service is not satisfied in s_{i+1}, let $p_{s_i,s_{i+1}}(x)$ be the probability that the $d_C + y$ credentials satisfy x server credentials that do not include the initially requested service. We have

$$\begin{aligned}
&p_{s_i,s_{i+1}}(x) = \\
&\quad (1 - p_A)\tbinom{N_S - d_S - a_S}{x}(p_T)^x(1 - p_T)^{N_S - d_S - a_S - x}
\end{aligned} \tag{5}$$

If s_{i+1} is not an accepting state and both $a_S + x = 0$ ($\Rightarrow a_S = x = 0$) and $a_C - y = 0$ ($\Rightarrow a_C = y$), s_{i+1} is a failed state. Let p_F be the probability that s_{i+1} is a failed state. We have

$$p_F = p_{s_i,s_{i+1}}(0) = (1 - p'_T)^{N_S - d_S + 1} \tag{6}$$

where $$p'_T = 1 - [\frac{1 - \frac{\binom{d_C + a_C}{m_S}}{\binom{N_C}{m_S}}}{1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}}}]^{k_S}$$

Note that the sum of all the outgoing transition probabilities from s_i is equal to 1, i.e.,

$$p_A + \sum_{x=0}^{N_S - d_S - a_S} p_{s_i, s_{i+1}}(x) = p_A + (1 - p_A) = 1 \tag{7}$$

Based on the p_T computed in (4), we are now able to compute the transition probabilities, $P(s_i, s_{i+1})$, for the three strategies described earlier in Section 7.1.

The eager strategy Under the eager strategy, state s_i is $(r, d_S, 0, d_C, a_C)$ and $y = a_C$. Consequently, we have

$$p_A^E = 1 - [\frac{1 - \frac{\binom{d_C + a_C}{m_S}}{\binom{N_C}{m_S}}}{1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}}}]^{k_S} \tag{8}$$

$$p^E_{s_i, s_{i+1}}(x) = (1 - p_A^E)\binom{N_S - d_S}{x}(p_A^E)^x(1 - p_A^E)^{N_S - d_S - x} \tag{9}$$

$$p_F^E = (1 - p_A^E)^{N_S - d_S + 1} \tag{10}$$

The prudent strategy Under the prudent strategy, y is equal to 1. Thus, we have

$$p_A^P = 1 - [\frac{1 - \frac{\binom{d_C + 1}{m_S}}{\binom{N_C}{m_S}}}{1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}}}]^{k_S} \tag{11}$$

$$p^P_{s_i, s_{i+1}}(x) = (1 - p_A^P)\binom{N_S - d_S - a_S}{x}(p_A^P)^x(1 - p_A^P)^{N_S - d_S - a_S - x} \tag{12}$$

$$p_F^P = (1 - p_A^P)^{N_S - d_S + 1} \tag{13}$$

Note that under the prudent strategy, if state s_i is $(r,\ d_S, a_S,\ d_C, 0)$ and $a_S > 0$, which means that the client has no satisfied credentials to disclose, s_i deterministically transitions to state $s_{i+1} : (r + 1, d_S, a_S, d_C, 0)$, i.e., $p^P_{s_i,s_{i+1}}(0) = 1$ (since $a_C = y = 0 \Rightarrow p^P_A = 0$). Transient state s_{i+1} transitions to state $s_3 : (r + 2, d_S + 1, a_S - 1, d_C, y)$ $(0 \leq y \leq N_C - d_C)$ with probability $p^P_{s_{i+1},s_{i+2}}(y)$, which means that, at the $(r + 1)$th round, the client discloses nothing to the server and at the $(r + 2)$th round, the server discloses one satisfied credential to the client.

The threshold strategy Given threshold Γ in transient state $s_i : (r, d_S, a_S, d_C, a_C)$, the client will disclose a minimum of t_C $(1 \leq t_C \leq a_C)$ credentials to the server such that $p^T_A(t_C) \geq \Gamma$ (i.e., s_{i+1} is an accepting state) or $1 - p^T_F(t_C) \geq \Gamma$ (i.e., the server has at least one credential satisfied with a probability greater than Γ), where

$$p^T_A(t_C) \quad = \quad 1 - [\frac{1 - \frac{\binom{d_C + t_C}{m_S}}{\binom{N_C}{m_S}}}{1 - \frac{\binom{d_C}{m_S}}{\binom{N_C}{m_S}}}]^{k_S} \tag{14}$$

$$p^T_F(t_C) \quad = \quad (1 - p^T_A)^{N_S - d_S + 1} \tag{15}$$

Recall that if $p^T_A(a_C) < \Gamma$ and $(1 - p^T_F(a_C)) < \Gamma$, the client will disclose all of the a_C credentials.

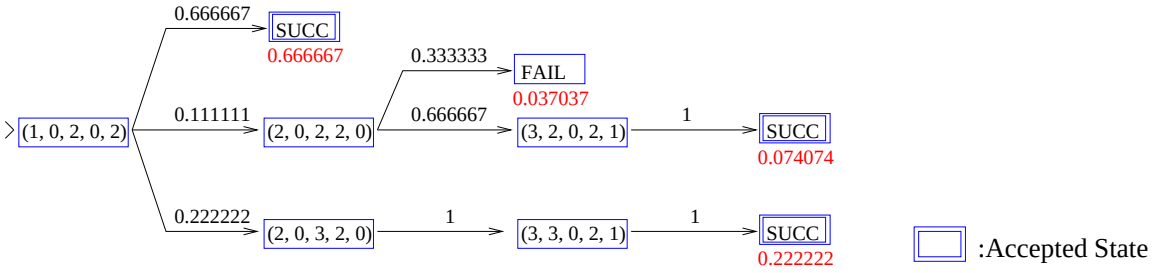


Figure 15: State exploration of the FSM using eager strategies.

Figure 15 shows the state exploration of the FSM using prudent strategies with parameters $N_S = N_C = 3$, $m_S = m_C = 1$, $k_S = k_C = 1$, and with initial state $(1, 0, 2, 0, 2)$, i.e., both the server and the client have two freely-disclosed credentials. The negotiation begins with the client's disclosure of credentials. In the figure, each transition is labelled with a transition probability. Beginning from the initial state, the FSM will eventually terminate in a failed (FAIL) state or a successful (SUCC) state. Each terminal state

has a probability (positioned below the state) that the FSM will reach the state, equal to the product of the transition probabilities along the path from the initial state to the terminal state. Note that the sum of the outgoing transition probabilities of a transient state is equal to 1. The sum of the probabilities of all terminal states is also equal to 1.

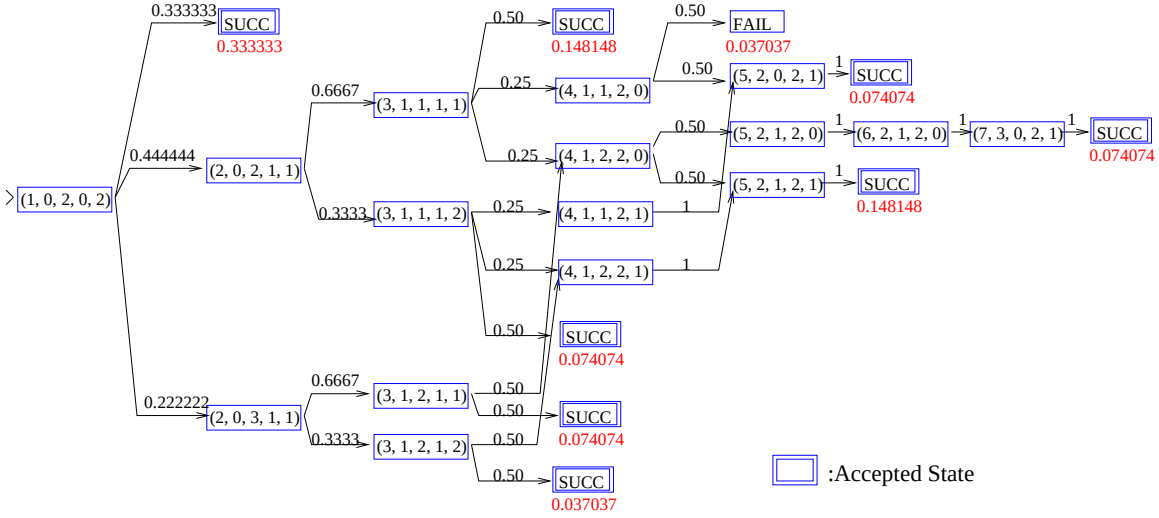


Figure 16: State exploration of the FSM using prudent strategies.

Figure 16 and Figure 17 show the state explorations of the FSM with the same parameters using prudent strategies and threshold strategies, respectively. Note that in Figure 16, in state $(5, 2, 1, 2, 0)$, the client has no available credentials to disclose. With probability 1, the FSM goes to state $(6, 2, 1, 2, 0)$, which means that the client informs the server that the client can not further the negotiation. Since the server still has one available undisclosed credential, the negotiation process continues with the server's disclosure of the credential; the FSM goes to state $(7, 3, 0, 2, 1)$.

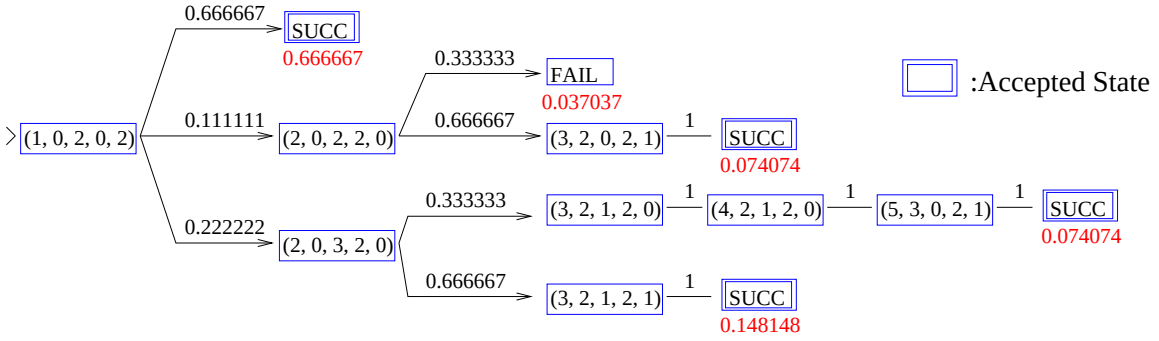
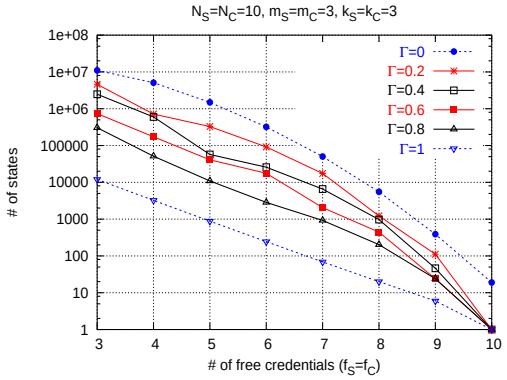


Figure 17: State exploration of the FSM using threshold strategies $(\Gamma = 0.5)$.

It is clear that the FSM explores more states under a prudent strategy than under the eager strategy

and the threshold strategy. The FSM generates the minimal number of states under the eager strategy. Figure 18 shows the number of states that the FSM explores under different strategies with the number of freely-disclosed credentials (f_S and f_C), given $N_S = N_C = 10, m_S = m_C = 3$ and $k_S = k_C = 3$. The number of the explored states decreases exponentially as f_S and f_C increase. Note that for a transition $s_i : (r, d_S, a_S, d_C, a_C) \rightarrow s_{i+1} : (r+1, d_S+x, a_S-x, d_C, a_C+y)$, we have $d_S+x+a_S-x+d_C+a_C+y \geq d_S + a_S + d_C + a_C$ since $y \geq 0$. Consider two initial states $s_0 = (1, 0, f_S, 0, f_C)$ and $s'_0 = (1, 0, f'_S, 0, f'_C)$ such that $f_S < f'_S$ and $f_C < f'_C$. With s'_0, the FSM does not explore states $s_i : (r, d_S, a_S, d_C, a_C)$ satisfying $d_S + a_S + d_C + a_C \leq f'_S + f'_C$, which are explored by the FSM with s_0. Thus, with more freely-disclosed credentials in the initial state, the FSM only explores proportional of the states that the FSM explores with less freely-disclosed credentials in the initial state.

We also observe that, using threshold strategies with $\Gamma = 0.5$, the FSM only explores around 1/10 of the states explored using eager strategies. Note that the prudent strategies are the threshold strategies with $\Gamma = 0$; the eager strategies are the threshold strategies with $\Gamma = 1$.



(a)

Figure 18: Number of states explored by the FSM using different strategies

7.4 Probability of a Successful Negotiation, and a Quantitative Comparison of Credential Disclosure Strategies

In this section, we use the transient FSM model to quantitatively investigate the probability that a negotiation ends in success (as a function of the number of freely-disclosed credentials) and then compare the expected number of rounds needed to reach a successful negotiation under the eager, prudent, and threshold strategies.

The probability that a negotiation is successful, i.e., the FSM terminates in an accepting state, is defined

as

$$p_{SUCC} = \sum_{s_i} Prob(\text{FSM accepts in state } s_i) \tag{16}$$

The expected number of exchange rounds when a negotiation terminates (either a successful negotiation or a failed negotiation) is defined as

$$E[R] = \sum_{r=1}^{\infty} r \times Prob(\text{FSM terminates at the } r\text{th round}) \tag{17}$$

Note that $E[R]$ is not conditioned on whether or not the negotiation is successful. We thus further define the expected number of exchange rounds given that the negotiation is successful, i.e.,

$$\begin{aligned} & E[R_{SUCC}] \\ & = \textstyle\sum_{r=1}^{\infty} r \times Prob(\text{FSM accepts at the } r\text{th round} \mid \text{FSM accepts}) \\ & = \frac{\sum_{r=1}^{\infty} r \times Prob(\text{FSM accepts at the } r\text{th round})}{p_{SUCC}} \end{aligned} \tag{18}$$

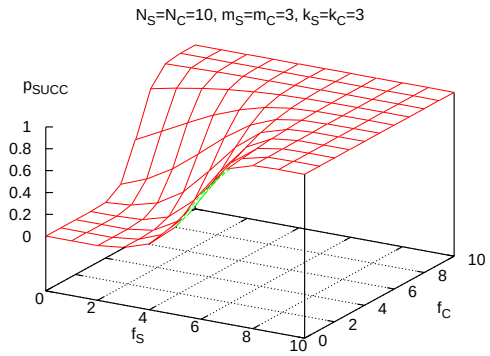
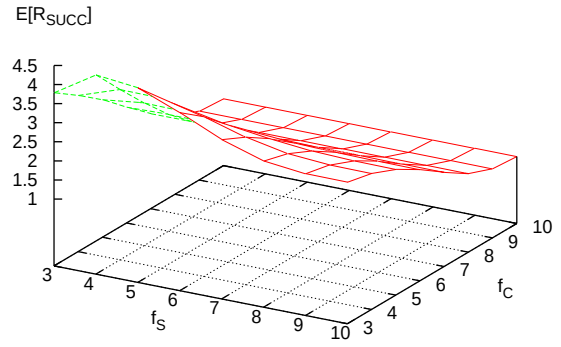


Figure 19: p_{SUCC} with the number of freely-disclosed credentials.

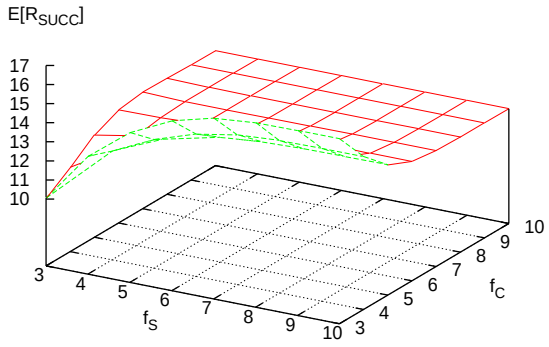
Figure 19 shows the probability, p_{SUCC}, that a negotiation is successful as a function of the number of freely-disclosed credentials (f_S and f_C), given $N_S = N_C = 10, m_S = m_C = 3$ and $k_S = k_C = 3$. For a particular configuration $(N_S, N_C, m_S, m_C, k_S, k_C)$, p_{SUCC} is determined once f_S and f_C are given, independent of the policy-disclosure strategy. Observe that when $f_S < m_C$ and $f_C < m_S$, p_{SUCC} is 0. We

also observe that, given that $f_C < m_S$, p_{SUCC} increases rapidly with f_S when $f_S > m_C$.

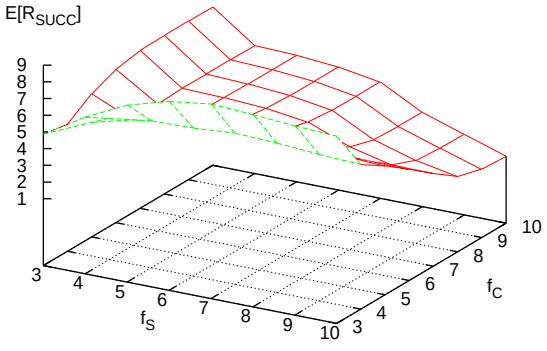
Ideally, p_{SUCC} indicates the probability that an initial state can lead to an accepted sate whereas $E[R_{SUCC}]$ represents how quickly the FSM reaches an accepted state from an initial state.



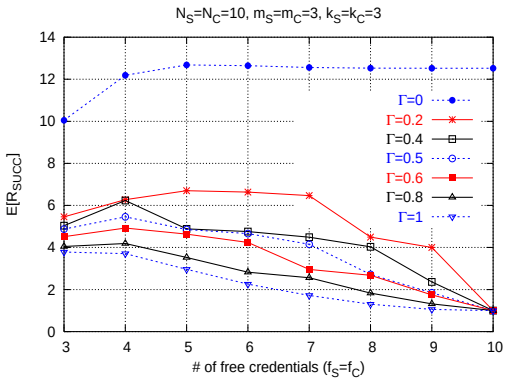
(a) Eager strategies



(b) Prudent strategies



(c) Threshold strategies ($\Gamma = 0.5$)



(d) Comparison of three strategies

Figure 20: Comparison of numerical results of three strategies

Figure 20(a), 20(b) and 20(c) show the expected number of successful exchanging rounds, $E[R_{SUCC}]$, with f_S and f_C using eager, prudent and threshold ($\Gamma = 0.5$) strategies respectively, given parameters: $N_S = N_C = 10, m_S = m_C = 3$ and $k_S = k_C = 3$. Figure 20(d) shows that from a particular initial state, the eager strategy has a smaller expected number of steps to reach a successful state $E[R_{SUCC}]$ than the threshold strategy, which in turn requires a smaller $E[R_{SUCC}]$ than a prudent strategy. Observe that,

with a small threshold $\Gamma = 0.2$, threshold strategies require only half of $E[R_{SUCC}]$ compared to prudent strategies.

8 Related work

There is a growing body of work on trust negotiation, as summarized in [21]. This work can be broadly classified into two groups, depending on whether the policies are considered sensitive or insensitive to disclosure. In [9, 11, 19, 22], policies are assumed to be insensitive and thus freely-disclosable. Work in [2, 14] considers protecting sensitive policies. Yu *et al.* [24] proposed a unified scheme to model both situations. This paper consequently addresses the MSC problem separately in the case that policies are freely-disclosed and in the case that policies themselves are sensitive.

Two negotiation strategies, namely the eager strategy and the parsimonious strategy, were proposed in [19]. Greedy strategies proposed in this paper are based on the eager strategy, but as we have seen, take sensitivity costs into account at each exchange round.

Our policy graph is similar to the AND/OR tree used by a brute-force backtracking strategy in [22]. As Yu *et al.* have noted, an AND/OR tree may have exponential size. Consequently, an efficient and complete strategy, referred to as *PRUNES*, is proposed in [22]. However, there are difference between a policy graph and an AND/OR tree. First, a policy graph is directed and may have cycles. Second, by making each credential appear at most once in the graph, and with a fine-grained improvement using the pruning technique, a policy graph is guaranteed to be of polynomial size. Last, an AND/OR tree is *virtual* to negotiators, i.e., the search of the AND/OR tree is actually conducted in a distributed manner by the negotiators at two sites and each negotiator only has partial information about the tree. Conversely, when policies are freely-disclosed, both negotiators will construct the same policy graph. We also note that the PRUNES strategy [22] finds only a single solution without regard to cost, whereas we show that finding a solution with minimum sensitivity cost is an **NP**-complete problem. Indeed, the fact that credentials have been treated without preference in previous work motivated our work in this paper. We assign a non-negative sensitivity cost to a credential and policy to reflect the need that, in practice, people want to disclose as little sensitive information as possible.

When policies are sensitive, partial information about policies can be inferred based on negotiators' behaviors. This form of information leakage is studied in [16, 17, 18, 23] and deserves further consideration in the future.

9 Conclusion and future work

In this report we have formulated and studied the Minimum Sensitivity Cost (MSC) problem in trust-negotiation protocols, where a non-negative sensitivity cost is assigned to the disclosure of each credential or policy. Even when policies can be freely-disclosed, the MSC problem is shown to be **NP**-complete. Fortunately, a variation of Dijkstra's algorithm can be used to solve the problem efficiently, achieving 95% of optimal in simulation for the cases studied. We proposed a safe and complete strategy known as the greedy strategy, to solve the MSC problem approximately when policies themselves are sensitive. Our simulation results showed that the greedy strategy achieves 88.6% of optimal for the cases studied. Finally, a simple FSM model of trust-negotiation protocols was developed and used to provide a quantitative evaluation of the number of exchange rounds needed to achieve a successful negotiation, and the probability of achieving a successful negotiation under various credential disclosure strategies.

This work can be extended in a number of different directions. We have assumed that credential *names* can be freely disclosed in the greedy strategy. However, possession-sensitive credentials may exist in practice [16, 18, 23]. Disclosure of possession-sensitive credential names thus has a cost. Modeling and understanding to consequences of this cost is an interesting avenue for future research. Our FSM model presently does not include the sensitivity cost in the state descriptions, and thus can not be used to compare the sensitivity cost of different trust-negotiation protocols. This, together with a relaxation of the assumptions made to compute the FSM transition probabilities are also potential areas for future research. Also of interest is to prevent policy inference that can be made based on the behaviors of the negotiators during trust negotiation.

References

[1] E. Bina, V. Jones, R. McCool, and M. Winslett. Secure access to data over the internet. In *Third ACM/IEEE International Conference on Parallel and Distributed Information Systems*, Austin, Texas, September 1994.

[2] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *ACM Conference on Computer and Communications Security (CCS)*, Athens, Greece, November 2000.

[3] S. Brainov and T. Sandholm. Contracting with uncertain level of trust. *Computational Intelligence, Special issue on Agent Technology for Electronic Commerce*, 18(4):501–514, 2002.

[4] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter base protocol. IETF, RFC 3588,
`http://www.faqs.org/rfcs/rfc3588.html`.

[5] K. Chopra and W. A. Wallace. Trust in electronic environments. In *36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, pages 331–340, Big Island, Hawaii, January, 2003.

[6] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*, chapter 24, pages 595–599. MIT Press, 1990.

[7] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys*, pages 2–16, Fouth Quarter 2000.

[8] Investorwords. http://www.investorwords.com/cgi-bin/getword.cgi?5084.

[9] W. Johnston, S. Mudumbai, and M. Thompson. Authorization and attribute certificates forwidely distributed access control. In *IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Stanford University, CA, June 1998.

[10] A. Kini and J. Choobinch. Trust in electronic commerce: Definitions and theoretical considerations. In *31st Annual Hawaii International Conference on System Sciences*, Hawaii, 1998.

[11] N. Li, W. Winsborough, and J. Mitchell. Distributed credential chain discovery in trust management. In *Eighth ACM Conference on Computer and Communications Security*, Philadelphia, PA, November 2001.

[12] N. Maxemchus and K. Sabnani. Probabilistic verification of communication protocols. In *VII-th Workshop on Protocol Specification, Testing, and Verification*, Zurich, Switzerland, 1987.

[13] S. Russell and P. Norvig. *Artificial-Intelligence: A Modern Approach*, chapter 5, pages 122–141. Prentice, 1995.

[14] K. Seamons, M. Winslett, and T. Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Symposium on Network and Distributed System Security*, San Diego, CA, February 2001.

[15] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Monterey, 2002.

[16] K. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting privacy during on-line trust negotiation. In *2nd Workshop on Privacy Enhancing Technologies (PET)*, San Francisco, April 2002.

[17] W. Winsborough and N. Li. Protecting sensitive attributes in automated trust negotiation. In *ACM Workshop on Privacy in the Electronic Society*, Washington, DC, November 2002.

[18] W. Winsborough and N. Li. Towards practical automated trust negotiation. In *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks (Policy 2002)*, Monterey, CA, June 2002.

[19] W. Winsborough, K. Seamons, and V. Jones. Negotiating disclosure of sensitive credentials. In *Second Conference on Security in Communication Networks*, Amalfi, Italy, September 1999.

[20] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Using digital credentials on the world-wide web. *Journal of Computer Security*, 5:255–267, 1997.

[21] M. Winslett, T. Yu, K. Seamons, A. Hess, J. Jarvis, B. Smith, and L. Yu. Negotiating trust on the web. *IEEE Internet Computing Sepcial Issue on Trust Management*, 6(6), November 2002.

[22] T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for trust negotiation over the internet. In *ACM Conference on Computer and Communications Security*, Athens, November 2000.

[23] T. Yu and M. Winslett. Policy migration for sensitive credentials in trust negotiation. In *Workshop on Privacy in the Electronic Society*, Washington, DC, October 2003.

[24] T. Yu and M. Winslett. A unified scheme for resource protection in automated trust negotiation. In *IEEE Symposium on Security and Privacy*, Berkeley, California, May 2003.

[25] T. Yu, M. Winslett, and K. Seamons. Interoperable strategies in automated trust negotiation. In *ACM Conference on Computer and Communications Security*, Philadelphia, Pennsylvania, November 2001.