IEICE TRANS. INF. & SYST., VOL.E92–D, NO.2 FEBRUARY 2009

PAPER *Special Section on Foundations of Computer Science*

Formal Language Theoretic Approach to the Disclosure Tree Strategy in Trust Negotiation

Yoshiaki TAKATA[†a)] ***and*** **Hiroyuki SEKI**[††], ***Members***

SUMMARY Trust negotiation is an authorizing technique based on digital credentials, in which both a user and a server gradually establish trust in each other by repeatedly exchanging their credentials. A trust negotiation strategy is a function that answers a set of credentials to disclose to the other party, depending on policies and the set of already disclosed credentials. The disclosure tree strategy (DTS), proposed by Yu et al., is one of the strategies that satisfies preferable properties. DTS in a simple implementation requires exponential time and space; however, neither an efficient algorithm nor the lower-bound of its complexity was known. In this paper, we investigate the computational complexity of DTS. We formulate subproblems of DTS as problems on derivation trees of a context-free grammar (CFG), and analyze the computational complexity of the subproblems using the concepts of CFGs. As a result, we show that two subproblems EVL and MSET of DTS are **NP**-complete and **NP**-hard, respectively, while both are solvable in polynomial time if we modify EVL not to require non-redundancy and MSET not to answer any subset useless for leading the negotiation to success.

key words: *trust management, trust negotiation, negotiation strategy, computational complexity, context-free grammar*

1. Introduction

Trust management [1]–[4], that is an authorizing technique based on digital credentials and policy specification, is a promising technology for modern network applications in which all potential users are not known in advance. In a trust management system, an administrator specifies a policy such as "If a user has a credential issued by company A, then provide service R to the user" beforehand, and the system decides whether or not a user should be provided with a service based on the policies and the credentials that the user presents. Digitally signed credentials guarantee the holders' attributes, and using them the system can provide an appropriate authorization to a user without a prior-registered user ID or password information.

Trust negotiation [5]–[9] is one of the approaches to trust management, in which both a user and a server gradually establish trust in each other by repeatedly exchanging their credentials. In contrast to a usual trust management system where a user should know and trust the server at the beginning of a transaction, trust negotiation supports trust establishment between strangers.

Figure 1 shows an example of trust negotiation. The negotiation in this example is performed by Bob, an employee of Company F, and Company A. Both parties have their own set of credentials, policies, and services. Assume that Bob starts the negotiation by requesting a special discount (denoted by R) from A (Fig. 1 (1)). Then, since A has a policy that A provides the service R only for an employee of Company F, A asks Bob to disclose a credential that guarantees Bob is an employee of F (Fig. 1 (2)). Though Bob can prove himself to be an employee of F by disclosing his employee ID B_1, his policy P_{B1} permits disclosing it only to a partner company of F. Hence, he asks A to disclose a credential that guarantees A is a partner of F (Fig. 1 (3)). A discloses its credential A_1 for proving itself to be a partner of F (Fig. 1 (4)). Then Bob discloses his employee ID since the policy P_{B1} is satisfied (Fig. 1 (5)). Finally, A provides the discount for Bob, and the negotiation ends successfully (Fig. 1 (6)).

We consider a transaction between two parties: a user and a server. Each party has its own set of credentials as well as its own policies. A policy is a rule of the form $A \leftarrow B_1 \wedge \ldots \wedge B_n$, which means "credential (or service) A can be disclosed (or provided) to the other party if he/she has disclosed $B_1, \ldots, B_n$." If a service requested by the user is protected by the server's policy, then the server sends back the policy for the service to the user, which is considered to be a request for the credentials in the right-hand side of the policy. Both parties repeatedly send credentials and/or policies to the other party in turn. The transaction ends either when the user obtains the requested service or when any of the parties cannot make the negotiation progress, i.e., the negotiation fails.

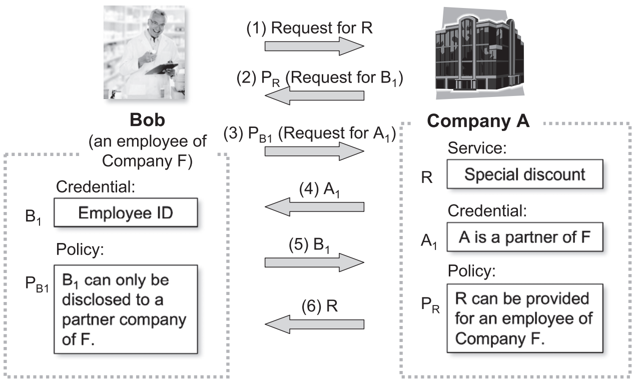
(1) Request for R
(2) P_R (Request for B_1)
(3) P_{B1} (Request for A_1)
(4) A_1
(5) B_1
(6) R
Bob
(an employee of Company F)
Credential:
B_1
Employee ID
Policy:
P_{B1}
B_1 can only be disclosed to a partner company of F.
Company A
Service:
R
Special discount
Credential:
A_1
A is a partner of F
Policy:
P_R
R can be provided for an employee of Company F.

Fig. 1 An example of trust negotiation.

Manuscript received March 26, 2008.
Manuscript revised July 1, 2008.

[†]The author is with Kochi University of Technology, Kami-shi, 782–8502 Japan.
[††]The author is with Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
a) E-mail: takata.yoshiaki@kochi-tech.ac.jp

DOI: 10.1587/transinf.E92.D.200

A *trust negotiation strategy* (of a party i) is a function that answers a set of credentials to disclose to the other party, depending on the policies and credentials of the party i and those disclosed by the other party. Yu et al. [10] proposed a set of strategies, the *DT family*, and showed that it satisfies preferable properties: If each party chooses his/her strategy among the DT family, then (1) a negotiation between the parties always terminates, and (2) whenever there exists a disclosure sequence obeying the policies of the two parties to obtain a requested service, the negotiation will end by the requester obtaining the service. The *disclosure tree strategy* (DTS) is the most cautious strategy in the DT family; that is, it discloses the minimal information to the other party to evolve the negotiation. DTS in a simple implementation requires exponential time and space; however, neither an efficient algorithm nor the lower-bound of its complexity was known.

In this paper, we investigate the computational complexity of DTS. DTS is defined in terms of a set of trees, called *disclosure trees*, which represent the dependency relation among credentials defined by disclosed policies. For example, if a policy $A \leftarrow B_1 \wedge \ldots \wedge B_n$ has been disclosed, then each node labeled with A has children labeled with $B_1, \ldots, B_n$. Considering policies to be the productions of a context-free grammar (CFG), we can formulate the subproblems of DTS as problems on derivation trees of the CFG. Since the other subproblems of DTS are trivial, this paper mainly concerns the following two subproblems:

- EVL: Decide whether there exists a non-*redundant* disclosure tree with a leaf labeled with some credential of the other party. A disclosure tree is *redundant* if there exists a path from the root to a leaf that has two or more nodes with the same label.
- MSET: Find a minimal subset of credentials and policies that makes the set of disclosure trees *evolvable* for the other party. A set of disclosure trees is *evolvable* for one party U if it contains a tree with a leaf labeled with a credential concerned with U.

We show that the EVL is **NP**-complete and MSET is **NP**-hard, while both are solvable in polynomial time if we modify EVL not to require non-redundancy and MSET not to answer any subset useless for leading the negotiation to success.

2. Trust Negotiation and Disclosure Tree Strategy

2.1 Access Control Policy

A *credential* is a digital document that guarantees a subject's attributes and is digitally signed by a trusted party. In this paper, we assume that each credential is concerned with exactly one subject. We say that C is a credential of a party i if C's subject is i. As far as a trust negotiation is concerned, the difference between a service and a credential is that a service does not occur in the right-hand side of a policy. So we treat a service as a special case of a credential in the following definitions. An access control scheme of a party i is a pair $(C_i, \mathcal{P}_i)$ where C_i is a finite set of credentials of i and $\mathcal{P}_i$ is a finite set of *access control policies* (or *policies* for short) defined below. For each credential $C \in C_i$, an access control policy, denoted by $p(C)$, is associated, and $\mathcal{P}_i = \{\, p(C) \mid C \in C_i \,\}$.

Definition 1. A *policy* is a logical expression of the form $C \leftarrow e$ where e is either *true* or a disjunction of conjunctions of credentials, or $\neg C$. If C is a credential of party i, then the subject of each credential in e should be different from i.

The following are sample policies:

$$\begin{aligned} R &\leftarrow (B_1 \wedge B_2) \vee B_3 \vee B_4 \\ B_1 &\leftarrow A_1 \wedge A_2 \\ \neg B_4 & \end{aligned}$$

In the right-hand side of a policy, a credential is used as a propositional symbol that is true if the credential has been disclosed by its subject. For example, the first of the above policies means that providing R to the other party is allowed if B_1 and B_2 have been disclosed or B_3 or B_4 has been disclosed. If the right-hand side of a credential's policy is satisfied by a set C of currently disclosed credentials (including the case that the right-hand side is *true*), then we say the credential is *unlocked* by C.

Definition 2. A credential X is *unlocked* by a set C of credentials, denoted as $C \models X$, if and only if $p(X) = X \leftarrow e$ where $e = true$ or there exists a conjunction $Y_1 \wedge \ldots \wedge Y_k$ in e such that $\{Y_1, \ldots, Y_k\} \subseteq C$.

A policy of the form $\neg C$ is called a *denial policy*, which means that the subject of C does not possess the credential or does not want to disclose the credential in any situation. A policy that is not a denial policy is called a *permission policy*.

2.2 Trust Negotiation

A trust negotiation is performed by two parties: a server (service provider) and a user (service requester). In the following, we fix the parties and call the former Alice and the latter Bob. The access control schemes of Alice and Bob are also fixed. We also fix a service R that Bob requests. A negotiation begins when Bob requests Alice to provide R. Then, Alice performs one of the following actions:

(1) Provide R if R is unlocked;
(2) Disclose a subset of Alice's unlocked credentials and/or a subset of Alice's policies; or
(3) Quit the negotiation.

If Alice chooses (2), then next Bob performs (2) or (3). And then Alice and Bob alternately perform one of the above actions. The negotiation terminates when R is provided to Bob or when one of the parties quits the negotiation. As a result of each action, a *message*, which is either a subset of credentials and policies or a notice of quitting a negotiation, is sent to the other party.

A *negotiation strategy* (or *strategy* for short) is a mapping σ that takes an access control scheme (C, P) and a set D of already disclosed credentials and policies as input and answers a message, i.e., credentials and policies to disclose to the other party (or a notice of quitting). Let (C_A, P_A) (resp. (C_B, P_B)) be the access control scheme of Alice (resp. Bob). A trust negotiation between Alice and Bob (starting from Bob) is formally a (finite or infinite) sequence of messages:

$$\text{REQ}_R, m_{A,0}, m_{B,0}, m_{A,1}, m_{B,1}, \ldots,$$

where REQ_R represents the request for R and each message $m_{i,j}$ ($i \in \{A, B\}$, $j \geq 0$) consists of policies and unlocked credentials. If the sequence is finite, then the last message is either $\{R\}$ or QUIT. Let D_j ($j \geq 0$) be the set of policies and credentials already disclosed up to point j; i.e.,

$$D_0 = \emptyset,\ D_1 = D_0 \cup m_{A,0},\ D_2 = D_1 \cup m_{B,0}, \ldots.$$

Let σ_A (resp. σ_B) be the negotiation strategy of Alice (resp. Bob). The trust negotiation obeying σ_A and σ_B is the one that satisfies $m_{A,j} = \sigma_A((C_A, P_A), D_{2j})$ and $m_{B,j} = \sigma_B((C_B, P_B), D_{2j+1})$ for $j \geq 0$.

If the two parties of a negotiation use arbitrary strategies, then the negotiation does not necessarily terminate. However, if each party chooses a strategy (that can be different from the other party's strategy) in a family of strategies called the DT family[†] [10], then the following good properties hold:

- Negotiations always terminate.
- Whenever there exists a disclosure sequence obeying the policies of the two parties to obtain a requested service, the negotiation will end by the requester obtaining the service.

The *Disclosure Tree Strategy* (DTS) [10], reviewed in the following subsections, is the most *cautious* strategy in the DT family; i.e., it discloses the least number of credentials. However, a simple implementation of the DTS requires exponential time and space, and it is not known whether or not there exists an efficient implementation of the DTS. This paper aims to investigate the computational complexity of the DTS.

2.3 Disclosure Tree

A *disclosure tree* is a tree structure that represents the dependency among credentials defined by a given subset of policies and credentials. For example, Fig. 2 shows the set of disclosure trees for a set $\{R \leftarrow (B_1 \wedge B_2) \vee B_3,\ B_1 \leftarrow A_1 \wedge A_2\}$ of policies. Note that by Definition 1, the right-hand side of the policy $p(A)$ of Alice's credential A only contains Bob's credentials (and vice versa).

Definition 3. A disclosure tree for a set $\mathcal{P}$ of permission policies, a set $\mathcal{P}_d$ of denial policies, and a set C of credentials is a finite, labeled, unordered tree that satisfies the following conditions:

- Each node is labeled with a credential's name.

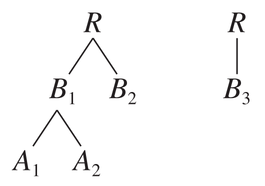

Fig. 2 Sample disclosure trees.

- The label of the root is R.
- Each node u and u's label A satisfy the following:
  - $A \notin C$; i.e., A has not been disclosed yet.
  - $p(A) \notin \mathcal{P}_d$; i.e., disclosing A is not denied.
  - If $p(A) \in \mathcal{P}$ and A is not unlocked by C, then there exists a conjunction $B_1 \wedge \ldots \wedge B_k$ in the right-hand side of $p(A)$ that satisfies the following: Assume that $\{B_1, \ldots, B_k\} \setminus C = \{B_{i_1}, \ldots, B_{i_m}\}$. (Note that $\{B_1, \ldots, B_k\} \nsubseteq C$ because A is not unlocked by C.) Then, u has exactly m children $u_1, \ldots, u_m$ and u_j's label is B_{i_j} for $1 \leq j \leq m$. In other words, the set of the labels of u's children equals the set of the credentials that are requested by A's policy and that have not been disclosed.
  - If $p(A) \notin \mathcal{P}$ or A is unlocked by C, then u is a leaf.

Note that the original definition [10] allows the label of an internal node to be unlocked. We restrict unlocked credentials to labels of leaves because permitting an unlocked node to have children of another disjunction is useless and the results given in the rest of this paper do not depend on this restriction.

We say a disclosure tree t is *redundant* if there exists a path in t from the root to a leaf that contains more than one node with the same label. We write the set of all (redundant and non-redundant) disclosure trees for $D = \mathcal{P} \cup \mathcal{P}_d \cup C$ as $view^{\text{R}}(D)$ and the set of non-redundant disclosure trees as $view^{\text{NR}}(D)$.

Definition 4. If a credential A of a party U is the label of a leaf of a disclosure tree in $view^{\text{NR}}(D)$ (resp. $view^{\text{R}}(D)$), then we say $view^{\text{NR}}(D)$ (resp. $view^{\text{R}}(D)$) is *evolvable* for U, because U can make the negotiation progress by disclosing A itself (if A is unlocked) or A's policy (if A is not unlocked).

The DTS is a strategy that selects a minimal subset m of credentials and policies to disclose so that $view^{\text{NR}}(D \cup m)$, the set of disclosure trees in the next state, will be evolvable for the other party.

Yu et al. [10] only concerned $view^{\text{NR}}(D)$ because of the following reason. For example, let $D = \{R \leftarrow B_1,\ B_1 \leftarrow A_1 \vee A_2,\ A_1 \leftarrow B_1 \wedge B_2\}$, where R, A_1, and A_2 are Alice's credentials and B_1 and B_2 are Bob's ones. Figure 3 shows some of the disclosure trees in $view^{\text{R}}(D)$. Disclosing either B_2 or $p(B_2)$ is not necessary for making R be unlocked, because Bob has to disclose B_1 in any way and disclosing B_1 immediately unlocks R. In this sense, it is better to ignore redundant disclosure trees when selecting credentials and policies

[†]Although Yu et al. named the family the DTS family, we call it the DT family for distinguishability from DTS.

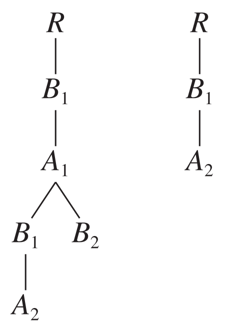

Fig. 3 Redundant and non-redundant disclosure trees.

to disclose. However, the complexity of deciding whether $view^{NR}(D)$ is evolvable for given D is **NP**-complete (Theorem 1), while we can decide whether $view^{R}(D)$ is evolvable in linear time (Proposition 2). Hence we define R-DTS, the same strategy as DTS except that it is based on $view^{R}(D)$, and investigate the complexity of DTS and R-DTS.

2.4 Context-Free Grammar for Representing Disclosure Trees

We can consider given permission policies to be productions of a context-free grammar (CFG) and disclosure trees to be derivation trees of the CFG.

Definition 5. For a subset $\mathcal{P}$ of permission policies, a subset $\mathcal{P}_d$ of denial policies, and a subset C of credentials, we define CFG $G(D)$ for $D = \mathcal{P} \cup \mathcal{P}_d \cup C$ as follows:

- A credential A is a terminal symbol of $G(D)$ if $p(A) \notin \mathcal{P} \cup \mathcal{P}_d$ or A is unlocked by C. Any other credentials are non-terminal symbols of $G(D)$[†].
- The start symbol of $G(D)$ is R.
- The set of productions of $G(D)$ is the smallest set that satisfies the following: If A is a non-terminal symbol such that $p(A) \in \mathcal{P}$ and there exists a conjunction $B_1 \wedge \ldots \wedge B_k$ in the right-hand side of $p(A)$ and $\{B_1, \ldots, B_k\} \setminus C = \{B_{i_1}, \ldots, B_{i_m}\} \neq \emptyset$, then the production set contains $A \to B_{i_1} \ldots B_{i_m}$, where the order of $B_{i_1}, \ldots, B_{i_m}$ in the right-hand side is not important and is determined arbitrarily. (Note that if $\{B_1, \ldots, B_k\} \subseteq C$, then A is unlocked by C and A is a terminal symbol.)

Obviously, a tree t is a derivation tree of $G(D)$ if and only if t is a disclosure tree of $view^{R}(D)$, if we consider t to be an unordered tree. Using this CFG representation, we can reduce problems about the set of disclosure trees to problems about CFG. For example, we can decide whether $view^{R}(D) = \emptyset$ (equivalently, $view^{NR}(D) = \emptyset$) by testing whether the language $L(G(D))$ of $G(D)$ is empty.

2.5 Disclosure Tree Strategy

The DTS in Alice's turn is a mapping defined by the following input and output:

Input: • (C_A, P_A): the access control scheme of Alice.
• D: the set of credentials and policies disclosed by any of the parties so far.

Note that R, the goal of Bob, is in C_A. For (C_A, P_A) and D, we let $LP_A = P_A \setminus D$ (the set of Alice's undisclosed policies) and $LC_A = \{A_i \mid A_i \in C_A \setminus D \text{ and } D \models A_i\}$ (the set of Alice's unlocked, undisclosed credentials).

Output: A message m satisfying the following conditions (1) to (3). The notice of quitting the negotiation is denoted by QUIT.

(1) If either $view^{NR}(D \cup P_A) = \emptyset$ or $view^{NR}(D)$ is not evolvable for Alice, then $m =$ QUIT.
(2) Otherwise, if R is unlocked by credentials in D, then $m = \{R\}$.
(3) Otherwise, m is a non-empty minimal subset of $LP_A \cup LC_A$ such that $view^{NR}(D \cup m)$ is evolvable for Bob.

The DTS in Bob's turn is defined in a similar way. R-DTS is the strategy obtained from DTS by replacing $view^{NR}$ with $view^{R}$.

Although the original DTS defined in [10] answers the set of all non-empty minimal m such that $view^{NR}(D \cup m)$ is evolvable for Bob (and Alice can arbitrarily choose any one message in the set), the number of such m's can be exponential to the description length of $D \cup C_A \cup P_A$. To make discussions on complexity simple, we modified the DTS so that it outputs only one message.

Consider (1), (2), and (3) in the output of DTS above. As mentioned in Sect. 2.4, (1a) deciding whether $view^{R}(D \cup P_A) = \emptyset$ (and $view^{NR}(D \cup P_A) = \emptyset$) can be reduced to the emptiness test of context-free languages, which is solvable in linear time. (2) Deciding whether R is unlocked is easy. Hence, the main problems for solving the DTS (resp. R-DTS) are:

(1b) deciding whether $view^{NR}(D)$ (resp. $view^{R}(D)$) is evolvable for Alice and
(3) computing one arbitrary minimal m such that $view^{NR}(D \cup m)$ (resp. $view^{R}(D \cup m)$) is evolvable for Bob.

We call the former EVL (which is a decision problem) and the latter MSET (which is a function problem). We prefix NR- (resp. R-) to the names of these problems to indicate that $view^{NR}$ (resp. $view^{R}$) is used in their definitions. In the next section we investigate the complexity of these problems.

3. NP-Hardness of DTS

Theorem 1. NR-EVL is **NP**-complete.

Proof. Solving NR-EVL is equivalent to finding a non-redundant derivation tree of $G(D)$ with a leaf labeled with a credential of Alice. This problem is in **NP** because it is solvable by the following non-deterministic polynomial time algorithm: Guess a sequence $C_0 C_1 \ldots C_k$ of symbols of $G(D)$,

[†]If $p(A) \in \mathcal{P}_d$, then A is a non-terminal symbol that does not appear in the left-hand side of any production; i.e., A is a useless symbol.

where $C_0 = R$, C_k is a credential of Alice, and no symbol appears twice or more. Then, check the following:

- C_k is a terminal symbol of $G(D)$.
- For $0 \leq i \leq k-1$, $G(D)$ has a production $C_i \to \alpha C_{i+1}\beta$ for some α and β. Moreover, for each non-terminal symbol C' in $\alpha\beta$, there exists a derivation, which does not include any of $C_0, C_1, \ldots, C_i$, from C' to some string of terminal symbols.

NP-hardness can be shown by a transformation from 3SAT as follows: Let $U = \{x_1, x_2, \ldots, x_m\}$ be the set of variables and $W = \{c_1, c_2, \ldots, c_n\}$ be the set of clauses of an arbitrary instance of 3SAT. Assume that $c_j = \{l_{j1}, l_{j2}, l_{j3}\}$ for $1 \leq j \leq n$. Let D be the set of the following policies:

$$\begin{aligned}
&R \leftarrow B_{x_1}\\
&\left.\begin{aligned}
&B_{x_i} \leftarrow A_{x_i} \vee A_{\overline{x_i}}\\
&A_{x_i} \leftarrow B_{x_{i+1}} \vee B_0\\
&A_{\overline{x_i}} \leftarrow B_{x_{i+1}} \vee B_0
\end{aligned}\right\} 1 \leq i \leq m\\
&B_{x_{m+1}} \leftarrow A_1\\
&A_1 \leftarrow B_{c_1} \wedge B_{c_2} \wedge \ldots \wedge B_{c_n}\\
&B_{c_j} \leftarrow (A_0 \wedge A'_{j1}) \vee (A_0 \wedge A'_{j2}) \vee (A_0 \wedge A'_{j3})\\
&\hspace{12em} 1 \leq j \leq n,\\
&\text{where for } 1 \leq k \leq 3,\ A'_{jk} = \begin{cases} A_{\overline{x_i}} \text{ if } l_{jk} = x_i,\\ A_{x_i} \text{ if } l_{jk} = \overline{x_i}.\end{cases}
\end{aligned}$$

The label of a leaf of a disclosure tree should be either A_0 (a credential of Alice) or B_0 (a credential of Bob). Every disclosure tree with a leaf labeled with A_0 consists of a single path from R to A_1 and a subtree rooted at A_1 (see Fig. 4). The path from R to A_1 should contain either A_{x_i} or $A_{\overline{x_i}}$ for every i. Containing A_{x_i} (resp. $A_{\overline{x_i}}$) in the path encodes the truth assignment $x_i = true$ (resp. *false*). By the definition of the policy of each B_{c_j}, there exists a non-redundant disclosure tree with a leaf labeled with A_0 in $view^{\text{NR}}(D)$ if and only if the instance of 3SAT is satisfiable. □

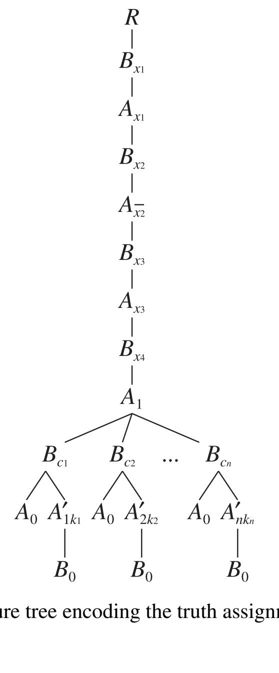

Fig. 4 A disclosure tree encoding the truth assignment $(x_1, x_2, x_3) = (true, false, true)$.

Note that by constructing and storing some data structure related to $view^{\text{NR}}(D)$ at the end of each step, Alice may be able to reduce computation time for her next turn. However, even doing so, Alice cannot solve every step for D in the proof of Theorem 1 in polynomial time, unless **P** = **NP**.

Proposition 2. R-EVL is solvable in $O(n)$ time, where n is the description length of D.

Proof. We can decide whether $view^{\text{R}}(D)$ is evolvable for Alice by testing whether $L(G(D))$ contains a string of terminal symbols that contains at least one of Alice's credentials. This test can be achieved by testing whether the intersection of $L(G(D))$ and a regular language R_A, which is the set of all the strings that contains at least one of Alice's credentials, is empty. This emptiness test can be performed in $O(nm^3)$ time, where m is the number of states of a finite automaton that recognizes R_A. Since R_A can be recognized by a finite automaton with two states, this proposition holds. □

Theorem 3. R-MSET is **NP**-hard.

Theorem 4. NR-MSET is **NP**-hard.

Proof of Theorems 3 and 4 is given in Sect. A.1. Note that although Theorem 4 can be proved by the same proof as Theorem 3, we provide another proof for Theorem 4 because the latter proof holds even under the monotonicity condition described in Sect. 4.

Also note that for D and LP_A in the proof of Theorem 3, $view^{\text{R}}(D \cup LP_A) = \emptyset$. In the definition of R-DTS, R-MSET is performed after checking $view^{\text{R}}(D \cup LP_A) \neq \emptyset$, and thus R-MSET would not be performed as a part of R-DTS for such D and LP_A. On the other hand, D' and LP_B in the proof of Theorem 4 satisfies $view^{\text{NR}}(D' \cup LP_B) \neq \emptyset$ and $view^{\text{NR}}(D')$ is evolvable for Bob.

Now we consider an upper bound of the complexity of R- and NR-MSET. For a class $\mathcal{X}$ of problems, let $\mathbf{NP}^{\mathcal{X}}$ be the class of problems solvable by a polynomial-time nondeterministic oracle Turing machine (PT-NOTM) with oracle for a problem in $\mathcal{X}$.

Corollary 5. R-MSET is in $\mathbf{NP}^{\mathbf{NP}}$ and NR-MSET is in $\mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$.

Proof. Since R-EVL is solvable in polynomial time, the following problem R-NONMINIMAL-MSET is in **NP**:

INPUT: D and $m \subseteq LP_A \cup LC_A$.
OUTPUT: Does there exist some $m' \subseteq m$ such that $m' \neq \emptyset$ and $view^{\text{R}}(D \cup m')$ is evolvable for Bob?

Since R-MSET can be solved by a PT-NOTM with oracle for R-NONMINIMAL-MSET, R-MSET $\in \mathbf{NP}^{\mathbf{NP}}$. NR-MSET $\in \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$ can be shown in a similar way. □

The complexity of the subproblems are summarized as follows:

	NR-	R-
EVL	**NP**-complete	$O(n)$ time
MSET	**NP**-hard in $\mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$	**NP**-hard in $\mathbf{NP}^{\mathbf{NP}}$

Note that in the proof of **NP**-hardness of R-MSET (Theorem 3), we do not depend on minimality of the solution m of R-MSET. We think that the minimality condition imposes higher complexity than **NP** on R-MSET. Since **NP**-hardness of NR-MSET (Theorem 4) can be proved by the same proof of Theorem 3, we think the complexity of NR-MSET is also higher than **NP**. Improving the lower bound of the complexity of R- and NR-MSET is future work.

4. Polynomial Solvability under Practical Assumptions

In this section, we give a polynomial-time algorithm for R-MSET under the following practical modification on the definition of MSET:

Monotonicity: In the definition of R-MSET (resp. NR-MSET), the output m should be a minimal non-empty subset of $LP_A \cup LC_A$ such that $view^{\mathrm{R}}(D \cup m')$ (resp. $view^{\mathrm{NR}}(D \cup m')$) is evolvable for Bob for any m' satisfying $m \subseteq m' \subseteq m \cup LP_A$.

We denote R-MSET (resp. NR-MSET) under monotonicity as M-R-MSET (resp. M-NR-MSET). M-R-DTS is the strategy obtained from R-DTS by replacing R-MSET with M-R-MSET.

As shown in the following example, any solution m of R-MSET (or NR-MSET) that does not satisfy monotonicity is useless for unlocking R. In this sense, excluding solutions that do not satisfy monotonicity is practical.

Example 1. Let

$$\begin{aligned} D = \{ \quad & R \leftarrow (B_1 \wedge B_2) \vee B_3,\ B_1 \leftarrow A_1, \\ & B_2 \leftarrow A_2,\ B_3 \leftarrow A_3\}, \\ LP_A = \{ & A_1 \leftarrow B_4,\ A_2 \leftarrow B_2,\ A_3 \leftarrow B_5\}. \end{aligned}$$

In the definition of R-MSET in the previous section, $m = \{A_1 \leftarrow B_4\}$ is a correct solution; however, it is not a solution under monotonicity, since in $view^{\mathrm{R}}(D \cup \{A_1 \leftarrow B_4, A_2 \leftarrow B_2\})$, there no longer exists a disclosure tree with a leaf labeled with Bob's credential due to the policies $B_2 \leftarrow A_2$ and $A_2 \leftarrow B_2$ (see Fig. 5). (Each disclosure tree should be finite by Definition 3, and thus the above cyclic policies eliminate trees containing B_4.) Such m is not useful for unlocking R, since the effect of the disclosure of m will finally be canceled by the disclosure of other policies.

Example 2. Let

$$\begin{aligned} D = \{ \quad & R \leftarrow (B_1 \wedge B_2) \vee B_3,\ B_1 \leftarrow A_1,\ B_2 \leftarrow A_2, \\ & B_3 \leftarrow A_3 \vee A_4,\ B_4 \leftarrow A_5,\ A_3 \leftarrow B_3 \wedge B_4\}, \\ LP_A = \{ & A_1 \leftarrow B_4,\ A_2 \leftarrow B_2,\ A_4 \leftarrow B_5,\ A_5 \leftarrow B_6\}. \end{aligned}$$

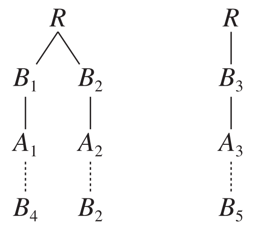

Fig. 5 Disclosure trees for showing the effect of monotonicity.

Figure 6 shows disclosure trees in $view^{\mathrm{R}}(D)$, augmented with policies in LP_A denoted by dotted lines. For these D and LP_A, $\{A_1 \leftarrow B_4\}$ is not a solution of M-R-DTS for the same reason as Example 1, and $\{A_5 \leftarrow B_6\}$ is not a solution of DTS since there is no disclosure tree with A_5 in $view^{\mathrm{NR}}(D)$. The following table shows correct solutions of DTS, R-DTS, and M-R-DTS, respectively.

	DTS	R-DTS	M-R-DTS
$\{A_1 \leftarrow B_4\}$	✓	✓	
$\{A_4 \leftarrow B_5\}$	✓	✓	✓
$\{A_5 \leftarrow B_6\}$		✓	✓

Since the proof of Theorem 4 in Sect. A.1 remains valid even under monotonicity, M-NR-MSET is **NP**-hard. Below we discuss polynomial solvability of M-R-MSET. Outline of the rest of this section is as follows:

1. We first show that under monotonicity, it is sufficient to consider disclosure trees without a label that is "useless" in $view^{\mathrm{R}}(D \cup LP_A)$ (Lemma 7). The concept of uselessness is borrowed from that of CFGs.
2. We define $UReq(B)$ as the family of minimal subsets of credentials that unlock B. We also define the *reachability graph* where nodes are credentials and an edge (x, y) means that x can reach y via an undisclosed policy followed by a path consisting of already disclosed policies.
3. By the reachability graph, we characterize a minimal set that Alice needs to disclose for constructing a tree evolvable for Bob by "connecting" already disclosed credentials and policies (Lemmas 8 and 9).

At first we define some additional concepts. We fix (C_A, P_A) and D. If a credential B of Bob is unlocked under $D \cup C$ for some subset C of Alice's undisclosed credentials, then we call C a *requested set* of B. For example, if $p(B) = B \leftarrow (A_1 \wedge A_2) \vee (A_2 \wedge A_3)$ and A_2 is disclosed in D, then the minimal requested sets of B are $\{A_1\}$ and $\{A_3\}$. We define $UReq(B)$ as the family of the minimal requested sets of B whose members are unlocked by credentials in D. In the above example, if A_1 and A_3 are unlocked by credentials in D, then $UReq(B) = \{\{A_1\}, \{A_3\}\}$. As a special case, if B is already unlocked by credentials in D, then $UReq(B) = \{\emptyset\}$. For convenience, we define $UReq(B) = \{\emptyset\}$ if $p(B)$ is undisclosed. Note that $UReq(B)$ is defined for a fixed D

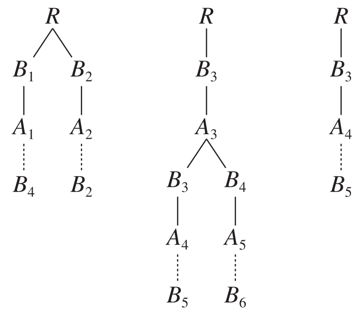

Fig. 6 Disclosure trees for Example 2.

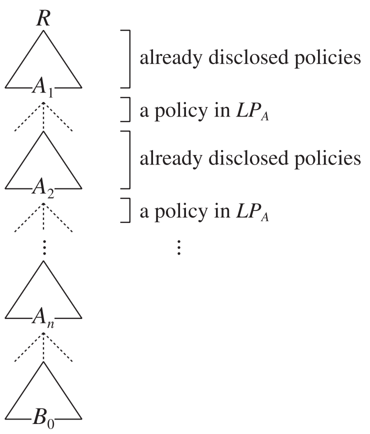

Fig. 7 Sequence of policies in LP_A that makes $view^{\mathrm{R}}(D \cup m)$ evolvable for Bob.

and is also fixed. Also note that $C \in UReq(B)$ implies $C \subseteq LC_A$ (see Sect. 2.5). By definition, we can easily prove the following lemma.

Lemma 6. If a credential B of Bob is the label of a leaf of a disclosure tree in $view^{\mathrm{R}}(D \cup m)$ for Alice's message m, then there exists $C \in UReq(B)$ such that $C \subseteq m$. □

In a CFG, a symbol Y is *reachable* from a non-terminal symbol X if $\alpha Y \beta$ can be derived from X for some α and β. X is *generating* if some string of terminal symbols can be derived from X. Let UL be the set of symbols in $G(D \cup LP_A)$ that are either non-generating or unreachable from R. In other words, UL is the set of useless symbols; i.e., any disclosure tree in $view^{\mathrm{R}}(D \cup LP_A)$ does not contain any symbol in UL as a label. The following lemma says that under monotonicity, it is sufficient to consider disclosure trees without symbols in UL. Proof of Lemma 7 is given in Sect. A.2.

Lemma 7. Let m be a subset of $LP_A \cup LC_A$ and assume that $view^{\mathrm{R}}(D \cup m')$ is evolvable for Bob for any m' such that $m \subseteq m' \subseteq m \cup LP_A$. Then, $view^{\mathrm{R}}(D \cup m)$ contains a disclosure tree without symbols in UL and with a leaf labeled with a credential of Bob.

Using Lemma 7, we would like to find m such that $view^{\mathrm{R}}(D \cup m)$ contains a disclosure tree without symbols in UL and with a leaf labeled with a credential of Bob. To do so, we have to find undisclosed policies that "connect" already disclosed policies and make a path from R to some B_0 such that $UReq(B_0) \neq \emptyset$ (see Fig. 7). To characterize a sequence $p(A_1), \ldots, p(A_n)$ of policies to disclose for constructing a path from R to B_0, below we define a directed graph called a *reachability graph*. Intuitively, the reachability graph has an edge (x, y) if y is reachable (only using productions without symbols in UL) from x when $p(x)$ is disclosed (Fig. 8).

Let G_U be the CFG obtained from $G(D)$ by removing the productions that contains one or more elements of UL. Thus each derivation tree of G_U represents a disclosure tree

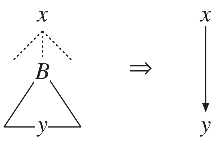

Fig. 8 Edge of the reachability graph.

under D without symbols in UL. We define the *reachability graph* (V, E) of D as a directed graph as follows:

- Let v_0 be a new symbol, V_A be the set of the left-hand sides of the policies in LP_A, and V_B be the set of Bob's credentials. We define $V = \{v_0\} \cup V_A \cup V_B$.
- E is the smallest set that satisfies the following: For each $x \in V_A \cup V_B$, $(v_0, x) \in E$ if x is reachable from R in G_U. For each $x \in V_A$ and $y \in V_A \cup V_B$, $(x, y) \in E$ if there is a credential B such that (1) the right-hand side of $p(x)$ contains a conjunction that contains B and does not contain any element of UL and (2) y is reachable from B in G_U.

By the definition of E, each vertex in V_B has no outgoing edge. For each $x \in V_A$, $(x, y) \in E$ for some y if and only if y becomes reachable from x, only using productions without symbols in UL, when $p(x)$ is disclosed. For convenience, we define $UReq(x) = \bigcup_{B \in V_B,\, (x,B) \in E} UReq(B)$ for $x \in V_A$. By definition, $UReq(x) \neq \emptyset$ for $x \in V_A$ if and only if $(x, B) \in E$ for some $B \in V_B$ such that $UReq(B) \neq \emptyset$.

Lemma 8. Let m be a subset of $LP_A \cup LC_A$ and assume that $view^{\mathrm{R}}(D \cup m')$ is evolvable for Bob for any m' such that $m \subseteq m' \subseteq m \cup LP_A$. Then, there exists a subset m'' of m that satisfies the following properties for some vertices $v_1, \ldots, v_n \in V_A$:

(i) $m'' \cap LP_A = \{p(v_1), \ldots, p(v_n)\}$,
(ii) $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq n-1$,
(iii) $(v_i, v_j) \notin E$ for $0 \leq i$ and $i+2 \leq j \leq n$,
(iv) $m'' \cap LC_A \in UReq(v_n)$, and
(v) No element of $UReq(v_i)$ is a proper subset of $m'' \cap LC_A$ for $0 \leq i \leq n$.

Proof. By Lemma 7, $view^{\mathrm{R}}(D \cup m)$ contains a disclosure tree t without symbols in UL and with a leaf u labeled with a credential B of Bob. Let $v_1, v_2, \ldots, v_n$ be the credentials in V_A that appear in this order on the path in t from the root to u. By the definition of E, $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq n-1$. If $(v_i, v_j) \in E$ for some i and j such that $0 \leq i$ and $i+2 \leq j \leq n$, then there exists a disclosure tree in $view^{\mathrm{R}}(D \cup m)$ with a leaf u' labeled with B such that the credentials in V_A that appear on the path from the root to u' are $v_1, \ldots, v_i, v_j, \ldots, v_n$. Since we can choose this tree as t, we can assume that $(v_i, v_j) \notin E$ for $0 \leq i$ and $i+2 \leq j \leq n$. Let $m_p = \{p(v_1), \ldots, p(v_n)\}$. Since the policy of each element of V_A is not a member of D by the definition of V_A, $m_p \subseteq m$.

Since B is the label of a leaf of t, by Lemma 6 there exists $C' \in UReq(B)$ such that $C' \subseteq m$. If a proper subset C'' of C' is a member of $UReq(v_i)$ for some i such that $0 \leq i \leq n$, then there exists $B' \in V_B$ such that $(v_i, B') \in E$ and

$C'' \in UReq(B')$ (by the definition of $UReq(v_i)$). Moreover, there exists a disclosure tree in $view^{\mathrm{R}}(D \cup m)$ with a leaf u' labeled with B' such that the credentials in V_A that appear on the path from the root to u' are $v_1, \ldots, v_i$. Since we can choose this tree as t and let $n = i$, we can assume that no member of $UReq(v_i)$ is a proper subset of C' for $0 \le i \le n$.

Let $m'' = m_p \cup C'$. By the above discussion, m'' and $v_1, \ldots, v_n$ satisfies (i)–(v). □

Lemma 9 (the converse of Lemma 8). Let m'' be a subset of $LP_A \cup LC_A$ that satisfies properties (i), (ii), and (iv) in Lemma 8 for some vertices $v_1, \ldots, v_n \in V_A$. Then, $view^{\mathrm{R}}(D \cup m)$ is evolvable for Bob for every m such that $m'' \subseteq m \subseteq m'' \cup LP_A$.

Proof. For each m, we can construct a disclosure tree in $view^{\mathrm{R}}(D \cup m)$ with a leaf labeled with a credential of Bob and without symbols in UL. □

Let EM be the set of all $m \subseteq LP_A \cup LC_A$ such that $view^{\mathrm{R}}(D \cup m')$ is evolvable for Bob for every m' that satisfies $m \subseteq m' \subseteq m \cup LP_A$. Also let M be the set of all m'' that satisfies properties (i)–(v) in Lemma 8. M-R-MSET is the problem to find one of the minimal elements of EM. By the following lemma, M-R-MSET is equivalently the problem to find one arbitrary element of M.

Lemma 10. M equals the set of the minimal elements of EM.

Proof. Lemma 9 can be rephrased as $M \subseteq EM$. Lemma 8 can be rephrased as that every $m \in EM$ has some subset $m'' \in M$. By definition, any proper subset of an element of M is not a member of M. By these facts, this lemma holds. □

Theorem 11. M-R-MSET is solvable in $O(n^2)$ time, where n is the description length of $D \cup LP_A$.

Proof. If $view^{\mathrm{R}}(D \cup LP_A) = \emptyset$, then there exists no m to output. If R is unlocked under D, then any m cannot make $view^{\mathrm{R}}(D \cup m)$ evolvable for Bob. Hence we assume that $view^{\mathrm{R}}(D \cup LP_A) \neq \emptyset$ and R is not unlocked by credentials in D. Let t be an arbitrary tree in $view^{\mathrm{R}}(D \cup LP_A)$. Then t does not contain symbols in UL but contains a credential B of Bob such that $UReq(B) \neq \emptyset$ by Lemma 6. (Note that $UReq(B)$ may be $\{\emptyset\}$.) There thus exists a vertex $v \in V_A$ in the reachability graph of D such that $UReq(v) \neq \emptyset$ and a path from v_0 to v exists.

The following polynomial-time algorithm outputs one element of M:

(1) Construct the reachability graph of D.
(2) Choose a sequence $v_0, v_1, \ldots, v_n$ of vertices in V_A as follows: Perform the depth-first search on the reachability graph starting from v_0. When a vertex $v \in V_A$ such that $UReq(v) \neq \emptyset$ is found, let the sequence $v_0, v_1, \ldots, v_n = v$ be the path from v_0 to v. (Thus $UReq(v_i) = \emptyset$ for $1 \le i \le n-1$.) By the above discussion, such a path should be found.
(3) If $(v_i, v_j) \in E$ for some i and j such that $0 \le i$ and $i+2 \le j \le n$, then remove $v_{i+1}, \ldots, v_{j-1}$ from the sequence. Repeat this step until no such i and j exist.
(4) Choose an arbitrary one of the minimal elements of $UReq(v_n)$. Let C' be the chosen element.
(5) Output $m = \{p(v_1), \ldots, p(v_n)\} \cup C'$.

The size $|V|$ of V is $O(n)$ and the size $|E|$ of E is $O(n^2)$. Each of the above steps (1)–(5) can be performed in $O(|V|^2 + |E|) = O(n^2)$ time. □

The complexity of the subproblems under monotonicity are summarized as follows:

	M-NR-	M-R-
MSET	**NP**-hard	$O(n^2)$ time

5. Discussion

5.1 Relation among DT Family, R-DTS and M-R-DTS

We review the definition of the DT family and discuss the relationship among the DT family, R-DTS, and M-R-DTS.

Yu et al. [10] defined the DT family based on the following definition of 'cautiousness.'

Definition 6 (cautiousness). For given strategies σ_1 and σ_2 and all possible inputs (C_A, P_A) and D to σ_1 and σ_2, if for every correct answer m of $\sigma_2((C_A, P_A), D)$ there exists an answer m' of $\sigma_1((C_A, P_A), D)$ such that $m' \subseteq m$, then we say σ_1 is *at least as cautious as* σ_2, denoted as $\sigma_1 \preceq \sigma_2$. Note that in this definition, we consider QUIT to be $\emptyset$.

The DT family is defined as $\{\sigma \mid \mathrm{DTS} \preceq \sigma\}$, i.e., the set of strategies such that DTS is at least as cautious as them.

As shown in Example 2, DTS $\not\preceq$ R-DTS since an answer of R-DTS (or its subset) is not an answer of DTS. DTS $\not\preceq$ M-R-DTS for the same reason. Thus either R-DTS or M-R-DTS is not in the DT family, and the properties of the DT family described in Sect. 2.2 do not necessarily hold if one party uses R-DTS or M-R-DTS and the other uses a strategy of the DT family. However, we can define another family of strategies, R-DT family, as $\{\sigma \mid \text{R-DTS} \preceq \sigma\}$. The R-DT family contains R-DTS and M-R-DTS, and if each party chooses a strategy in the family, then the same properties as the DT family hold.

Note that R-DTS $\preceq$ M-R-DTS since if m is an answer of M-R-DTS, then m is also an answer of R-DTS or otherwise a non-empty subset of m is an answer of R-DTS. On the other hand, M-R-DTS $\not\preceq$ R-DTS as shown in Example 2. Thus we can say R-DTS is more cautious than M-R-DTS. However, as shown in Example 1, R-DTS (and DTS) may answer a message useless for unlocking R, and thus cautiousness does not mean that each message only contains useful policies for unlocking R.

5.2 Relation among DTS, Formal Language Theory, and Propositional Logic

Using the terminology in formal language theory, NR-EVL is equivalent to the following problem:

INPUT: a CFG G and a subset Σ_A of terminal symbols of G.
OUTPUT: Does there exist a non-redundant derivation tree containing a symbol in Σ_A?

We showed this problem is **NP**-complete (Theorem 1). R-EVL is a variation of NR-EVL that does not require non-redundancy and can be reduced to the emptiness problem of CFG (Theorem 2).

Similarly, (an essential part of) NR-MSET can be rephrased as follows:

INPUT: a CFG $G = (N, \Sigma, P_1, S)$ where N is the set of non-terminal symbols, Σ is the set of terminal symbols, P_1 is the set of productions, and S is the start symbol, and another set P_2 of productions over N and Σ such that $P_1 \cap P_2 = \emptyset$, and a subset $\Sigma_B \subseteq \Sigma$.
OUTPUT: a minimal non-empty subset m of P_2 such that $G' = (N, \Sigma, P_1 \cup m, S)$ has a non-redundant derivation tree containing a symbol in Σ_B.

R-MSET is a variation of NR-MSET obtained by removing the word 'non-redundant.' Both NR- and R-MSET are **NP**-hard.

For NR-EVL, the authors could not find any similar problem in literature on formal language theory and complexity [11]–[13] or on regular tree languages [14]. (Note that the set of derivation trees of a CFG can be seen as a regular tree language.) It seems that in formal language theory, a problem such as NR-EVL in which non-redundant trees are important is rare.

Similarly, for NR- and R-MSET, there is no similar result in literature on propositional logic [15]. Since policies are defined as propositional formulae and disclosure trees are similar to proof trees, these problems seem to be related to propositional logic. However, while most studies on propositional logic concern algorithms and complexity for deciding satisfiability of formulae, NR- and R-MSET do not concern satisfiability. MSET can be rephrased as follows using the terminology of propositional logic:

INPUT: two set P_A, P_B of propositional formulae.
OUTPUT: Suppose that parties A and B possess P_A and P_B, respectively, and they add a subset of his/her formulae to a proof tree in turn. The question is to find a minimal subset of formulae that prevent the other party from reaching deadlock.

This problem seems to be a problem in game theory rather than propositional logic. However, MSET is different from a typical problem in game theory such as finding a move that force the other party into deadlock.

6. Conclusion

In this paper, we discussed the computational complexity of a trust negotiation strategy called DTS. We formulated subproblems of DTS as problems on derivation trees of a context-free grammar (CFG), and analyzed the computational complexity of the subproblems using the concepts of CFGs. As a result, we showed that the original DTS is **NP**-hard while it is polynomially solvable if we modify it to prohibit any useless disclosure for leading the negotiation to success.

Improving the gap between the lower and upper bound of the complexity is future work. Future work also includes the analysis of trust negotiation strategies from a point of view other than computational complexity, e.g., the number of credentials disclosed in one negotiation, the number of actions required for one negotiation, and so on.

References

[1] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis, "The KeyNote trust-management system version 2," RFC 2704, Sept. 1999.
[2] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," IEEE Security and Privacy, pp.164–173, June 1996.
[3] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid, "Access control meets public key infrastructure: Or, assigning roles to strangers," IEEE Security and Privacy, pp.2–14, May 2000.
[4] S. Ruohomaa and L. Kutvonen, "Trust management survey," International Conference on Trust Management, pp.77–92, May 2005.
[5] H. Koshutanski and F. Massacci, "Interactive credential negotiation for stateful business processes," International Conference on Trust Management, pp.256–272, May 2005.
[6] T. Ryutov, L. Zhou, B.C. Neuman, T. Leithead, and K.E. Seamons, "Adaptive trust negotiation and access control," ACM Symposium on Access Control Models and Technologies, pp.139–146, June 2005.
[7] W.H. Winsborough and N. Li, "Towards practical automated trust negotiation," IEEE International Workshop on Policies for Distributed Systems and Networks, pp.92–103, June 2002.
[8] W.H. Winsborough and N. Li, "Safety in automated trust negotiation," IEEE Security and Privacy, pp.147–160, May 2004.
[9] M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating trust on the Web," IEEE Internet Comput., vol.6, no.6, pp.30–37, Nov. 2002.
[10] T. Yu, M. Winslett, and K.E. Seamons, "Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation," ACM Trans. Information and System Security, vol.6, no.1, pp.1–42, Feb. 2003.
[11] G. Rozenberg and A. Salomaa, eds., Handbook of Formal Languages: vol.1. Word, Language, Grammar, Springer, 1997.
[12] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., 1979.
[13] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.
[14] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi, Tree automata techniques and applications, 2004. Available at http://tata.gforge.inria.fr/
[15] H.K. Büning and T. Lettmann, Propositional Logic: Deduction and Algorithms, Cambridge Tracts in Theoretical Computer Science, vol.48, Cambridge University Press, 1999.

Appendix: Proof of Theorems and Lemmas

A.1 Proof of Theorems 3 and 4

Proof of Theorem 3. We show the **NP**-hardness by a transformation from 3SAT. Let (U, W) be an arbitrary instance of 3SAT where $U = \{x_1, x_2, \ldots, x_{n'}\}$ be the set of variables and $W = \{c_1, c_2, \ldots, c_n\}$ be the set of clauses. Define D and LP_A as follows:

$$D: \quad \begin{aligned} & R \leftarrow B_{x_1} \\ & B_{x_i} \leftarrow A_{x_i} \wedge A_{\overline{x_i}} \qquad 1 \le i \le n' \\ & B_{x_{n'+1}} \leftarrow A'_{11} \wedge A'_{12} \wedge A'_{13} \\ & B'_{jk} \leftarrow A''_{jk} \wedge A'_{j+1,1} \wedge A'_{j+1,2} \wedge A'_{j+1,3} \\ & \qquad \text{for } 1 \le j \le n \text{ and } 1 \le k \le 3, \\ & \text{where } A''_{jk} = \begin{cases} A_{\overline{x_i}} \text{ if } l_{jk} = x_i, \\ A_{x_i} \text{ if } l_{jk} = \overline{x_i}. \end{cases} \end{aligned}$$

$$LP_A: \quad \begin{aligned} & \left.\begin{aligned} A_{x_i} &\leftarrow B_{x_{i+1}} \\ A_{\overline{x_i}} &\leftarrow B_{x_{i+1}} \end{aligned}\right\} 1 \le i \le n' \\ & A'_{jk} \leftarrow B'_{jk} \qquad 1 \le j \le n,\ 1 \le k \le 3 \\ & A'_{n+1,1} \leftarrow B_0 \end{aligned}$$

For these D and LP_A, Alice should have no unlocked credentials; i.e., $LC_A = \emptyset$.

For example, consider a sample instance of 3SAT in which $n = 3$ and $W = \{\{x_1, \overline{x_2}, \overline{x_3}\}, \{\overline{x_1}, x_2, x_3\}, \{\overline{x_1}, \overline{x_2}, \overline{x_3}\}\}$. Figure A· 1 shows a disclosure tree in $view^{\mathrm{R}}(D \cup m)$ for some $m \subseteq LP_A$. Dotted lines denote policies not in D. The upper part of the tree above B_{x_4} encodes a truth assignment. The rest of the tree represents which literal is true in each clause. In Fig. A· 1, l_{11} ($= x_1$), l_{23} ($= x_3$), and l_{32} ($= \overline{x_2}$) are true.

Let m be a subset of $LP_A \cup LC_A$ ($= LP_A$) such that $view^{\mathrm{R}}(D \cup m)$ is evolvable for Bob. Among Bob's credentials, only B_0 can be the label of a leaf of a disclosure tree in $view^{\mathrm{R}}(D \cup m)$. Hence m should contain either $p(A_{x_i})$ or $p(A_{\overline{x_i}})$ for each i (that encodes the truth assignment for x_i); otherwise, $view^{\mathrm{R}}(D \cup m)$ would not contain any disclosure tree with B_0. By the same reason, m should contain $p(A'_{j1})$ or $p(A'_{j2})$ or $p(A'_{j3})$ for each j ($1 \le j \le n$). Suppose that $p(A'_{jk}) \in m$ and $A''_{jk} = A_{\overline{x_i}}$ (i.e., $l_{jk} = x_i$). Then, m contains $p(A_{x_i})$; otherwise, m contains $p(A_{\overline{x_i}})$ instead, implying that infinite recursion would occur and $view^{\mathrm{R}}(D \cup m)$ would be empty. In this transformation $p(A_{x_i}) \in m$ means $x_i = true$ and $p(A'_{jk}) \in m$ means that the kth literal of c_j evaluates to true. Therefore if there exists a disclosure tree with B_0 in $view^{\mathrm{R}}(D \cup m)$, then each clause c_j evaluates to true under the truth assignment represented by the upper part of the tree, and thus the instance of 3SAT is satisfiable. The converse direction can be proved in a similar way.

Thus there exists $m \subseteq LP_A \cup LC_A$ such that $view^{\mathrm{R}}(D \cup m)$ is evolvable for Bob if and only if the instance of 3SAT is satisfiable. □

Proof of Theorem 4. In this proof, we exchange the roles of Alice and Bob; i.e., we assume that Bob is performing DTS to select m.

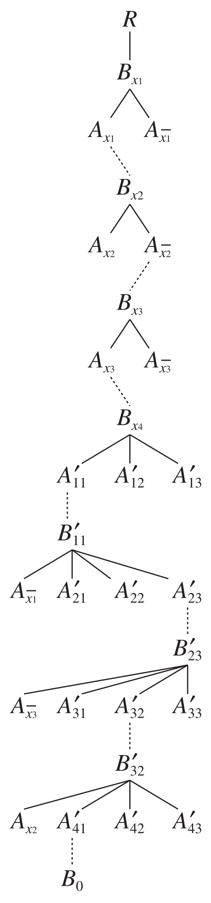

Fig. A· 1 A disclosure tree for the set $\{\{x_1, \overline{x_2}, \overline{x_3}\}, \{\overline{x_1}, x_2, x_3\}, \{\overline{x_1}, \overline{x_2}, \overline{x_3}\}\}$ of clauses and the truth assignment $(x_1, x_2, x_3) = (true, false, true)$.

Let D' be the union of D in the proof of Theorem 1 and $\{A_2 \leftarrow B_2\}$, where A_2 and B_2 are new symbols. Also let the set LP_B of Bob's undisclosed policies be $\{B_0 \leftarrow A_2,\ B_2 \leftarrow A_3\}$, where A_3 is another new symbol. If $view^{\mathrm{NR}}(D')$ contains a disclosure tree with a leaf labeled with A_0, i.e., $view^{\mathrm{NR}}(D')$ is already evolvable for Alice, then $view^{\mathrm{NR}}(D' \cup m)$ is also evolvable for Alice for any $m \subseteq LP_B$. The minimal non-empty solutions are therefore $m = \{B_0 \leftarrow A_2\}$ and $m = \{B_2 \leftarrow A_3\}$. On the other hand, if $view^{\mathrm{NR}}(D')$ does not contain a disclosure tree with a leaf labeled with A_0, then $view^{\mathrm{NR}}(D' \cup m)$ is evolvable for Alice only if $m = LP_B$; otherwise, there never exists a path from R to A_3, which is the sole candidate of the label of a leaf of a disclosure tree among Alice's credentials in this case (see Fig. A· 2). Thus we can decide whether or not the instance of 3SAT is satisfiable by finding a minimal m such that $view^{\mathrm{NR}}(D' \cup m)$ is evolvable for Alice and by checking whether $m \neq LP_B$. □

A.2 Proof of Lemma 7

At first we show the following auxiliary lemma.

Lemma 12. Let m be a subset of $LP_A \cup LC_A$.

(1) If X is reachable from R in $G(D \cup m)$, then X is reach-

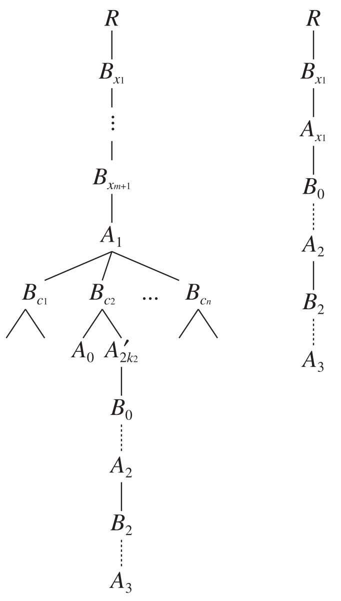

Fig. A·2 Disclosure trees for the proof of Theorem 4.

able from R also in $G(D \cup LP_A)$.

(2) If X is unlocked by credentials in $D \cup m$, then X is generating in $G(D \cup LP_A)$.

(3) If a disclosure tree $t \in view^{\mathrm{R}}(D \cup m)$ contains a node labeled with a symbol in UL, then t also contains a leaf labeled with a symbol in UL.

Proof. (1) $D \cup LP_A$ includes more policies and less credentials than $D \cup m$, and thus $G(D \cup LP_A)$ preserves the reachability in $G(D \cup m)$.

(2) If $p(X) \notin D \cup LP_A$, then X is a terminal symbol of $G(D \cup LP_A)$ and thus X is generating in $G(D \cup LP_A)$. Hence we assume $p(X) \in D \cup LP_A$. Since X is unlocked by credentials in $D \cup m$, the right-hand side of $p(X)$ contains a conjunction $B_1 \wedge \ldots \wedge B_k$ such that $\{B_1, \ldots, B_k\} \subseteq D \cup (m \cap LC_A)$. Thus $G(D \cup LP_A)$ contains a production $X \rightarrow B_{i_1} \ldots B_{i_j}$ where $\{B_{i_1}, \ldots, B_{i_j}\} = \{B_1, \ldots, B_k\} \setminus D \subseteq LC_A$. Since all credentials in LC_A are unlocked by D, $B_{i_1}, \ldots, B_{i_j}$ are terminal symbols of $G(D \cup LP_A)$, and thus X is generating in $G(D \cup LP_A)$.

(3) Let u be a node of t labeled with a symbol X in UL. Since X is a label in a disclosure tree, X is reachable from R in $G(D \cup m)$, and thus X is reachable from R also in $G(D \cup LP_A)$ by (1). X is therefore non-generating in $G(D \cup LP_A)$.

Below we show a sufficient condition: if u is not a leaf, then u has a child labeled with a symbol in UL. If u is not a leaf, then $p(X) \in D \cup m$ and thus $p(X) \in D \cup LP_A$. Assume that a production $X \rightarrow Y_1 \ldots Y_j$ of $G(D \cup m)$ is applied to u; i.e., the children of u are labeled with $Y_1, \ldots, Y_j$. Then, the right-hand side of $p(X)$ should contain a conjunction $Y_1 \wedge \ldots \wedge Y_j \wedge Z_1 \wedge \ldots \wedge Z_k$ such that $\{Z_1, \ldots, Z_k\} \subseteq D \cup m$. $G(D \cup LP_A)$ therefore has a production $X \rightarrow Y_1 \ldots Y_j Z_{i_1} \ldots Z_{i_l}$ where $\{Z_{i_1}, \ldots, Z_{i_l}\} = \{Z_1, \ldots, Z_k\} \setminus D$. $Z_{i_1}, \ldots, Z_{i_l}$ are members of $m \cap LC_A$ and thus unlocked by credentials in D. Therefore $Z_{i_1}, \ldots, Z_{i_l}$ are generating in $G(D \cup LP_A)$ by (2). Since X is non-generating in $G(D \cup LP_A)$, at least one of $Y_1, \ldots, Y_j$ is non-generating in $G(D \cup LP_A)$; i.e., u has a child labeled with a symbol in UL. □

Proof of Lemma 7. We show this lemma by proving its contraposition. Assume that every disclosure tree in $view^{\mathrm{R}}(D \cup m)$ with a leaf labeled with a credential of Bob contains a node labeled with an element of UL. By Lemma 12 (3), such a tree contains a leaf labeled with some $X \in UL$. Since X is reachable from R in $G(D \cup m)$, X is non-generating in $G(D \cup LP_A)$ by Lemma 12 (1) and thus X should be a credential that is not unlocked by credentials in $D \cup m$ by Lemma 12 (2) and whose policy $p(X)$ is in $D \cup LP_A$. Since X is a leaf's label, X is a terminal symbol of $G(D \cup m)$, and thus $p(X) \notin D \cup m$. Therefore $p(X) \in LP_A \setminus m$.

Now consider the disclosure trees under $D \cup m \cup \{p(X)\}$. Since $X \in UL$, the disclosure of $p(X)$ only creates or modifies or removes disclosure trees that contain elements of UL. Thus, every disclosure tree in $view^{\mathrm{R}}(D \cup m \cup \{p(X)\})$ with a leaf labeled with a credential of Bob should also contain a node labeled with an element of UL. We can repeat the same discussion by substituting $m \cup \{p(X)\}$ for m as long as there exists a disclosure tree with a leaf labeled with a credential of Bob. However, since LP_A is finite, this repetition eventually ends. At that time there must be no disclosure tree with a leaf labeled with a credential of Bob. Therefore for some m' such that $m \subseteq m' \subseteq m \cup LP_A$, $view^{\mathrm{R}}(D \cup m')$ is not evolvable for Bob. □

Yoshiaki Takata received the Ph.D. degree in information and computer science from Osaka University in 1997. He was with Nara Institute of Science and Technology as an Assistant Professor in 1997–2007. In 2007, he joined the faculty of Kochi University of Technology. His current research interests include formal specification and verification of software systems.

Hiroyuki Seki received the Ph.D. degree in information and computer science from Osaka University in 1987. He was with Osaka University as an Assistant Professor in 1990–1992 and an Associate Professor in 1992–1994. In 1994, he joined the faculty of Nara Institute of Science and Technology, where he has been a Professor since 1996. His current research interests include formal language theory and formal approach to software development.