# Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation

TING YU and MARIANNE WINSLETT
University of Illinois at Urbana-Champaign
and
KENT E. SEAMONS
Brigham Young University

Business and military partners, companies and their customers, and other closely cooperating parties may have a compelling need to conduct sensitive interactions on line, such as accessing each other's local services and other local resources. Automated trust negotiation is an approach to establishing trust between parties so that such interactions can take place, through the use of access control policies that specify what combinations of digital credentials a stranger must disclose to gain access to a local resource. A party can use many different strategies to negotiate trust, offering tradeoffs between the length of the negotiation, the amount of extraneous information disclosed, and the computational effort expended. To preserve parties' autonomy, each party should ideally be able to choose its negotiation strategy independently, while still being guaranteed that negotiations will succeed whenever possible, i.e., that the two parties' strategies will interoperate. In this paper we provide the formal underpinnings for that goal, by formalizing the concepts of negotiation protocols, strategies, and interoperation. We show how to model the information flow of a negotiation, for use in analyzing strategy interoperation. We also present two large sets of strategies whose members all interoperate with one another, and show that these sets contain many practical strategies. We develop the theory both for black-box propositional credentials and credentials with internal structure, and for access control policies whose contents are (resp. are not) sensitive. We also discuss how these results fit into TrustBuilder, our prototype system for trust negotiation.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Algorithm, Security

Additional Key Words and Phrases: Automated Trust Negotiation, Interoperable Strategies

Name: Ting Yu and Marianne Winslett
Address: Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W.Springfield Ave., Urbana, IL 61802; email: {tingyu, winslett}@cs.uiuc.edu
Name: Kent E. Seamons
Address: Computer Science Department, Brigham Young University, 3361 TMCB, Provo, UT 84602; email: seamons@cs.byu.edu

## 1. INTRODUCTION

With billions of users on the Internet, most interactions will occur between strangers, i.e., entities that have no pre-existing relationship and may not share a common security domain. Before strangers can conduct sensitive interactions, a sufficient level of mutual trust must be established. For this purpose, the *identity* of the participants (e.g., their social security number, fingerprint, institutional tax ID) will often be irrelevant to determining whether or not they should be trusted. Instead, the *properties* of the participants, e.g., employment status, citizenship, group membership, will be most relevant. Traditional security approaches based on identity require a stranger to obtain an identity within the local security domain, such as a local login, capability, or credential, before being allowed to access local resources; but the same problem arises when the stranger needs to prove on-line that she is eligible to obtain a particular kind of identity within the domain. Automated trust establishment offers a solution to this problem.

With automated trust establishment, strangers establish trust by exchanging *digital credentials*, the on-line analogues of paper credentials that people carry in their wallets: digitally signed assertions by a credential issuer about the credential owner. A digital credential is a statement signed using the issuer's private key, and can be verified using the issuer's public key. Typically, a digital credential contains the issuer's assertion that a particular entity possesses certain properties. Often, the assertion is expressed using attribute name/value pairs. Often, the public keys of the entities described in the credential are included in the credential, so that the entities can use their corresponding private keys to answer challenges or otherwise prove that the credential refers to them; other approaches are also possible [Li et al. 2000]. Digital credentials can be implemented using, for example, X.509 [IETF 2002] certificates.

While some resources are freely accessible to all, many require protection from unauthorized access. Access control policies can be used to protect many kinds of resources, such as services accessed through URLs, roles in role-based access control systems, and capabilities in capability-based systems. Since digital credentials themselves can contain sensitive information, their disclosure can also be governed by access control policies.

> Suppose that Alice is a landscape designer who wishes to order plants from Champaign Prairie Nursery (CPN). She fills out an order form on the web, checking an order form box to indicate that she wishes to be exempt from sales tax. Upon receipt of the order, CPN will want to see a valid credit card or Alice's account credential issued by CPN, and a current reseller's license. Alice has no account with CPN, but she does have a digital credit card. She is willing to show her reseller's license to anyone, but she will only show her credit card to members of the Better Business Bureau.

In our approach to automated trust establishment, trust is established incrementally by exchanging credentials and requests for credentials, an iterative process known as *trust negotiation*. While a *trust negotiation protocol* defines the ordering of messages and the type of information messages will contain, a *trust negotiation strategy* controls the exact content of the messages, i.e., which credentials to

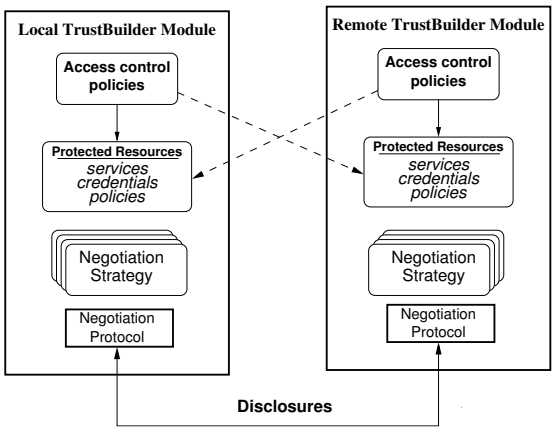



Fig. 1. The TrustBuilder architecture for automated trust negotiation.

disclose, when to disclose them, and when to terminate a negotiation. Figure 1 introduces our *TrustBuilder* architecture for trust negotiation. Each participant in the negotiation has an associated security agent that manages the negotiation. The security agent mediates access to local protected resources, i.e., services, access control policies, and credentials. We say a credential or access control policy is *disclosed* if it has been sent to the other party in the negotiation, and that a service is disclosed if the other party is given access to it. Disclosure of protected resources is governed by access control policies, which specify what credentials the other party needs to disclose in order to gain access to local resources (as indicated by the dotted lines in figure 1. During a negotiation, the security agent uses a local negotiation strategy to determine what local resources to disclose next, and to accept new disclosures from the other party.

Figure 1 shows a single protocol for establishing trust, and assumes there will be a variety of negotiation strategies that must be supported. All trust negotiation strategies share the goal of building trust through the disclosure of digital credentials, culminating in obtaining access to a protected resource. Once enough trust has been established that a particular credential can be disclosed to the other party, a local negotiation strategy must determine whether the credential is relevant to the current stage of the negotiation. Different negotiation strategies will use different definitions of relevance, involving tradeoffs between computational cost, the length of the negotiation, and the number of disclosures. Because the Internet is a freewilling place where autonomy is considered important, strangers will often differ in their strategic choices. Thus we need to be able to allow strangers considerable independence in their choice of negotiation strategy, while still providing guarantees that the chosen strategies will interact (interoperate) correctly during negotiation.

To address these issues, in this paper we characterize a broad class of strategies and design a strategy-independent, language-independent trust negotiation protocol that ensures the interoperability of these strategies within the TrustBuilder trust negotiation architecture. We do this first for propositional credentials and

unprotected access control policies (sections 3 - 6) and then for credentials with internal structure, a much more technically challenging but also more realistic scenario (sections 7 - 9). We present the propositional case separately for pedagogical reasons: structured credentials and sensitive policies add a great deal of complexity to the underlying theory.

The work reported in this paper falls within the broad scope of the TrustBuilder project, which is exploring the concept of ubiquitous trust negotiation. The TrustBuilder project includes both theoretical and practical components. On the theory side, our goal is to establish a sound theoretical underpinning for the practical side's testbed prototypes, which test scalability and reusability of trust negotiation middleware in different contexts. In this type of work, we have found that the theory must precede and guide the implementation. Thus some of the results reported in this paper have already been implemented in one of the TrustBuilder prototypes, and some have not. We will return to this topic in section 10.

We are sometimes asked whether the heavy theoretical artillery in this paper is necessary — won't typical trust negotiations be short and simple? We certainly hope so, and the short and simple case is what the TrustBuilder prototypes are optimized for. However, we believe that the theory must address the general case. As soon as we set a hard limit on trust negotiation — e.g., no more than $n$ rounds of messages — someone will design an application that requires $n + 1$ rounds of messages.

We are also asked whether trust negotiation has a future, given the lack of popularity of the Public Key Infrastructure. We believe that public/private keys will become popular once there are compelling reasons for subjects to have them. For example, if trust negotiation allows labor-intensive sensitive business processes, such as registering parties for an on-line auction, to be automated, then trust negotiation will be adopted in a grass-roots manner.

In this paper we have omitted discussion of a number of issues that we find particularly interesting, but lack of room to explore. The first of these is ways to protect against attacks against the trust negotiation process, beyond the secure communication already provided in TrustBuilder prototypes. The second is how to handle access control policy predicates that have a non-static interpretation, such as time-based predicates [Bonatti and Samarati 2000], checks for revocation, authentication of ownership, references to the local computing environment, requirements that two principals act together jointly, etc. These useful additional kinds of predicates require straightforward extensions to the semantics we are using. We also do not discuss the process of obtaining credentials; in this paper the party who discloses a credential has full responsibility for obtaining it and caching it locally. This paper also assumes that all the trust negotiation software is trusted; in practice, this assumption can be relaxed under certain conditions. Finally, we only consider strategies that treat access control policies as black boxes. More specifically, we do not consider strategies that reason about whether one set of policies is entailed by another set of policies. For example, suppose the policies for two of Alice's credentials $C_1$ and $C_2$ are $x.age \geq 18$ and $x.age \geq 21$ respectively. If Bob indicates that he has no credentials that can satisfy $C_1$'s policy, then, theoretically Alice can realize that Bob cannot satisfy $C_2$'s policy either, and avoid disclosing it. While

theoretically interesting, such strategies would be extremely expensive in practice.

## 2. RELATED WORK

Credential-based authentication and authorization systems fall into three groups: identity-based, property-based, and capability-based. Originally, public key certificates, such as X.509 [IETF 2002] and PGP [Zimmerman 1994], simply bound keys to names, and X.509 v.3 certificates later extended this binding to general properties (attributes). Such certificates form the foundation of identity-based systems, which authenticate an entity's identity or name and use it as the basis for authorization. Identity is not a useful basis for our aim of establishing trust among strangers.

Systems have emerged that use property-based credentials to manage trust in decentralized, distributed systems [Herzberg et al. 2000; Johnson et al. 1998; Winsborough et al. 2000]. Johnson et al. [1998] use attribute certificates (property-based credentials) and use-condition certificates (policy assertions) for access control. Use-condition certificates enable multiple, distributed stakeholders to share control over access to resources. In their architecture, the access control policy evaluation engine retrieves the certificates associated with a user to determine if the use conditions are met. Their work could use our approach to protect sensitive certificates.

The Trust Establishment Project at the IBM Haifa Research Laboratory [Herzberg et al. 2000] has developed a system for establishing trust between strangers according to policies that specify constraints on the contents of public-key certificates. Servers can use a collector to gather supporting credentials from issuer sites. Each credential contains a reference to the site associated with the issuer. That site serves as the starting point for a collector-controlled search for relevant supporting credentials. Security agents in our work could adopt the collector feature, and we could use their policy definition language. Their work could use our approach to protect sensitive credentials and gradually establish trust.

Capability-based systems manage delegation of authority for a particular application. Capability-based systems are not designed for establishing trust between strangers, since clients are assumed to possess credentials that represent authorization of specific actions with the application server. In the capability-based KeyNote system of Blaze et al. [1999; 1998], a credential describes the conditions under which one principal authorizes actions requested by other principals. KeyNote policies delegate authority on behalf of the associated application to otherwise untrusted parties. KeyNote credentials express delegation in terms of actions that are relevant to a given application. KeyNote policies do not interpret the meaning of credentials for the application. This is unlike policies designed for use with property-based credentials, which typically derive roles from credential attributes. The Simple Public Key Infrastructure [IETF 2001] uses a similar approach to that of KeyNote by embedding authorization directly in certificates.

Bonatti and Samarati [2000] introduced a uniform framework and model to regulate service access and information release over the Internet. Their framework is composed of a language with formal semantics and a policy filtering mechanism. Their language can be easily extended to be used in automated trust negotiation. Further, the policy filtering mechanism introduced in their work can be integrated

with our work to provide better protection of sensitive access control policies. For example, service renamings in their policy filtering mechanism are used in this paper to distinguish between local and remote credentials and to hide credential semantics.

The P3P standard [W3C 2002] focuses on negotiating the disclosure of a user's sensitive private information based on the privacy practices of the server. Trust negotiation is generalized to base disclosure on any server property of interest to the client that can be represented in a credential. The work on trust negotiation focuses on certified properties of the credential holder while P3P is based on data submitted by the client that are claims the client makes about itself. Support for both kinds of information in trust negotiation is warranted.

SSL [Frier et al. 1996], the predominant credential-exchange mechanism in use on the web, and its successor TLS [Dierks and Allen 1999; Farrell 1998] support credential disclosure during client and server authentication. TLS is suited for identity-based credentials and requires extension to make it adaptable to property-based credentials [Hess et al. 2002]. Needed additions include protection for sensitive server credentials and a way for the client to explain its policies to the server.

Islam et al. [1997] show how to control downloaded executable content based on role-based access control models. Their system assumes that all the appropriate credentials accompany requests for downloaded content. Their work could be extended using our approach to disclose policies and conduct negotiations.

Li et al. [2001] introduce a role-based trust-management language $RT_0$, which can be easily used to map entities to roles based on the properties described in their credentials. They also present an algorithm to locate and retrieve credentials that are not available locally. This *credential chain discovery* is an important component of trust negotiation, and should be supported by trust negotiation software packages.

Delegations [Li et al. 2000; Blaze et al. 1999] play important roles in distributed trust management systems, and delegation languages will certainly be part of the trust establishment world. However, the two topics seem to be orthogonal in the sense that current delegation policy languages and compliance checkers are not suitable as a basis for trust negotiation. The mismatch seems to stem from the fact that trust negotiation relies heavily on general-purpose credentials, which are often used for a purpose quite different from what the credential issuer had in mind.

The first trust negotiation strategies proposed included a naive strategy that discloses credentials as soon as they are unlocked and discloses no policy information, as well as a strategy that discloses credentials only after each party determines that trust can be established, based on reviewing the other party's policies [Winsborough et al. 2000]. Yu et al. [2000] introduced a strategy that would establish trust whenever possible and had certain efficiency guarantees. Seamons et al. [2001] introduced policy graphs as a way to protect sensitive policy information by establishing trust gradually. The fact that none of these early strategies will interoperate demonstrates the need for trust negotiation protocols and strategy families to support interoperability between negotiation strategies. The current paper extends an earlier work [Yu et al. 2001], which examined interoperable strategies while modeling access control policies as propositional formulas. In this paper, we also consider

credentials with internal structure and policy graphs that protect sensitive policies. These extensions make the negotiation protocol and strategies more suitable for realistic systems.

## 3. ACCESS CONTROL POLICIES

We assume that the information contained in access control policies (*policies*, for short) and credentials can be expressed as finite sets of statements in a language with a formal semantics, so that two strategies can agree on the interpretation of a credential or policy. XML or logic programming languages may be suitable languages in practice [Herzberg et al. 2000; Bonatti and Samarati 2000; Apt et al. 1999], subject to the limits discussed below. For convenience, we will assume that the language semantics allows us to describe the meaning of a set $X$ of statements in the language as the set of all models that satisfy $X$, in the usual logic sense. For purely practical reasons, we require that the language be *monotonic*, i.e., if a set of statements $X$ entails policy $P$, then any superset of $X$ will also entail $P$; that way, once a negotiation strategy has determined that the credentials disclosed by a participant satisfy[1] the policy of a resource, the strategy knows that the same policy will be satisfied for the rest of the negotiation, and does not have to be rechecked.

In the first half of this paper, *resource* refers to a credential or service, and we will treat resources as propositional symbols - black boxes that have an externally visible key (the propositional symbol) but no other externally visible structure. Each resource has exactly one access control policy $P(C_1, \ldots, C_k)$, where $P(C_1, \ldots, C_k)$ is a Boolean expression involving only credentials $C_1, \ldots, C_k$ that the other party may possess, Boolean constants *true* and $false$, the Boolean operators $\vee$ and $\wedge$, and parentheses as needed. $C_i$ is true if and only if the other party has disclosed credential $C_i$. We use $C \leftarrow P(C_1, \ldots, C_k)$ to denote that $C$'s policy is $P(C_1, \ldots, C_k)$. We can distinguish between local and remote resources (by renaming propositional symbols as necessary), so that it is always clear whether a propositional symbol in a policy refers to a local or non-local credential. Resource $C$ is *unlocked* if its access control policy is satisfied by the set of credentials disclosed by the other party. A resource is *unprotected* if its policy is always satisfied. The *denial policy* $C \leftarrow false$ means that either the party does not possess $C$, or else will not disclose $C$ under any circumstances (i.e., $C$'s policy is equivalent to *false*). A party implicitly has a denial policy for each credential it does not possess. If the disclosure of a set $\mathcal{C}$ of credentials satisfies resource $R$'s policy, then we say $\mathcal{C}$ is a *solution set* for $R$. Further, if none of $\mathcal{C}$'s proper subsets is a solution set for $R$, we say $\mathcal{C}$ is a *minimal solution set* for $R$.

Given sequence $\mathcal{Q} = (C_1, \ldots, C_n)$ of disclosures of protected resources, if each $C_i$ is unlocked at the time it is disclosed, $1 \leq i \leq n$, then we say $\mathcal{Q}$ is a *safe disclosure sequence*. The goal of trust negotiation is to find a safe disclosure sequence $\mathcal{Q} = (C_1, \ldots, C_n = R)$, where $R$ is the resource to which access was originally requested. When this happens, we say that trust negotiation succeeds. If $C_i = C_j$ and $1 \leq i < j \leq n$, then we say $\mathcal{Q}$ is *redundant*. Since the language used to represent policies

[1] We use the term *satisfied* in the colloquial English sense in the remainder of the paper: does the stranger satisfy the policy? More precisely, do the stranger's disclosures entail the policy?

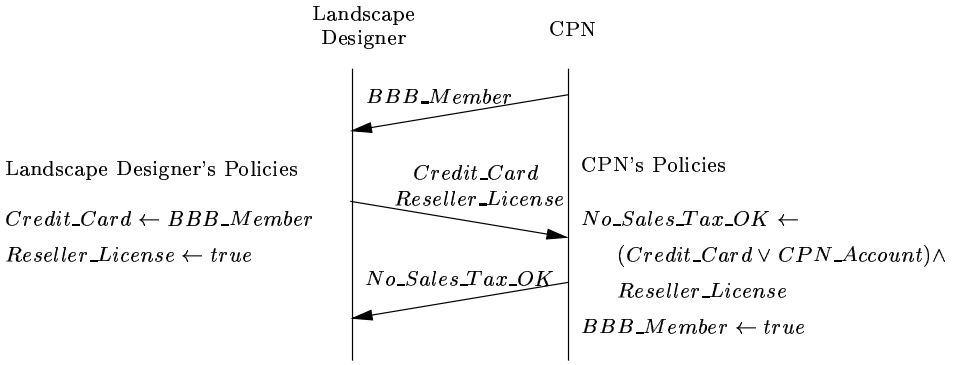



Fig. 2. An example of access control policies and a safe disclosure sequence that establishes trust between the server and the client.

and credentials is monotonic, we can remove the later duplicates from a redundant safe disclosure sequence and the resulting sequence is still safe. Figure 2 shows a safe disclosure sequence for the landscape designer's purchase from CPN discussed earlier. Appendix A contains a more complex example using structured credentials.

It is important to note that this example, and our algorithms that follow, rely on *lower levels of software* to perform the functions associated with disclosure of a credential: verification of its contents, checks for revocation as desired, checks of validity dates, authentication of ownership, etc., as is normally done for X.509 certificates. The necessary calls to these functions, and the translation of credentials into the language used to represent policies, are not shown in our algorithms.

## 4. THE TRUSTBUILDER PROTOCOL AND STRATEGY FAMILIES

Previous work on trust negotiation has not explicitly proposed any trust negotiation protocols, instead defining protocols implicitly by the way each negotiation strategy works. This is one reason why no two different previously proposed strategies can interoperate — their underlying protocols are totally different.

We remedy this problem by defining a simple protocol for TrustBuilder, shown in figure 3. Formally, a *message* in TrustBuilder Protocol 1 is a set $\{R_1, \ldots, R_k\}$ where each $R_i$ is a disclosure of a local credential, a local policy, or a local service. When a message is the empty set $\emptyset$, we also call it a *failure message*. Further, to guarantee the safety and timely termination of trust negotiation no matter what policies and credentials the parties possess, TrustBuilder Protocol 1 requires the negotiation strategies used with it to enforce the following three conditions throughout negotiations:

(1) Every disclosure must be safe.

(2) If a message contains a denial policy disclosure $C \leftarrow false$, then $C$ must appear in a previously disclosed policy. This helps to focus the negotiation and helps prevent infinite negotiations.

(3) A credential or policy can be disclosed at most once. Again, this helps focus the negotiation.

```
TrustBuilder_Agent(L,R)
Input: L is the set of local resources and policies.
       R is the resource to which the client originally requested access.
Output: the result of a negotiation, which can either be FAIL or SUCCEED.
Let M be an empty disclosure message sequence.
r=NOT_TERMINATED.
If (R is a local resource) then //Negotiation is initiated by the other party.
    r=TrustBuilder_send_response(M,L,R).
    If (r==SUCCEED or r==FAIL) then //Negotiations have terminated.
        return r.
While (r==NOT_TERMINATED)
    Receive message m from the other party.
    Add m to the end of M.
    r=TrustBuilder_check_for_termination(m,R).
    If (r==SUCCEED or r==FAIL) then return r.
    r=TrustBuilder_send_response(M,L,R).
    If (r==SUCCEED or r==FAIL) then return r.
End of TrustBuilder_Agent.

TrustBuilder_send_response(M,L,R)
S_m=Local_strategy(M,L,R).
//S_m contains the candidate messages the local strategy suggests.
Choose any single message m' from S_m.
Send m' to the remote party.
Add m' to the end of M.
r=TrustBuilder_check_termination(m',R).
return r.
End of TrustBuilder_send_response.

TrustBuilder_check_for_termination(m, R)
If (m == ∅) then return FAIL. //Negotiations have failed.
If (R ∈ m) then return SUCCEED. //Negotiations have succeeded.
return NOT_TERMINATED.
End of TrustBuilder_check_for_termination.
```

Fig. 3. Pseudocode for TrustBuilder Protocol 1.

Before the negotiation starts, the client sends the original resource request message to the server, indicating its desire to access service $R$.[2] This request triggers the negotiation, and the server invokes its local security agent with the call TrustBuilder_Agent$(L, R)$, where $L$ is the set of the server's credentials and policies. Meanwhile, the client also invokes its security agent. Then the client and server exchange messages until either the service $R$ is disclosed by the server or one party sends a failure message. The whole negotiation process is shown in figure 3.

In the remainder of this paper, unless otherwise noted, we discuss only strategies that can be called from TrustBuilder Protocol 1 and satisfy the three conditions above. A formal definition of a negotiation strategy is given below.

DEFINITION 4.1. *A* strategy *is a function $f$ which takes three parameters $\mathcal{M}$, $L$ and $R$, where $R$ is the resource to which the client originally requested access,*

[2]The architecture in figure 1 is symmetric and peer-to-peer; we use the terms *client* and *server* as a convenient way of distinguishing the two parties.

*$\mathcal{M} = (m_1, \ldots, m_k)$ is a sequence of disclosure messages such that $m_i \neq \emptyset$ and $R \notin m_i$ for $1 \leq i \leq k$, and $L$ is the set of local resources and policies. The output of $f$ is $S_m$, which is a set of disclosure messages. Further, every disclosure in a message in $S_m$ must be of a local resource or policy, as must be all the disclosures in $m_{k-2i}$, for $1 \leq k-2i < k$. The remaining disclosures in $\mathcal{M}$ are of resources and policies belonging to the other party.*

Intuitively, based on the disclosures made so far, plus the local resources and policies, a negotiation strategy will suggest the next set of disclosures to send to the other party. Note that a strategy returns a *set* of possible disclosure messages, rather than a single message. Practical negotiation strategies will suggest a single next message, but the ability to suggest several possible next messages will be very convenient in our formal analysis of strategy properties, so we include it both in the formal definition of a negotiation strategy and also in the TrustBuilder Protocol 1 pseudocode in figure 3.

DEFINITION 4.2. *Strategies $f_A$ and $f_B$ are* compatible *if whenever there exists a safe disclosure sequence for a party Alice to obtain access to a resource owned by party Bob, the trust negotiation will succeed when Alice uses $f_A$ and Bob uses $f_B$. If $f_A = f_B$, then we say that $f_A$ is* self-compatible.

DEFINITION 4.3. *A* strategy family *is a set $\mathcal{F}$ of mutually compatible strategies, i.e., $\forall f_1 \in \mathcal{F}, f_2 \in \mathcal{F}$, $f_1$ and $f_2$ are compatible. We say a set $\mathcal{F}$ of strategies is* closed *if given a strategy $f'$, if $f'$ is compatible with every strategy in $\mathcal{F}$, then $f' \in \mathcal{F}$.*

One obvious advantage of strategy families is that a security agent (SA) can choose strategies based on its needs without worrying about interoperability, as long as it negotiates with other SAs that use strategies from the same family. As another advantage, under certain conditions, an SA does not need to stick to a fixed strategy during the entire negotiation process. It can adopt different strategies from the family in different phases of the negotiation. For example, during the early phase, since the trust between two parties is very limited, an SA may adopt a cautious strategy for disclosing credentials. When a certain level of trust has been established, in order to accelerate the negotiation, the SA may adopt a less cautious strategy. However, without the closure property, a family may not be large enough for practical use. As an extreme example, given any self-compatible strategy $f$, $\{f\}$ is a strategy family. The closure property guarantees the maximality of a strategy family.

TrustBuilder Protocol 1 is so simple and natural that it is natural to wonder whether there are any other interesting protocols. Similarly, is there more than one interesting strategy family?

The answer to both questions is a resounding "yes", especially for the structured credentials discussed in the second half of the paper. To help us choose between potential protocols and families to present here, we have developed a set of (often conflicting) desiderata, presented below.

(1) The protocol should be as simple as possible, with a minimal number of types of messages and minimal restrictions on their sequencing.

(2) The family must include strategies that can be implemented using today's typical trust negotiation environment underpinnings. For example, the reasoning powers provided by the XSB logic programming engine [Sagonas et al. 1994], or the IBM Trust Policy Language (TPL) engine [Herzberg et al. 2000], should be sufficient.

(3) The family must include strategies that will run efficiently in today's typical trust negotiation environments. For example, we do not like families in which the strategy must find *all* minimal solution sets for a policy as soon as negotiation begins.

(4) The family must include as many as possible of what we call "unintelligent" strategies, which tend to be extremely easy to implement but not too clever at determining which policies and credentials are relevant to the negotiation. We think such strategies will be popular, and we give them priority over cleverer strategies that will be harder to implement and often slower at run time.

## 5. CHARACTERIZING SAFE DISCLOSURE SEQUENCES

In this section, we define the concepts that we use to describe the progress of a negotiation and to characterize the behavior of different strategies. We define the trees and tree operations that give meaning to strategies' actions. Negotiation strategies do not need to materialize these trees; rather, the trees provide the formal basis for what a strategy does. This section is the heart of the paper, and readers who survive its onslaught of definitions and examples will understand why some strategies interoperate and others do not. This section also shows how to generate a set of strategies based on the set's member who is the most reluctant to disclose information. The concepts introduced in this section will be used to define a useful strategy family in section 6, and will resurface in revised form when we examine credentials with internal structure.

In the remainder of the paper, we use $R$ to represent the resource to which access was originally requested, unless otherwise noted.

DEFINITION 5.1. *A* disclosure tree *for $R$ is a finite tree satisfying the following conditions:*

*(1) The root is labeled with $R$.*

*(2) Except for the root, each node is labeled with a credential. When the context is clear, we refer to a node by its label. If two nodes are labeled with the same credential, we will explicitly distinguish them when we refer to them.*

*(3) The children of a node $C$ form a minimal solution set for $C$.*

*When all the leaves of a disclosure tree $T$ are unprotected credentials, we say $T$ is a* full *disclosure tree. Given a disclosure tree $T$, if there is a credential appearing twice in the path from a leaf node to the root, then we call $T$ a* redundant *disclosure tree.*

Figure 4 shows example disclosure trees. Note that $T_3$ is redundant and $T_4$ is a full disclosure tree. We use abstract examples because they are much more concise and can be easily manipulated to show all the different cases that the theory must cover.

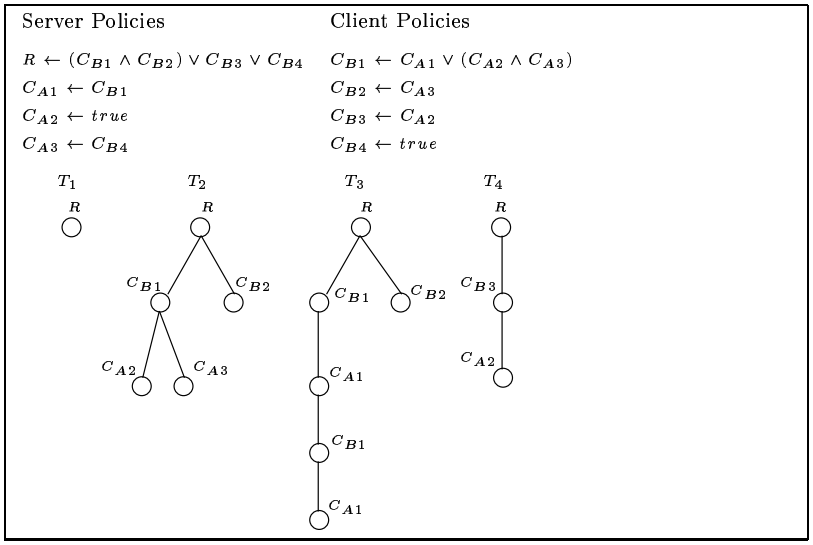



Fig. 4. Example disclosure trees for a set of policies

The following theorems state the relationship between disclosure trees and safe disclosure sequences that lead to the granting of access to resource $R$.

THEOREM 5.1. *Given a non-redundant safe disclosure sequence $\mathcal{Q} = (C_1, \ldots, C_n = R)$, there is a full non-redundant disclosure tree $T$ such that the following both hold:*

*(1) The nodes of $T$ are a subset of $\{C_1, \ldots, C_n\}$.*

*(2) For all credential pairs $(C_1', C_2')$ such that $C_1'$ is an ancestor of $C_2'$ in $T$, $C_2'$ is disclosed before $C_1'$ in $\mathcal{Q}$.*

PROOF. By induction on $n$. When $n = 1$, resource $R$ is unprotected. We can have a disclosure tree with only the root node, and the theorem holds.

Assume when $1 \leq n \leq k$, the theorem holds.

When $n = k + 1$, since access to $R$ is eventually granted, we can find a minimal solution set $\{C_{j_1}, \ldots, C_{j_t}\}$ for $R$ among the credentials in $\mathcal{Q}$. Since $\mathcal{Q}$ is a non-redundant safe disclosure sequence, for each $C_{j_i}$, where $1 \leq i \leq t$, $(C_1, \ldots, C_{j_i})$ is a safe disclosure sequence with length no more than $k$. By the induction hypothesis, there is a non-redundant disclosure tree $T_i$ whose root is $C_{j_i}$ and all the nodes are credentials appearing in $(C_1, \ldots, C_{j_i})$, which is a prefix of $\mathcal{Q}$. And for every credential pair $(C_1', C_2')$ such that $C_1'$ is an ancestor of $C_2'$ in $T_i$, $C_2'$ is disclosed before $C_1'$ in $(C_1, \ldots, C_{j_i})$. By adding $R$ as the root and all the roots of $T_i$, $1 \leq i \leq t$, as children of $R$, we get a full non-redundant disclosure tree that satisfies conditions 1) and 2) in the theorem. Therefore the theorem holds. □

THEOREM 5.2. *Given a full disclosure tree for $R$, there is a non-redundant safe disclosure sequence ending with the disclosure of $R$.*

PROOF. Let $\mathcal{Q} = (C_1, \ldots, C_n = R)$ be the post-order traversal of $T$. According to the definition of full disclosure trees, when a credential is disclosed in $\mathcal{Q}$, either it

is unprotected or one of its minimal solution sets has been disclosed. Therefore its disclosure is safe, and $\mathcal{Q}$ is a safe disclosure sequence. As discussed earlier, we can remove all but the first disclosure of each credential to make $\mathcal{Q}$ non-redundant. □

By theorems 5.1 and 5.2, we get the following corollary immediately.

COROLLARY 5.1. *There is a safe disclosure sequence ending with the disclosure of $R$ if and only if there is a full non-redundant disclosure tree.* □

Without loss of generality, from now on, we consider only non-redundant disclosure sequences.

Since there is a natural mapping between safe disclosure sequences and disclosure trees, during the negotiation, theoretically one could determine whether a potential credential or policy disclosure is helpful by examining all the disclosure trees for $R$. At the beginning of a negotiation, before disclosures begin, the only relevant disclosure tree for the client contains a single node $R$. As the negotiation proceeds, other trees may become relevant. The following definitions help us describe the set of relevant trees.

DEFINITION 5.2. *Given a disclosure tree $T$ and a set $\mathcal{C}$ of credentials, the* reduction *of $T$ by $\mathcal{C}$, reduction($T, \mathcal{C}$), is the disclosure tree $T'$ which is obtained by removing all the subtrees rooted at a node labeled with resource $C \in \mathcal{C}$. Given a set $\mathcal{T}$ of disclosure trees, $reduction(\mathcal{T}, \mathcal{C}) = \{reduction(T, \mathcal{C}) \mid T \in \mathcal{T}\}$.*

If $\mathcal{C}$ is the set of credential disclosures made so far, then reducing $T$ by $\mathcal{C}$ prunes out the part of the negotiation that has already succeeded. Intuitively, if a credential $C$ has been disclosed, then we already have a safe disclosure sequence for $C$. We do not need to disclose additional credentials or policies in order to get a full disclosure tree rooted at $C$. Tree reduction can be viewed as a reformulation of partial evaluation [Bonatti and Samarati 2000] in the context of disclosure trees. An example of a disclosure tree reduction is shown in figure 5(a).

DEFINITION 5.3. *Given a disclosure tree $T$ and a policy set $\mathcal{P}$ containing no denial policies, the* expansion *of $T$ by $\mathcal{P}$, $expansion(T, \mathcal{P})$, is the set of all disclosure trees $T_i$ such that*

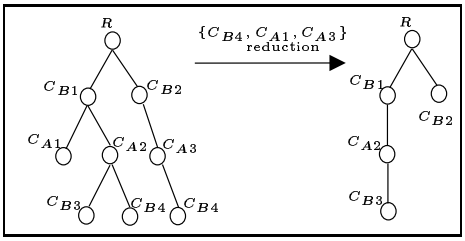
(1) *$T_i$ can be reduced to $T$, i.e., there exists a set $\mathcal{C}$ of credentials such that $reduction(T_i, \mathcal{C}) = T$.*
(2) *For each edge $(C_1, C_2)$ in $T_i$, if $(C_1, C_2)$ is not an edge of $T$, then $C_1$'s policy is in $\mathcal{P}$.*
(3) *For each leaf node $C$ of $T_i$, either $\mathcal{P}$ does not contain $C$'s policy, or $T_i$ is redundant.*

*Given a set of disclosure trees $\mathcal{T}$, $expansion(\mathcal{T}, \mathcal{P}) = \bigcup_{T \in \mathcal{T}} expansion(T, \mathcal{P})$.*

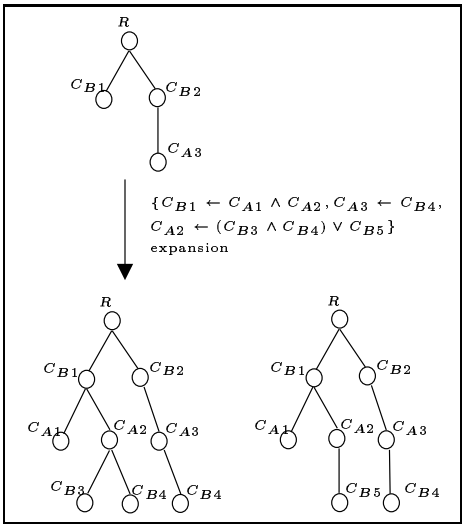
A disclosure tree can expand when a party receives new policy disclosures. An example of a disclosure tree expansion is shown in figure 5(b).

DEFINITION 5.4. *Given a set $\mathcal{T}$ of disclosure trees and a set $\mathcal{P}_d$ of denial policies, the* denial pruning *of $\mathcal{T}$ by $\mathcal{P}_d$, denoted $prune_{denial}(\mathcal{T}, \mathcal{P}_d)$, is the set*

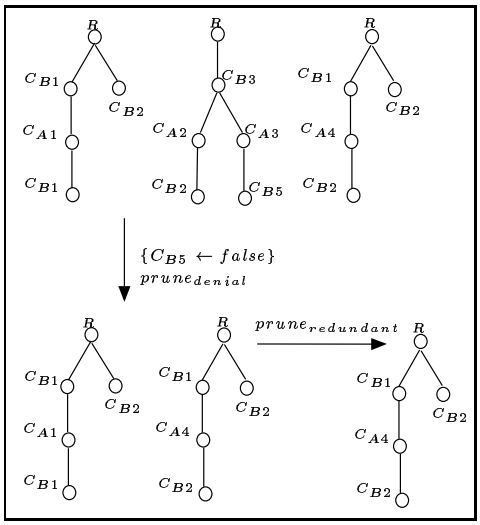
$$\{T \mid T \in \mathcal{T} \text{ and } T \text{ contains no resource whose policy is in } \mathcal{P}_d\}.$$



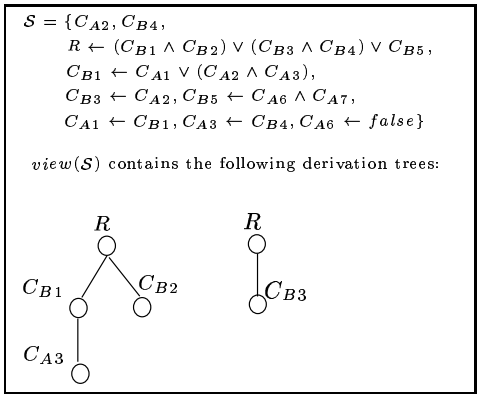

(a) Example of a disclosure tree reduction




(b) Example of a disclosure tree expansion




(c) Example of denial pruning and redundancy pruning




(d) Example of a view

Fig. 5. Examples of operations on disclosure trees

Since a full disclosure tree contains only credentials that the two parties possess, if a disclosure tree node is labeled with a credential with a denial policy, that tree cannot evolve into a full disclosure tree, and is no longer relevant.

DEFINITION 5.5. *Given a set $\mathcal{T}$ of disclosure trees, the* redundancy pruning *of $\mathcal{T}$, denoted $prune_{redundant}(\mathcal{T})$, is the set*

$$\{T \mid T \in \mathcal{T} \text{ and } T \text{ is not a redundant disclosure tree}\}.$$

The rationale for redundancy pruning will be shown after we introduce more

operations on disclosure trees. Examples of denial and redundancy pruning are shown in figure 5(c).

DEFINITION 5.6. *Given a disclosure tree $T$ and a set $\mathcal{P}_d$ of denial policies, a set $\mathcal{P}$ of non-denial policies, and a set $\mathcal{C}$ of credentials, let $\mathcal{S} = \mathcal{P}_d \cup \mathcal{P} \cup \mathcal{C}$. The* evolution *of $T$ by $\mathcal{S}$, denoted evolution$(T, \mathcal{S})$, is*

$$prune_{redundant}(\ prune_{denial}(\ reduction(\ expansion(T, \mathcal{P}), \mathcal{C}), \mathcal{P}_d).$$

*Given a set $\mathcal{T}$ of disclosure trees, evolution$(\mathcal{T}, \mathcal{S}) = \bigcup_{T \in \mathcal{T}} evolution(T, \mathcal{S})$. As a special case, when $T$ is the disclosure tree containing only a root node $R$, then we say evolution$(T, \mathcal{S})$ is the* view *of $\mathcal{S}$, denoted view$(\mathcal{S})$.*

During the negotiation, let $\mathcal{S}$ be the set of credentials and policies disclosed so far and $L$ be the local policies of a negotiation party. Then $view(S \cup L)$ contains all the relevant disclosure trees which can be seen by this party. An example view is shown in figure 5(d).

Sometimes even though a tree may evolve into a full tree later in the negotiation, it is nonetheless redundant and can be removed by redundancy pruning, whose correctness is guaranteed by the following theorem.

THEOREM 5.3. *Let $T$ be a full but redundant disclosure tree. Then there is a full disclosure tree $T'$ that is not redundant.* □

PROOF. Let $\mathcal{Q} = (C_1, \ldots, C_n = R)$ be the post-order traversal of $T$. Obviously $\mathcal{Q}$ is a safe disclosure sequence. According to corollary 5.1, there is a full non-redundant disclosure tree. □

Suppose $\mathcal{S}$ is the set of currently disclosed credentials and policies. By theorem 5.3, if a redundant tree may evolve into a full tree, then the corresponding non-redundant tree is already included in $view(S)$. So the redundant trees are not relevant for the remainder of the negotiation.

In order to make a negotiation successful whenever the policies of the two negotiation parties allow, the negotiation strategy should make sure no possible full disclosure trees have been overlooked. A disclosure tree also tells a party what may contribute to the success of a negotiation. As an example, suppose Bob requests to access Alice's resource $R$. $\mathcal{S}$, the set of disclosures so far, and $view(\mathcal{S})$ are shown in figure 6. Suppose now it is Alice's turn to send a message to Bob. From the disclosure tree, it is clear to an outside observer that credentials $C_{A1}$ and $C_{A2}$ must be disclosed if the negotiation is to succeed. So Alice's negotiation strategy can now disclose $C_{A1}$'s and/or $C_{A2}$'s policy. This example shows that in order to let Alice know what might be the next appropriate message, a disclosure tree should have at least one leaf node that is a credential that Bob wants Alice to disclose. We have the following definition:

DEFINITION 5.7. *Disclosure tree $T$'s* evolvable leaves *for party $A$, denoted as evolvable$(T, A)$, are the set of leaf nodes $C$ of $T$ such that either $C = R$ and $A$ is the server, or $C$ appears in a policy that party $B$ disclosed to $A$. If evolvable$(T, A) \neq \emptyset$, we say $T$ is* evolvable *for $A$.*

The disclosure tree in figure 6 is evolvable for both Alice and Bob. If there is no evolvable tree, then a cautious party will choose to end the negotiation even if

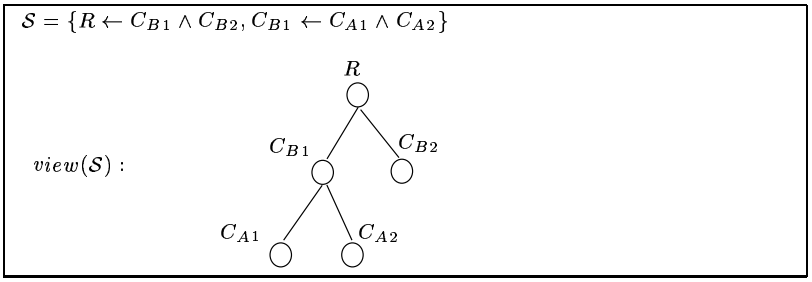



Fig. 6. $view(\mathcal{S})$

the policies of the two parties allow success. Therefore, to ensure that negotiations succeed whenever possible, a strategy must ensure that the other party will have an evolvable tree when the other party needs to make its next disclosure. The only exception is when the strategy knows that no disclosure tree can evolve into a full tree. If a negotiation reaches a point where every leaf node of some disclosure tree is unlocked, then the tree is a full tree and corresponds to a safe disclosure sequence.

If $\mathcal{F}$ is a strategy family, then intuitively, every strategy in $\mathcal{F}$ always discloses enough information to keep the negotiation moving toward success, if success is possible. If $\mathcal{F}$ is also closed, then $\mathcal{F}$ must also contain those strategies that disclose only the minimal amount of information needed to continue negotiations. Therefore it is helpful to formally define a relationship between strategies based on the information they disclose.

DEFINITION 5.8. *Given two negotiation strategies $f_1$ and $f_2$, if for all possible inputs $\mathcal{M}$, $L$, and $R$ to $f_1$ and $f_2$, we have*

$$\forall m \in f_2(\mathcal{M}, L, R)\ \exists m' \in f_1(\mathcal{M}, L, R) \text{ such that } m' \subseteq m$$

*then we say $f_1$ is* at least as cautious as *$f_2$, denoted as $f_1 \preceq f_2$ or $f_2 \succeq f_1$.*

Caution defines a partial order between strategies. Intuitively, if $f_2 \succeq f_1$ then $f_2$ always discloses at least as much information as $f_1$ does.

DEFINITION 5.9. *Given a strategy $f$, the set of strategies* generated *by $f$, denoted $StraSet(f)$, is the set $\mathcal{F} = \{f' | f' \succeq f\}$. $f$ is called the* generator *of $\mathcal{F}$.*

## 6. THE DISCLOSURE TREE STRATEGY FAMILY

In this section, we present the *disclosure tree strategy* (DTS), and prove that DTS generates a closed family. DTS can be thought of as a strategy that is willing to reason very carefully about the trees presented in the previous section, in order to avoid making any unnecessary disclosures. Though DTS is mainly of theoretical interest, the family DTS generates contains many highly practical strategies. We will give two example practical strategies belonging to the family DTS generates.

Throughout this section, we assume that $\mathcal{M} = (m_1, \ldots, m_k)$ is a sequence of messages such that $m_i \neq \emptyset$ and $R \notin m_i$ for $1 \leq i \leq k$. We assume $L_A$ and $L_B$ are the local policies of parties Alice and Bob, respectively, and $\mathcal{S}_\mathcal{M} = \bigcup_{1 \leq i \leq k} m_i$. Without loss of generality, we assume Alice will send the next message to Bob.

DEFINITION 6.1. *The* Disclosure Tree Strategy *(DTS for short) is a strategy* $DTS(\mathcal{M}, L_A, R)$ *such that:*

*(1)* $DTS(\mathcal{M}, L_A, R) = \{\emptyset\}$ *if and only if* $view(\mathcal{S}_\mathcal{M} \cup L_A) = \emptyset$ *or* $view(\mathcal{S}_\mathcal{M})$ *has no evolvable tree for Alice.*

*(2) Otherwise,* $DTS(\mathcal{M}, L_A, R)$ *contains all messages* $m'$ *such that one of the following conditions holds:*

—$m' = \{R\}$*, if* $R$ *is unlocked by credentials in* $\mathcal{S}_\mathcal{M}$*;*

—$m'$ *is a non-empty set of credentials and policies such that* $view(\mathcal{S}_\mathcal{M} \cup m')$ *contains at least one evolvable tree for Bob, and no non-empty proper subset of* $m'$ *has this property.*

Condition 1 states under what circumstances the DTS strategy will terminate the negotiation with a failure message. Condition 2 guarantees that the other party will have an evolvable tree. Therefore, the other party can always send a message back that evolves a disclosure tree. Thus, no failure message will be sent unless there is no disclosure tree at all, in which case the negotiation cannot succeed anyway. Formally, we have the following theorems:

THEOREM 6.1. *The set of strategies generated by DTS is a family.*

PROOF. Suppose Alice and Bob adopt strategies $f_1$ and $f_2$ respectively, and $f_1$ and $f_2$ belong to $StraSet(DTS)$. We need to prove that if the negotiation fails, there is no safe disclosure sequence leading to the granting of access to $R$.

When the negotiation fails, without loss of generality, we assume it is Alice who sends the failure message. Suppose that before Alice sends the failure message, $\mathcal{M} = (m_1, \ldots, m_k)$ is the sequence of messages. Since $\emptyset \in f_1(\mathcal{M}, L_A, R)$ and $f_1 \succeq DTS$, we must have $DTS(\mathcal{M}, L_A, R) = \{\emptyset\}$. According to definition 6.1, one of the following must be true:

(1) $view(\mathcal{S}_\mathcal{M} \cup L_A) = \emptyset$.

(2) $view(\mathcal{S}_\mathcal{M})$ has no evolvable tree for Alice.

When only 2) holds , we must have $DTS((m_1, \ldots, m_{k-1}), L_B, R) = \{\emptyset\}$. Otherwise, since $f_2 \succeq DTS$ and $R \notin m_k$, $m_k$ must be a superset of a minimal set $m'$ of credentials and policies such that $view((\mathcal{S}_\mathcal{M} - m_k) \cup m')$ has at least one evolvable tree for Alice. Therefore, $view(\mathcal{S}_\mathcal{M})$ also has at least one evolvable tree for Alice, which contradicts 2). Since $DTS((m_1, \ldots, m_{k-1}), L_B, R) = \{\emptyset\}$, that means that before Bob sent $m_k$, one of 1) or 2) was true. At the beginning of the negotiation, 2) is not satisfied. Therefore in some previous stage of the negotiation, 1) must have been true.

When 1) holds, that means no tree will evolve to be a full disclosure tree. So there is no safe disclosure sequence leading to the granting of access to $R$. □

THEOREM 6.2. *If a strategy* $f$ *and* $DTS$ *are compatible, then* $f \succeq DTS$.

PROOF. Suppose that Alice is the local party. If $DTS \not\preceq f$, then there exists a choice of $\mathcal{M}$, $L_A$ and $R$ such that

$$\exists m' \in f(\mathcal{M}, L_A, R) \text{ such that } \forall m \in DTS(\mathcal{M}, L_A, R), m \not\subseteq m'$$

Then both of the following must be true:

—$view(\mathcal{S}_{\mathcal{M}} \cup L_A) \neq \emptyset$.
—$view(\mathcal{S}_{\mathcal{M}})$ has at least one evolvable tree for Alice.

Otherwise, according to definition 6.1, $DTS(\mathcal{M}, L_A, R) = \{\emptyset\}$ and $\emptyset \subseteq m'$. Also, we have $R \notin m'$. Otherwise, $R$ is unlocked by $\mathcal{S}_{\mathcal{M}}$ and $\{R\} \in DTS(\mathcal{M}, L_A, R)$.

$DTS(\mathcal{M}, L_A, R)$ contains all the minimal sets $m$ of local policies and credentials such that $view(\mathcal{S}_{\mathcal{M}} \cup m)$ has at least one evolvable tree for Bob. Since $DTS \not\preceq f$, $view(\mathcal{S}_{\mathcal{M}} \cup m')$ has no evolvable tree for Bob. Thus, after sending $m'$, DTS at Bob will send a failure message and end the negotiation. However, since $view(\mathcal{S}_{\mathcal{M}} \cup L_A) \neq \emptyset$, there is a tree in $view(\mathcal{S}_{\mathcal{M}} \cup L_A)$ whose leaves $\{C_1, \ldots, C_t\}$ are all evolvable for Bob and none of those credentials' policies is in $\mathcal{S}_{\mathcal{M}}$. So if all those credentials are unprotected, then there exists a full disclosure tree, which means that $f$ and DTS are not compatible, leading to a contradiction. □

We call the family generated by DTS the *DTS family*. By theorems 6.1 and 6.2, we get the following corollary immediately.

COROLLARY 6.1. *The DTS family is closed.* □

As we mentioned in section 4, one advantage of a strategy family can be the ability to adopt different strategies from a family in different phases of the negotiation. Correct interoperability is guaranteed as long as both parties' strategies are from the same family.

DEFINITION 6.2. *Let $f_1$ and $f_2$ be two strategies. A strategy $f'$ is a* hybrid *of $f_1$ and $f_2$ if for all choices of $\mathcal{M}$, $L$ and $R$, we have $f'(\mathcal{M}, L, R) \subseteq f_1(\mathcal{M}, L, R) \cup f_2(\mathcal{M}, L, R)$ and $f' \neq f_1$ and $f' \neq f_2$.*

If a security agent adopts different DTS family strategies in different phases of trust negotiation, it is equivalent to adopting a hybrid of those strategies.

THEOREM 6.3. *Let $f_1$ and $f_2$ be strategies in the DTS family and let $f'$ be a hybrid of $f_1$ and $f_2$. Then $f'$ is also in the DTS family.*

PROOF. $\forall \mathcal{M}, L, R$, let $f'(\mathcal{M}, L, R) = \{m_1, \ldots, m_k\}$. Since $f'$ is a hybrid of $f_1$ and $f_2$, $m_i \in f_1(\mathcal{M}, L, R)$ or $m_i \in f_2(\mathcal{M}, L, R)$, for all $1 \leq i \leq k$. Because both $f_1$ and $f_2$ are in the DTS family, there exists $m' \in DTS(G, L, R)$ such that $m' \subseteq m_i$. Thus, $f' \succeq DTS$. □

Therefore, as long as both parties use strategies from the DTS family, they can switch between different practical strategies as often as they like, and trust negotiation will still succeed whenever possible.

Although disclosure trees are a useful tool for understanding strategy properties, it would require exponential time and space to materialize all the disclosure trees during a negotiation. Fortunately, many strategies in the DTS family are quite efficient, such as TrustBuilder1-Simple and TrustBuilder1-Relevant in figure 7. Both strategies take parameters $\mathcal{M} = (m_1, \ldots, m_k)$, the sequence of messages so far, $L$, the set of local resources and policies and $R$, the resource to which access was originally requested. The output is a set containing a single message.

The TrustBuilder1-Simple strategy (Figure 7(a)) puts all undisclosed policies and unlocked credentials in the next message to the other party. If all the policies and unlocked credentials have already been disclosed, it will send a failure message.

**The TrustBuilder1-Simple Strategy**
$\mathcal{S}_{\mathcal{M}} = \bigcup_{1 \le i \le k} m_i$.
$m = \emptyset$.
For every local credential $C$ that is unlocked by $\mathcal{S}_{\mathcal{M}}$
    $m = m \cup \{C\}$.
For every local locked credential $C$
    if ($C$'s policy $P$ is not a denial policy)
        then $m = m \cup \{P\}$.
For every policy $P' \in \mathcal{S}_{\mathcal{M}}$ such that $P' \notin L$
    For every credential that $C$ appears in $P'$ and has a denial policy
        $m = m \cup \{C \leftarrow false\}$.
$m = m - \mathcal{S}_{\mathcal{M}}$.
return $\{m\}$.

(a)

**The TrustBuilder1-Relevant Strategy**
$\mathcal{S}_{\mathcal{M}} = \bigcup_{1 \le i \le k} m_i$.
$m = \emptyset$.
For every local credential $C$ syntactically relevant to $R$
    if ($C$ is unlocked by $\mathcal{S}_{\mathcal{M}}$)
        then $m = m \cup \{C\}$.
    else $m = m \cup \{C'$*s policy*$\}$.
For every policy $P' \in \mathcal{S}_{\mathcal{M}}$ such that $P' \notin L$
    For every credential $C$ that appears in $P'$ and has a denial policy
        $m = m \cup \{C \leftarrow false\}$.
$m = m - \mathcal{S}_{\mathcal{M}}$.
return $\{m\}$.

(b)

Fig. 7. Pseudocode for two strategies in the DTS family

We say a credential $C$ is *syntactically relevant* to resource $R$ iff $C$ appears in $R$'s policy, or $C$ appears in the policy of a credential $C'$ that is relevant to $R$. The TrustBuilder1-Relevant strategy (figure 7(b)) discloses a credential $C$'s policy only if $C$ is syntactically relevant to $R$. Similarly, TrustBuilder1-Relevant only discloses syntactically relevant unlocked credentials.

In the CPN example in section 1, if Alice adopts the TrustBuilder1-Simple strategy, then when she receives CPN's policy for exemption from sales tax, she will disclose all her unlocked credentials and her policies for her locked credentials, whether or not they are relevant. For example, Alice may disclose her library card, and she may tell CPN that in order to see her patient ID, CPN should present an employee certificate issued by Busey Hospital. On the other hand, suppose CPN adopts the TrustBuilder1-Relevant strategy. Then from all the disclosures Alice made, CPN will identify that only her credit card and her reseller's license are relevant. Thus, CPN will only focus its efforts on satisfying Alice's policy for access to her credit card, instead of trying to gain access to Alice's patient ID.

LEMMA 6.1. *If a credential $C$ appears in a disclosure tree for $R$, then $C$ is relevant to $R$.* □

THEOREM 6.4. *TrustBuilder1-Simple and TrustBuilder1-Relevant belong to the DTS family.*

PROOF. Because TrustBuilder1-Simple $\succeq$ TrustBuilder1-Relevant, it suffices to show that DTS $\preceq$ TrustBuilder1-Relevant. Suppose TrustBuilder1-Relevant$(\mathcal{M}, L, R) = \{m\}$. Let $m'$ be a message in $DTS(\mathcal{M}, L_A, R)$. If a credential $C$ or its policy is disclosed in $m'$, then $C$ must appear in a disclosure tree. According to lemma 6.1, $m' \subseteq m$. Thus DTS $\preceq$ TrustBuilder1-Relevant. □

## 7. CREDENTIALS WITH INTERNAL STRUCTURE, AND SENSITIVE POLICIES

The assumptions that policies do not contain sensitive information and that credentials can be modeled as propositional symbols make it easier to characterize the necessary properties of interoperable strategies. However, the simplifications usually do not hold in practice. In the rest of the paper, we extend our discussion of interoperable strategies to models with more expressive power. Appendix A presents a detailed example that readers should refer to throughout the remainder of the paper.

Realistic access control policies tend to contain sensitive information, because the details of Alice's policy $P$ for disclosure of credential $C$ tend to give hints about $C$'s contents. For example, if there is any information in $C$ that is so sensitive that Alice chooses to control its disclosure very tightly, then it is possible to guess at the nature of that information by looking at $P$: is her parole officer allowed to see $C$? Her welfare case worker? The local HIV, cancer, or mental health clinic? In a corporate setting, a web page accessible to all Microsoft employees plus the employees of a certain IBM department strongly suggests a secret project between the two parties. An Enron employees' stock sale web page whose policy spelled out exactly who could sell stock and when might have raised many eyebrows. More generally, a company's internal and external policies are part of its corporate assets, and it will not wish to indiscriminately broadcast its policies in their entirety.

To protect sensitive policies, Seamons et al. [2001] introduced the concept of policy graphs. The essential idea is that the policy for a resource can be disclosed gradually.

DEFINITION 7.1. *A policy graph for a protected resource $R$ is a finite directed acyclic graph with a single source node and a single sink node. The sink node represents the protected resource $R$. All the other nodes in the graph represent policies. If $R$ is unprotected, then its policy graph has only one node representing $R$.*

Intuitively, a directed edge from node $A$ to node $B$ in a policy graph means that Alice must present credentials that satisfy the policy represented by node $A$ before she can gain access to Bob's resource represented by node $B$. Policy graphs are a generalization of the prerequisite/requisite rules introduced by Bonatti and Samarati [2000], which can be considered as policy graphs with two levels.

Policy graphs as defined here address many privacy problems associated with sensitive policies, but they are not a cure-all. Our current research is looking at ways to more fully decouple disclosure and satisfaction of a policy. This will allow, for example, Bob to gain access to a resource of Alice's that is so sensitive that its policies are *never* disclosed. In the meantime, policy graphs are sufficient to address the issues raised by the examples given earlier, by, e.g., requiring Bob to present an employee ID from an unspecified company (IBM, Microsoft, the local health clinic,

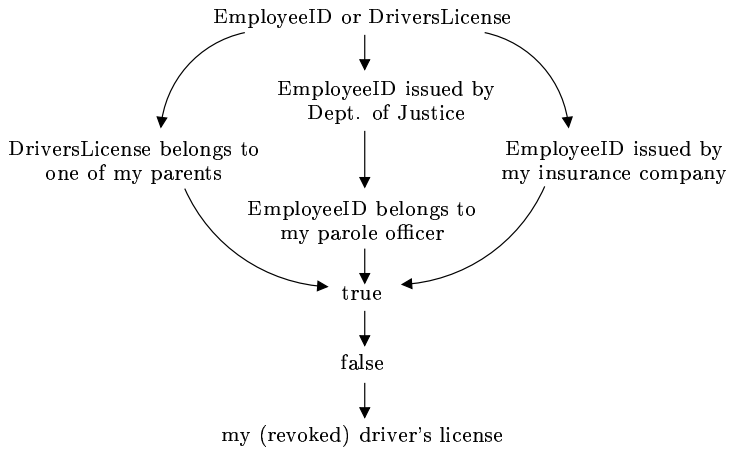



Fig. 8. Example policy graph

etc.) as an initial screening step. If Bob cannot present the "right" employee ID (as defined by a child node in the policy graph), in theory Bob never needs to see the policy node that Bob fails to satisfy (the check for a specific employer in the employee credential).

For example, suppose that Alice has had her driver's license revoked for driving under the influence of alcohol. She may be very sensitive about the fact that she does not possess a driver's license, which, in her home state of California, is abnormal for anyone over the age of 15 and strongly suggests that she has a seriously blemished driving record. She can define a policy subgraph $G$ that describes who she trusts enough to disclose this fact to, then add a new sink to this subgraph, containing the policy "false" (see figure 8), meaning that no one will get to see her license. In the picture, the sink node represents the empty credential, a legal credential under our definition of a credential. Once enough trust has been established ($G$), Alice can disclose the policy "false", which at that point is tantamount to admitting that she does not have a license. If $G$ is the empty graph, Bob may interpret the policy "false" as meaning that Alice does not disclose her driver's license to anyone.

As the above example illustrates, policy graphs offer the possibility that Bob will make many disclosures and receive little or no reward from Alice, because the upper reaches of her policy graphs may contain policies he has no hope of satisfying. If we outlaw unsatisfiable policies, the same effect can be obtained by requiring obscure credentials. One way to address this issue is by allowing Bob's policies to include a request for a credential from Alice that states that her set of policy graphs has been audited and received a seal of approval from one of a specified set of acceptable auditing firms. For example, Bob might require such credentials if he believes that he is a likely target of a denial of service attack. As another approach, Alice might also choose to disclose information about the depth or structure of her policy graphs.

In the remainder of the paper, we model a credential as a set of attribute-value

pairs $\{C.a_1 = v_1, \ldots, C.a_k = v_k\}$, where $a_i$ is an attribute name and $v_i$ is an atomic uninterpreted value for that attribute, for $1 \leq i \leq k$. $C$ is a locally-assigned name for the credential, such as a random number. Credential names should not give away any information about credential contents, and may be changed between negotiations. For convenience, we will assume that Alice can tell which credential names refer to her own credentials, and which refer to Bob's, by renaming if necessary. Similarly, policies and other resources must have local names so that Alice and Bob can discuss them. Each policy node is associated with the local name of the resource whose policy graph it belongs to. Therefore, in the rest of the paper, when given a policy node $P$, we assume we can tell which resource it is related to.

In the rest of the paper we adopt a simple policy language in our examples. The semantics of the language is straightforward to readers with a basic background in mathematical logic. Essentially, the language allows conjunctions and disjunctions of restrictions on credential attribute values. Credentials are represented by variables, not by credential names. For example, Alice can require that Bob present a student ID issued by the University of Illinois showing that he is a full time student:

$$x.issuer = \text{“15697206”} \wedge x.type = \text{“Student ID”} \wedge x.status = \text{“Full Time”} \wedge requesterAuthenticatesTo\ (x.owner)$$

where 15697206 is the public key of the University of Illinois. (As mentioned before, the handling of the *requesterAuthenticatesTo* predicate is not part of this paper, but we include it here for completeness.) Once we devise interoperable strategies for these kinds of credentials and policies, further extensions can allow, for example, credentials to have a more complex internal structure, or policies to have a different syntactic style (e.g., an XML-based language).

For the examples of sensitive policies given earlier to work, we must have a means of referring to the same credential in different policy graph nodes. This allows a policy writer to gradually enforce sensitive constraints on credentials without showing all constraints to the other party at the beginning of the negotiation.

DEFINITION 7.2. *Given a set $\mathcal{C}$ of credentials $\{C_1, \ldots, C_m\}$ and a finite set $\mathcal{P}$ of policies with free variables $x_1, \ldots, x_n$, if there is an assignment from $\mathcal{C}$ to $x_1, \ldots, x_n$ that satisfies all members of $\mathcal{P}$, then we say $\mathcal{C}$ is a* solution set *for $\mathcal{P}$ and satisfies $\mathcal{P}$. If no proper subset of $\mathcal{C}$ is a solution set for $\mathcal{P}$, then we say $\mathcal{C}$ is a* minimal *solution set for $\mathcal{P}$.*

As an example, consider the formula $P$: $before(x.date\_of\_birth, \text{“12/31/1980”}) \wedge y.state = \text{“New York”} \wedge x \neq y$. If the other negotiation participant presents two credentials $C_1 = \{C_1.type = \text{“Drivers License”}, C_1.date\_of\_birth = \text{“10/12/1978”}, C_1.number = \text{“Y10001234”}, \ldots\}$ and $C_2 = \{C_2.type = \text{“Student ID”}, C_2.number = \text{“z0010079”}, C_2.state = \text{“New York”}, C_2.status = \text{“Full Time”}, \ldots\}$, then $\mathcal{C} = \{C_1, C_2\}$ is a solution set for $P$. Since no proper subset of $\mathcal{C}$ satisfies $P$, $\mathcal{C}$ is a minimal solution set.

DEFINITION 7.3. *Suppose $G$ is the policy graph of a resource $R$. Let $P$ be a node in $G$ and $P_0 \rightarrow \ldots \rightarrow P_k = P$ be a path from the source node $P_0$ to $P$. We call such a path a* policy path *to $P$. Let $\mathcal{C}$ be a set of credentials. If $\mathcal{C}$ satisfies $P_0 \wedge \ldots \wedge P_{k-1}$, then we say $\mathcal{C}$ satisfies that policy path and $P$ is* unlocked *by $\mathcal{C}$. As*

*a special case, the source node in $G$ is always unlocked. If $P$ is a policy node and $P_0 \wedge \ldots \wedge P_{k-1} \wedge P$ is satisfied by $\mathcal{C}$, then we say $P$ is* context satisfied *by $\mathcal{C}$ and $\mathcal{C}$ is a* context solution set *for $P$. If no proper subset of $\mathcal{C}$ is a context solution set for $P$, then $\mathcal{C}$ is a* minimal *context solution set for $P$.*

Intuitively, a policy path to $P$ indicates when $P$ can be safely disclosed. If a policy is sensitive, we add more ancestors to $P$ (thus more constraints) in $G$ to control $P$'s disclosure. The definition of context satisfaction implicitly forces the same free variable in policy nodes along a path to be bound to the same credential, enabling gradual enforcement of sensitive constraints. For example in appendix A, one of the paths to gain access to Susan's *patientID* is (policy 7) $\rightarrow$ (policy 8) $\rightarrow$ (policy 11). To context satisfy this path, $x$ in policy 8 and policy 11 should be bound to the same credential. If Alice presents two employeeID credentials, one issued by Busey Hospital with job title "Adjunct Physician" and the other issued by Carle Clinic with job title "Staff Physician", then she does not context satisfy the above path and should not gain access to the client's *patientID* credential.

Definition 7.3 extends the definition of an unlocked resource in section 3 so that our concept of safe disclosures still holds: during a negotiation, a resource (a credential or a policy) cannot be disclosed unless it is unlocked by credentials disclosed by the other party.

DEFINITION 7.4. *Let $G$ be the policy graph of a resource $R$ and let $\mathcal{C}$ be a set of credentials. The* traversal *realized by $\mathcal{C}$, written traversal$(G, \mathcal{C})$, is a subgraph of $G$ that satisfies the following conditions:*

(*1*) *The subgraph contains all the nodes in $G$ that are unlocked by $\mathcal{C}$.*

(*2*) *If $P_1 \rightarrow P_2$ is an edge in $G$ and $P_1$ is context satisfied by $\mathcal{C}$ (thus $P_2$ is unlocked), then $P_1 \rightarrow P_2$ is an edge in the subgraph.*

We use traversals to represent the part of a policy graph that has been unlocked so far during a trust negotiation.

When credentials have internal structure and no externally obvious key, when Alice receives Bob's disclosed policy $P$ for a credential $C$, she cannot tell what information $P$ is protecting: the name "$C$" is now opaque. Theoretically, $C$ can be bound to any credential-valued variable in any of Alice's local policies. In the absence of hints from Bob about which variable $C$ should be bound to, Alice will have to consider $C$ to be relevant to every one of her unsatisfied local policies. She will not be able to focus her trust negotiation efforts on Bob's truly relevant credentials and policies, and trust negotiation will become prohibitively expensive in the general case.

To address this problem, we introduce the concept of a binding. When Bob discloses $C$'s policy $P$, we require him also to disclose what $C$ is relevant to — to *bind* $C$. There are many possible forms that this binding could take. For example, if some or even all of the attribute values in the credential are not sensitive, then Bob can disclose them. The most natural attribute to disclose will often be the credential type, e.g., $C.type$ = "Drivers License". With the credential type and/or additional attribute values in hand, in practice, Alice will be able to tell which policy $C$ may satisfy.

If Bob does not want to disclose all of $C$'s contents, he could identify a particular variable $x$ in Alice's policies as a binding for $C$. This is not tantamount to revealing that $C$'s contents satisfy all the constraints on $x$: Bob may know already that $C$ does not satisfy the constraints on $x$, or that Alice cannot possibly satisfy his policy for gaining access to $C$ (if indeed he possesses $C$). However, he may not yet trust Alice enough to reveal anything about $C$'s contents, or the fact that he does not possess $C$. The use of such approaches to preserve privacy is a focus of our current research, and we do not discuss it further in this paper.

If Bob finds it too revealing to bind $C$ to a particular variable $x$, he can instead bind $C$ to one or more of Alice's policies — those policies that $C$'s disclosure may help to satisfy. If Bob divulges every such binding of $C$, then Alice can focus her trust negotiation on potentially relevant policies and credentials, reducing its cost. If Bob discloses some bindings that are not actually relevant, either to preserve his privacy or because he is using an unintelligent strategy, then we would like trust negotiation's final outcome to be unaffected, although it may take Alice and Bob longer to reach that final outcome.

In this paper, when Bob discloses $P$, he must disclose a binding of $C$ to one or more of Alice's policies. Bob can disclose bindings of $C$ to additional policies as the negotiation proceeds.

Unfortunately, this approach to bindings breaks many of the subtle knowledge assumptions that allowed DTS family strategies to interoperate. Alice no longer has such a clear picture of what Bob's disclosure trees (if he cared to materialize them) would look like. Although the remainder of the paper superficially looks like a replay of the definitions and theorems of the first half of the paper, in fact it reworks the entire underpinnings of strategy interoperation, because of the decrease in what Alice can deduce about what Bob knows.

## 8. THE TRUSTBUILDER PROTOCOL FOR SENSITIVE POLICIES AND STRUCTURED CREDENTIALS

The introduction of policy graphs and structured credentials adds much complexity to automated trust negotiation. In our exploration of strategy families for structured credentials, we have not found a family that satisfies all the desiderata outlined in section 4. In the rest of this paper, we present a family that requires a more complex protocol than TrustBuilder Protocol 1, but does meet the other desiderata. Protocol 2 retains the structure of Protocol 1, but adds several new message types, as explained in this section. Further, the family we present requires complete disclosure of the unlocked portion of policy graphs that are considered relevant to the negotiation. This lessens the degree of protection that policy graphs provide for sensitive policies. With structured credentials, there seem to be many strategy families worthy of investigation, and the tradeoffs between them are a subject of our current research.

DEFINITION 8.1. *Let $G$ be the policy graph of a resource $R$ and let $\mathcal{C}$ be the set of disclosed credentials. A* policy graph disclosure *(or* traversal disclosure*) takes the form $(R, traversal(G, \mathcal{C}))$.*

At the beginning of a negotiation, since no credentials have been disclosed, the traversal only contains the source node in the resource's policy graph. Later on,

as more credentials are disclosed and more policy nodes are unlocked, traversal disclosures will cover more of a policy graph. In practice, only the new information in a traversal needs to be disclosed. However, to simplify our presentation, we assume each policy graph disclosure contains the complete current traversal of that graph.

DEFINITION 8.2. *Let $G$ be the policy graph of a resource $R$ and let $\mathcal{C}$ be the set of disclosed credentials. Let $\mathcal{C}'$ be a set of credentials such that $traversal(G, \mathcal{C})$ is a proper subgraph of $traversal(G, \mathcal{C} \cup \mathcal{C}')$. Then we say $\mathcal{C}'$* extends *$traversal(G, \mathcal{C})$ (or $\mathcal{C}'$ is an extension to $traversal(G, \mathcal{C})$). If no proper subset of $\mathcal{C}'$ has this property, then we say $\mathcal{C}'$ is a* minimal extension *to $traversal(G, \mathcal{C})$. An extension disclosure for $traversal(G, \mathcal{C})$ takes the form $extension(\mathcal{C}', R, traversal(G, \mathcal{C}))$.*

Of course, extension disclosures list credential local IDs, not actual credential contents. Intuitively, if $\mathcal{C}'$ is a minimal extension to $traversal(G, \mathcal{C})$, then $\mathcal{C}'$ is a minimal set of credentials that will reveal new structural information about $R$'s policy graph. Extensions formalize the concept of bindings discussed earlier.

During the negotiation, Alice's policy compliance checker may be invoked several times to get different extensions to the same traversal $G$. When there are no more new extensions (or there is no extension at all) to $G$, Alice will send a special extension disclosure in the form $no\_more\_extensions(R, G)$.

In summary, the following disclosures are allowed in TrustBuilder Protocol 2.

(1) Policy graph disclosures, $(R, traversal(G, \mathcal{C}))$.
(2) Extension disclosures, $extension(\{C_1, \ldots, C_k\}, R, traversal(G, \mathcal{C}))$ and $no\_more\_extensions(R, traversal(G, \mathcal{C}))$.
(3) Credential disclosures.

A message in TrustBuilder Protocol 2 is a set $\{R_1, \ldots, R_k\}$, where each $R_i$ is one of the above disclosures. As before, the empty message indicates that the negotiation has failed. The pseudocode for TrustBuilder Protocol 2 is the same as that for TrustBuilder Protocol 1 (figure 3). The difference between the two protocols resides in the types of information contained in messages. TrustBuilder Protocol 2 requires the negotiation strategies used with it to enforce the following conditions throughout negotiations:

(1) All credential disclosures must be safe. More precisely, a credential can be disclosed only after it is unlocked.
(2) All policy graph disclosures must be safe and complete. More precisely, when $(R, G)$ is disclosed, $G$ must be the traversal of resource $R$'s policy graph realized by the set of credentials disclosed so far.
(3) The same disclosure cannot be sent more than once.
(4) If $no\_more\_extensions(C, G)$ is disclosed in a message, then no extension disclosure $extension(\mathcal{C}, C, G)$ can be sent in subsequent messages, as there should be no more extensions to traversal $G$. This is required by the semantics of *no_more_extensions* disclosures.
(5) If a traversal disclosure $(R, G')$ is disclosed in a message, then no traversal disclosure $(R, G'')$ such that $G''$ is a proper subgraph of $G'$ can be disclosed in subsequent messages.

Our definition of policy graph disclosures is not the only possible way to disclose edges of a policy graph $G$. We mentioned earlier that traversals can be disclosed incrementally, a practical twist that has no effect on the underlying theory. Disclosing entire traversals rather than allowing Alice to disclose a subgraph of a traversal does have a significant impact on the theory, because Alice knows that she knows everything that Bob knows about part of $G$: no detail has been omitted, even if Alice has already satisfied that detail by her previous disclosures. This completeness allows Alice to reuse her knowledge of $G$ in different contexts that may arise later in the negotiation. Further, if she cannot find a solution set for any path through Bob's disclosed subgraph, Alice knows that she will never get access to the resource $G$ protects, and she can focus her efforts elsewhere. As discussed earlier, this power comes at a price: Bob must disclose policy graph details that he might prefer to keep private. On the other hand, if we do not require complete traversal disclosure, we obtain some nice privacy guarantees, but the resulting family of interoperable strategies excludes many strategies that are attractive from an implementation perspective but are not intelligent enough to meet certain increased reasoning requirements that flow from this seemingly small change in the definition of a policy graph disclosure.

In the remainder of the paper, we study interoperability among strategies that follow TrustBuilder Protocol 2 and enforce the above conditions. Similar to our discussion in section 5, we will introduce a new concept, called *binding trees*, to describe the progress of a trust negotiation, and based on it we will define a strategy family.

## 9. BINDING TREES AND THE BINDING TREE STRATEGY FAMILY

### 9.1 Binding Trees

DEFINITION 9.1. *A* binding tree *is a finite tree $T$ satisfying the following conditions:*

(1) *Each non-root node represents a credential, a service, a traversal of a policy graph, or else is a special node called a* more_extensions node.

(2) *The root represents the resource ("service") to which access was originally requested.*

(3) *Each credential/service node $C$ has at most one child. The child, if present, must be a traversal node that represents a traversal of $C$'s policy graph.*

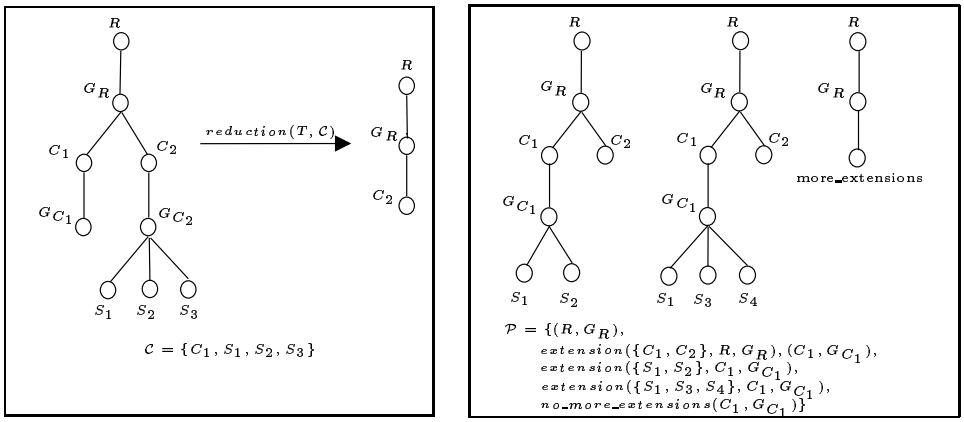
(4) *If a traversal node has children, they can either be credential nodes or a single more_extensions node.*

(5) *If a traversal node's children are credential nodes, then they represent an extension to the traversal.*

(6) *A more_extensions node has no children.*

*A credential node is* redundant *if in the path from the root to this node, there is another credential node representing the same credential. A binding tree is* redundant *if it has a redundant credential node.*

Binding trees have several major differences from disclosure trees. Because we support policy graphs, a credential will usually be associated with more than one




(a) Example of a binding tree reduction: $reduction(T, \mathcal{C})$

(b) Example of a binding tree expansion: $expansion(\mathcal{P})$

Fig. 9. Examples of operations on binding trees

traversal during a trust negotiation. Therefore, we need traversal nodes to distinguish different traversals. Second, we let credential nodes be children of traversal nodes, which is quite natural considering our definition of extension disclosures. Finally, a special "more_extensions" node is used because extensions to a traversal may not be disclosed all at once. We use a more_extensions node to indicate whether Alice has received all of Bob's extensions to a traversal.

Next we will show how to use binding trees to describe the progress of a trust negotiation.

DEFINITION 9.2. *Given a binding tree $T$ and a set $\mathcal{C}$ of credential disclosures, the* reduction *of $T$ by $\mathcal{C}$, written $reduction(T, \mathcal{C})$, is the binding tree $T'$ obtained by removing all the subtrees rooted at nodes that represent credentials in $\mathcal{C}$ or traversals that $\mathcal{C}$ extends. Given a set $\mathcal{T}$ of binding trees, the reduction of $\mathcal{T}$, written $reduction(\mathcal{T}, \mathcal{C})$, is the set $\{reduction(T, \mathcal{C}) \mid T \in \mathcal{T}\}$.*

An example of binding tree reduction is shown in figure 9(a).

As we mentioned earlier, during a trust negotiation, there may be several traversals for the same credential, corresponding to different stages of the negotiation. According to the definition of a traversal, a traversal disclosed earlier is always a subgraph of the ones disclosed later. Therefore, given a set of traversals of a credential, it is easy to identify the one disclosed most recently. In the rest of the paper, unless otherwise noted, when we refer to a credential's traversal, we always mean its most recent one disclosed so far during a negotiation.

DEFINITION 9.3. *Given a set $\mathcal{P}$ of extension disclosures and policy graph disclosures, the* expansion *of $\mathcal{P}$, written $expansion(\mathcal{P})$, is the set of all binding trees such that each binding tree $T$ in the set satisfies the following conditions:*

*(1) The child of a credential/service node $C$ represents the most recent traversal of $C$ that appears in $\mathcal{P}$.*

(2) *For every traversal node $G$ in $T$ whose parent is a node $C$, if the children of $G$ are $C_1, \ldots, C_t$, then extension$(\{C_1, \ldots, C_t\}, C, G) \in \mathcal{P}$.*

(3) *For every traversal node $G$ in $T$, if its child is a more_extensions node, then no_more_extensions$(C, G) \notin \mathcal{P}$.*

(4) *If a leaf node is a credential/service node $C$, then either $\mathcal{P}$ does not contain any traversal disclosure for $C$, or $C$ is a redundant node.*

(5) *If a traversal node $G$ has no child, then no_more_extensions$(C, G)$ is the only extension disclosure for $G$ in $\mathcal{P}$.*

An example of binding tree expansion is shown in figure 9(b).

LEMMA 9.1. *Let $\mathcal{C}$ contain all the credentials disclosed so far during a negotiation and let $R$ be a sensitive local resource whose policy graph is $G$. Let $P_0 \rightarrow \cdots \rightarrow P_k \rightarrow R$ be a policy path to $R$. If the remote party possesses credentials that can satisfy the policy path, then there exists an $i$, $0 \leq i \leq k$, such that $P_0 \rightarrow \cdots \rightarrow P_i$ is a path in traversal$(G, \mathcal{C})$ and $P_i$ is a sink node in traversal$(G, \mathcal{C})$.*

PROOF. According to the definition of traversal, the source node $P_0$ of $G$ belongs to *traversal*$(G, \mathcal{C})$. Let $P_i$ be the first node in the path such that $P_0 \rightarrow \cdots \rightarrow P_i$ is in *traversal*$(G, \mathcal{C})$ but $P_i \rightarrow P_{i+1}$ is not. Then $P_i$ must be a sink node in *traversal*$(G, \mathcal{C})$. Otherwise, $P_i$ should be context satisfied by $\mathcal{C}$, which means $P_{i+1}$ is unlocked. Therefore, the edge $P_i \rightarrow P_{i+1}$ is in *traversal*$(G, \mathcal{C})$, a contradiction. □

COROLLARY 9.1. *Let $\mathcal{C}$ contain all the credentials disclosed so far during a negotiation and let $R$ be a local resource with policy graph $G$. If the remote party does not possess any extension to traversal$(G, \mathcal{C})$, then there is no safe credential disclosure sequence that leads to the granting of access to $R$.*

PROOF. If the remote party can gain access to $R$, then there is a policy path $P_0 \rightarrow \cdots \rightarrow P_k \rightarrow R$ such that the remote party possesses a set of credentials $\mathcal{C}'$ satisfying $P_0 \wedge \cdots \wedge P_k$. According to lemma 9.1, there is a $P_i$ such that $P_0 \rightarrow \cdots \rightarrow P_i$ is in *traversal*$(G, \mathcal{C})$ and $P_i$ is a sink node. Since $\mathcal{C}'$ also satisfies $P_0 \wedge \cdots \wedge P_i$, $\mathcal{C}'$ is an extension to *traversal*$(G, \mathcal{C})$, which leads to a contradiction. □

Suppose $\mathcal{M}$ is the sequence of messages exchanged between Alice and Bob so far during a negotiation. If *no_more_extensions*$(C, G)$ is the only extension disclosure for $G$ that Alice sends to Bob, then, by corollary 9.1, Bob knows that Alice cannot gain access to his credential $C$. Thus, he may consider that message to be Alice's refusal of access to $C$. In the remainder of the paper, we will say a credential $C$ is *refused* when *no_more_extensions*$(C, G)$ is the only extension disclosure for $G$ in $\mathcal{M}$. A refusal of a credential is sent by the party who requested that credential, in contrast to a propositional denial disclosure.

DEFINITION 9.4. *Given a set $\mathcal{T}$ of binding trees and a set $\mathcal{C}_d$ of refused credentials, the* refusal pruning *of $\mathcal{T}$, written prune$_{refusal}(\mathcal{T}, \mathcal{C}_d)$, is the set*

$$\{T \mid T \in \mathcal{T} \text{ and } T \text{ does not contain credentials in } \mathcal{C}_d\}.$$

DEFINITION 9.5. *Given a set $\mathcal{T}$ of binding trees, the* redundancy pruning *of $\mathcal{T}$, written prune$_{redundant}(\mathcal{T})$, is the set*

$$\{T \mid T \in \mathcal{T} \text{ and } T \text{ is not redundant}\}.$$

Since the pruning operations are the same as those for disclosure trees, we do not show examples in this section.

DEFINITION 9.6. *Let $\mathcal{S}$ be the union of a set $\mathcal{P}$ of policy graph disclosures and extension disclosures, a set $\mathcal{C}$ of credential disclosures and a set $\mathcal{C}_d$ of refused credentials. The* view *regarding $\mathcal{S}$, written $view(\mathcal{S})$, is*

$$prune_{redundant}(prune_{refusal}(reduction(expansion(\mathcal{P}), \mathcal{C}), \mathcal{C}_d)).$$

LEMMA 9.2. *Let $\mathcal{S}$ contain all the disclosures made by the two negotiation participants at a certain stage of a negotiation. If there is a safe credential disclosure sequence that leads to the granting of access to resource $R$, then either $R$ is disclosed in $\mathcal{S}$ or $view(\mathcal{S}) \neq \emptyset$.*

PROOF. First, $R$ is not refused in $\mathcal{S}$. Otherwise, by corollary 9.1, there should be no safe disclosure sequence leading to the granting of access to $R$. Suppose the safe disclosure sequence is $(C_1, \ldots, C_k, R)$. Without loss of generality, we assume $(C_1, \ldots, C_k, R)$ is non-redundant. We will prove the lemma by induction on $k$.

When $k = 0$, $R$ is unprotected. If $R$ is not disclosed in $\mathcal{S}$, then $view(\mathcal{S})$ should contain a binding tree with only the root node $R$. The lemma holds.

Assume when $k < n$, $1 \leq n$, the lemma holds. When $k = n$, consider the non-trivial case where $R$ is not disclosed in $\mathcal{S}$ and those credentials disclosed in $\mathcal{S}$ do not extend $R$'s latest traversal. Let $R$'s latest traversal be $G_R$. If $no\_more\_extensions(R, G_R) \notin \mathcal{S}$, then $view(\mathcal{S})$ contains tree $T$, where the root has a traversal node $G_R$ whose only child is a more_extensions node. If $no\_more\_extensions(R, G_R) \in S$, then $\mathcal{S}$ contains all the extensions to $G_R$. By lemma 9.1, there exists a set $\mathcal{C} \subseteq \{C_1, \ldots, C_k\}$ which is a minimal extension to $G_R$. Since $G_R$ is not extended by credentials in $\mathcal{S}$, there exist credentials in $\mathcal{C}$ that are not disclosed in $\mathcal{S}$. Let $\mathcal{C}' \subseteq \mathcal{C}$ contain all such credentials. For all $C \in \mathcal{C}'$, $(C_1, \ldots, C_i = C)$ is a safe disclosure sequence and $i < k$. By the induction hypothesis, we can have a binding tree whose root represents $C$. By attaching all such trees to the traversal node $G_R$, we get a binding tree $T$ for $R$ and $T \in view(\mathcal{S})$. □

If the policy graphs of Alice and Bob allow a successful negotiation, then there is always a non-redundant safe disclosure sequence leading to the granting of access to $R$. Therefore, by lemma 9.2, we have the following corollary immediately.

COROLLARY 9.2. *If there exists a sequence of messages $(m_1, \ldots, m_k)$ exchanged between two parties such that for all $m_i$, $1 \leq i \leq k$, $R \notin m_i$ and $view(\bigcup_{1 \leq i \leq k} m_i) = \emptyset$, then there is no safe disclosure sequence that leads to the granting of access to $R$.* □

The view of an ongoing negotiation represents all the latest disclosures as well as the connections between them. By examining the binding trees in a view of a negotiation, we can determine which traversals and credentials are relevant to a negotiation. For example, let $C$ be Alice's credential and let $G$ be $C$'s latest traversal. Suppose that $G$ is also a traversal node of a binding tree $T$ in the view. If $G$ has a more_extensions node as its child, then Alice knows she may expect Bob to disclose more extensions to $G$ later (or else disclose $no\_more\_extensions(C, G)$). Similarly, if $C$ is a leaf node in $T$, then Bob will expect Alice to disclose the latest traversal of $C$'s policy graph.

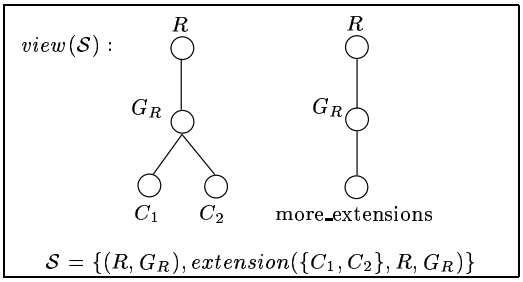



Fig. 10. Suppose Bob only possesses $R$. After receiving $R$'s latest traversal $G_R$, if Alice only discloses a binding to $G_R$, then Bob cannot make any further disclosure in the next message.

Though we have extended the TrustBuilder protocol and introduced new concepts to accommodate structured credentials and policy graphs, our definitions for a strategy, strategy family and a generator are unchanged.

### 9.2 The Binding Tree Strategy

Consider the example shown in figure 10. Suppose $R$ is Bob's only resource. Further, suppose $\{C_1, C_2\}$ is an extension to $R$'s traversal $G_R$. If Alice only sends $extension(\{C_1, C_2\}, R, G_R)$ to Bob, then, according to TrustBuilder Protocol 2, Bob has no further disclosures. This example shows that whenever Alice sends a message to Bob, Alice must ensure that Bob can respond with a non-failure message, unless failure is inevitable. We call this a *continuity* requirement. If Alice and Bob both materialize all the binding trees, then the continuity requirement can be met easily, and we can also guarantee that every disclosure a strategy makes is relevant to the current negotiation. However, it may take exponential time and space to materialize binding trees in a view, so materialization is impractical.

Fortunately, there are other ways to meet the continuity requirement. For example, suppose Alice discloses a policy graph's latest traversal $G$. Then in the next message, Bob can disclose an extension to $G$ or refuse access to the credential that $G$ protects. Similarly, if Alice discloses credentials that satisfy the traversal of the policy graph of one of Bob's credentials, then the negotiation will proceed because Bob can at least disclose that credential's policy graph's new traversal, or the credential itself if it is now unlocked. Formally, we have the following definition.

DEFINITION 9.7. *Given a set $\mathcal{S}$ of disclosures, we say Alice can* expand *$\mathcal{S}$, if one of the following conditions holds.*

(1) *Alice has a credential $C$ such that*
   —*$C$ appears in an extension disclosure in $\mathcal{S}$ and there is no traversal disclosure of $C$'s policy graph in $\mathcal{S}$; or,*
   —*Credentials disclosed in $\mathcal{S}$ extend $C$'s latest traversal in $\mathcal{S}$.*

(2) *Bob has a credential $C$ such that*
   —*A traversal disclosure of $C$'s policy graph is in $\mathcal{S}$; and*
   —*The credentials disclosed in $\mathcal{S}$ do not extend the latest traversal $G$ of $C$'s policy graph in $\mathcal{S}$; and*

—*no_more_extensions*$(C, G)$ *is not in* $\mathcal{S}$.

Note that Bob can easily check whether Alice can expand $\mathcal{S}$ without materializing binding trees.

LEMMA 9.3. *Let* $\mathcal{S}$ *be a set of disclosures that Alice can expand, and let* $\mathcal{S}'$ *be a set of disclosures that Bob could make. Then Alice can expand* $\mathcal{S} \cup \mathcal{S}'$.

PROOF. No matter what Bob discloses in $\mathcal{S}'$, the two conditions in definition 9.7 will still hold for $\mathcal{S} \cup \mathcal{S}'$ if they hold for $\mathcal{S}$. Therefore, Alice can expand $\mathcal{S} \cup \mathcal{S}'$. □

DEFINITION 9.8. *Suppose Alice and Bob are the two negotiation parties. Let* $\mathcal{M} = (m_1, \ldots, m_k)$ *be a sequence of messages such that* $m_i \neq \emptyset$ *and* $R \notin m_i$ *for* $1 \leq i \leq k$. *Let* $L_A$ *and* $L_B$ *be the local credentials, services and policy graphs of Alice and Bob respectively. Let* $\mathcal{S}_\mathcal{M} = \bigcup_{1 \leq i \leq k} m_i$. *Without loss of generality, assume it is Alice's turn to send the next message to Bob. The* binding tree strategy (BTS) *is a strategy such that* $BTS(\mathcal{M}, L_A, R)$ *satisfies the following conditions:*

(1) $BTS(\mathcal{M}, L_A, R) = \{\emptyset\}$ *if and only if one of the following conditions holds:*
   —$view(\mathcal{S}_\mathcal{M}) = \emptyset$.
   —*There exists a set* $\mathcal{S}'$ *of safe disclosures that Alice can make such that* $view(\mathcal{S}_\mathcal{M} \cup \mathcal{S}') = \emptyset$.
   —*Alice cannot expand* $\mathcal{S}_\mathcal{M}$.

(2) *Otherwise,* $BTS(\mathcal{M}, L_A, R)$ *contains all the messages* $m'$ *such that one of the following conditions holds:*
   —$m' = \{R\}$, *if Alice possesses* $R$ *and* $R$ *is unlocked by credentials disclosed in* $\mathcal{S}_\mathcal{M}$;
   —$m'$ *is a minimal non-empty set of disclosures such that* $\mathcal{S}_\mathcal{M} \cap m' = \emptyset$ *and Bob can expand* $\mathcal{S}_\mathcal{M} \cup m'$.

Is it possible that Alice cannot find a non-empty set $m'$ of disclosures such that Bob can expand $S \cup m'$? The following lemma answers the question.

LEMMA 9.4. *Let* $\mathcal{S}$ *be the set of all disclosures made so far during a trust negotiation, and suppose Alice can expand* $\mathcal{S}$. *Suppose it is Alice's turn to send the next message. Let* $\mathcal{S}'$ *be the set of all the safe disclosures Alice can make in the next message. Then if* $view(\mathcal{S} \cup \mathcal{S}') \neq \emptyset$, *Bob can expand* $\mathcal{S} \cup \mathcal{S}'$.

PROOF. Since Alice can expand $\mathcal{S}$, $\mathcal{S}' \neq \emptyset$. Let $T \in view(\mathcal{S} \cup \mathcal{S}')$. According to definition 9.6, $T$ is not redundant and $T$ does not contain nodes representing credentials refused in $\mathcal{S} \cup \mathcal{S}'$. Consider a leaf node of $T$. We have:

(1) The leaf node cannot represent a traversal $G$ whose parent represents $C$, one of Bob's credentials. Otherwise, since $T \in view(\mathcal{S} \cup \mathcal{S}')$, $C$ is not refused in $\mathcal{S} \cup \mathcal{S}'$. Therefore, $\mathcal{S} \cup \mathcal{S}'$ should contain all of Alice's extensions to $G$, which means $G$ cannot be a leaf node.

(2) The leaf node cannot represent a credential $C$ of Alice's. Otherwise, since $C$ is a leaf node of $T$, $C$ is not unlocked by credentials disclosed in $\mathcal{S}$. Therefore, $\mathcal{S} \cup \mathcal{S}'$ should contain $C$'s latest traversal, which means $C$ should not be a leaf node.

(3) If the leaf node is a more_extensions node, then its parent cannot represent a traversal $G$ of a credential $C$ of Bob's. This is because $\mathcal{S} \cup \mathcal{S}'$ contains all of Alice's extensions to $G$ (including $no\_more\_extensions(C, G)$). Therefore $G$ should not have a more_extensions node as its child in $T$.

Therefore, the leaf node can only represent a credential of Bob's, or a more_extensions node whose parent represents a traversal of one of Alice's credentials. According to definition 9.7, in either case, Bob can expand $\mathcal{S} \cup \mathcal{S}'$. □

Lemma 9.4 tells us that if $BTS(M, L_A, R) \neq \{\emptyset\}$, then there always exists a non-empty minimal set $m'$ of disclosures of Alice such that $\mathcal{S} \cap m' = \emptyset$ and Bob can expand $\mathcal{S} \cup m'$.

THEOREM 9.1. *The set of strategies generated by BTS is a family.*

PROOF. Suppose Alice and Bob adopt strategies $f_1$ and $f_2$, respectively, from the set of strategies generated by BTS. We only need to prove that when the negotiation fails, there is no safe disclosure sequence leading to the granting of access to the originally requested resource $R$.

When the negotiation fails, without loss of generality, we assume it is Alice that sends the failure message. Suppose that before Alice sends the failure message, $\mathcal{M} = (m_1, \ldots, m_k)$ is the sequence of exchanged messages. Therefore $m_k$ is the last message that Bob sends to Alice. Let $L_A$ and $L_B$ be the local credentials, services, and policy graphs of Alice and Bob respectively, and $\mathcal{S}_\mathcal{M} = \bigcup_{1 \leq i \leq k} m_i$. Since $\emptyset \in f_1(\mathcal{M}, L_A, R)$ and $f_1 \succeq BTS$, we must have $BTS(\mathcal{M}, L_A, R) = \{\emptyset\}$. According to definition 9.8, one of the following conditions holds:

(1) $view(\mathcal{S}_\mathcal{M}) = \emptyset$.

(2) There exists a set $\mathcal{S}'$ of safe disclosures of Alice such that $view(\mathcal{S}_\mathcal{M} \cup \mathcal{S}') = \emptyset$.

(3) Alice cannot expand $\mathcal{S}_\mathcal{M}$.

When condition 3) holds but not 1) and 2), we must have $BTS((m_1, \ldots, m_{k-1}), L_B, R) = \{\emptyset\}$. Otherwise, since $f_2 \succeq BTS$ and $R \notin m_k$, $m_k$ must be a superset of a minimal set $m'$ of disclosures such that Alice can expand $(\mathcal{S}_\mathcal{M} - m_k) \cup m'$. According to lemma 9.3, Alice can expand $\mathcal{S} = ((\mathcal{S}_\mathcal{M} - m_k) \cup m') \cup (m_k - m')$, which contradicts the assumption that condition 3) holds. Since $BTS((m_1, \ldots, m_{k-1}), L_B, R) = \{\emptyset\}$, before Bob sent $m_k$, one of the above conditions was true. At the beginning of the negotiation, condition 3) was not true (when a client sends a request to access resource $R$, the server can expand the empty disclosure sequence), we know that in some previous stage of the negotiation, condition 1) or 2) must have been true.

When condition 1) or 2) holds, according to corollary 9.2, there is no safe disclosure sequence leading to the granting of access to $R$. □

THEOREM 9.2. *If a strategy $f$ and BTS are compatible, then $f \succeq BTS$.*

PROOF. Suppose Alice is using $f$ and $L_A$ contains her local credentials, services and policy graphs. We will prove the theorem by contradiction.

Assume $f \not\succeq BTS$. Then there exists a sequence of messages $\mathcal{M} = (m_1, \ldots, m_k)$ such that

$$\exists m' \in f(\mathcal{M}, L_A, R) \text{ such that } \forall m \in BTS(\mathcal{M}, L_A, R), m \not\subseteq m'.$$

Let $\mathcal{S}_{\mathcal{M}} = \bigcup_{1 \leq i \leq k} m_i$. Then all of the following must be true.

(1) $view(\mathcal{S}_{\mathcal{M}}) \neq \emptyset$.

(2) $view(\mathcal{S}_{\mathcal{M}} \cup \mathcal{S}') \neq \emptyset$, where $\mathcal{S}'$ contains all the safe disclosures Alice can make in the next message.

(3) Alice can expand $\mathcal{S}_{\mathcal{M}}$.

Otherwise, according to definition 9.8, $\emptyset \in BTS(\mathcal{M}, L_A, R)$ and $\emptyset \subseteq m'$, which leads to a contradiction. Also we have $R \notin m'$. Otherwise, $R$ is unlocked by credentials disclosed in $\mathcal{S}_{\mathcal{M}}$, which means $\{R\} \in BTS(\mathcal{M}, L_A, R)$. And $\{R\} \subseteq m'$.

Since every $m \in BTS(\mathcal{M}, L_A, R)$ is a non-empty minimal set of disclosures such that Bob can expand $\mathcal{S}_{\mathcal{M}} \cup m$, Bob cannot expand $\mathcal{S}_{\mathcal{M}} \cup m'$. Thus, after sending $m'$, Bob's strategy BTS will send a failure message and end the negotiation.

However, consider a binding tree $T \in view(\mathcal{S}_{\mathcal{M}} \cup \mathcal{S}')$. By the proof of theorem 9.1, every leaf node in $T$ either represents one of Bob's credentials or a more_extensions node whose parent is the latest traversal of one of Alice's credentials. By induction on the height of $T$, we will prove that there is an arrangement of Bob's local credentials and policy graphs such that there is a safe disclosure sequence that leads to the granting of access to $R$. If we are successful in proving our claim, then that means there should be a successful negotiation between the two parties. Thus if Alice terminates the negotiation as a failure, this contradicts the fact that $f$ and $BTS$ are compatible.

We use $height(T)$ to denote the height of $T$, ignoring more_extensions nodes.

When $height(T) = 1$, $T$ only contains the root node $R$. Thus $R$ is owned by Bob. Let $G_R$ be $R$'s latest traversal disclosed in $S$. We can choose $R$'s policy graph to have an edge from every sink node in $G_R$ to $R$. Therefore, if the disclosures in $\mathcal{S}_{\mathcal{M}} \cup \mathcal{S}'$ are made, $R$ can be unlocked, which means that there is a successful negotiation.

When $|T| = 2$, besides a possible more_extensions node, $T$ will only contain the root node $R$ and its latest traversal $G_R$. In this case, $R$ is owned by Alice. Since $R$'s policy graph is not disclosed and $R$ is not refused by Bob in $\mathcal{S}_{\mathcal{M}}$, according to our assumption about a strategy's reasoning ability (i.e., if a policy (or traversal) is not explicitly refused by Bob, Alice cannot infer whether Bob can satisfy that policy), we can choose for Bob to have a set $\mathcal{C}$ of unprotected credentials such that $\mathcal{C} \cap \mathcal{S}_{\mathcal{M}} = \emptyset$ and credentials in $\mathcal{C} \cup \mathcal{S}_{\mathcal{M}}$ context satisfy $R$.

Assume when $|T| < n$, $n \geq 2$, our claim holds.

When $|T| = n$, consider a child node $C$ of $R$'s traversal node $G_R$ in $T$. The height of the subtree rooted at $C$ is smaller than $n$. By the induction hypothesis, we can choose Bob's local credentials and policy graphs such that $C$ can be unlocked.
If Bob possesses $R$, similarly to the case when $height(T) = 1$, we can choose $R$'s policy graph such that there is an edge from every sink node in $G_R$ to $R$. Thus when credentials represented by the children node of $G_R$ are disclosed, $R$ is unlocked.

If Alice possesses $R$, let the children of $G_R$ be $\{C_1, \ldots, C_k\}$. Similarly to the case when $height(T) = 2$, according to our assumption about a strategy's reasoning ability, we can choose for Bob to have a set $\mathcal{C}$ of unprotected credentials such that $\mathcal{C} \cap (\mathcal{S}_{\mathcal{M}} \cup \{C_1, \ldots, C_k\}) = \emptyset$ and $\mathcal{C} \cup \mathcal{S}_{\mathcal{M}} \cup \{C_1, \ldots, C_k\}$ context satisfies $R$. □

We call the family generated by BTS the *BTS family*.

COROLLARY 9.3. *The BTS family is closed.* □

THEOREM 9.3. *Let $f_1$ and $f_2$ be strategies in the BTS family and let $f'$ be a hybrid of $f_1$ and $f_2$. Then $f'$ is also in the BTS family.*

PROOF. The proof is exactly the same as that for theorem 6.3. □

As mentioned earlier, when we designed TrustBuilder Protocol 2, one of the major desiderata was that some strategy family following the protocol should contain efficient strategies that are easy to implement in practice. Now we will present two such members of the BTS family. They both take parameters $\mathcal{M} = (m_1, \ldots, m_k)$, a sequence of messages; $L$, the set of all local resources and policy graphs; and $R$, the resource to which the client originally requested access. The output is a set containing a single message.

The first strategy is TrustBuilder2-Simple, which makes no effort to identify which disclosures are relevant. When it sends a message, it simply tries to make sure the other party can expand the resulting message sequence. It will terminate the negotiation only if it fails to find such a message. The pseudocode for the TrustBuilder2-Simple strategy is shown in figure 11(a).

We can update the definition of *syntactic relevance* as follows:

(1) $R$ is syntactically relevant to $R$.

(2) A traversal of a relevant resource is syntactically relevant to $R$.

(3) A credential that appears in an extension to a relevant traversal is syntactically relevant to $R$.

The second efficient strategy, the TrustBuilder2-Relevant strategy, only makes disclosures that are syntactically relevant to $R$ whenever possible. The pseudocode for the TrustBuilder2-Relevant strategy is shown in figure 11(b). An example of the two strategies interacting is given in Appendix A.

THEOREM 9.4. *Both TrustBuilder2-Simple and TrustBuilder2-Relevant belong to the BTS family.*

PROOF. By theorem 9.2, we only need to prove that the two strategies are compatible with BTS.

Suppose Alice and Bob are using TrustBuilder2-Simple and BTS respectively. When the negotiation fails, let $\mathcal{M} = (m_1, \ldots, m_k)$ be the message exchange sequence before the failure message is sent and let $\mathcal{S}_\mathcal{M} = \bigcup_{1 \leq i \leq k} m_i$. If Alice sends the failure message, according to the pseudocode for TrustBuilder2-Relevant, Alice cannot find a set $m$ of disclosures such that $m \cap \mathcal{S}_\mathcal{M} = \emptyset$ and Bob can expand $\mathcal{S}_\mathcal{M} \cup m$. According to lemma 9.4, there is no safe disclosure sequence leading to the granting of access to $R$. On the other hand, if Bob sends the failure message, then $m_k$ is sent by Alice. Thus Bob can expand $\mathcal{S}_\mathcal{M}$. According to definition 9.8, we have either $view(\mathcal{S}_\mathcal{M}) = \emptyset$ or $view(\mathcal{S}_\mathcal{M} \cup \mathcal{S}') = \emptyset$, where $S'$ contains all the disclosures that Bob can make in the next message. By corollary 9.2, there is no safe disclosure sequence leading to the granting of access to $R$.

Since TrustBuilder2-Simple should be invoked before TrustBuilder2-Relevant sends a failure message, the argument above is also true for TrustBuilder-Relevant. In summary, both strategies are compatible with BTS. □

**The TrustBuilder2-Simple Strategy**
$\mathcal{S}_{\mathcal{M}} = \bigcup_{1 \leq i \leq k} m_i$.
Let $\mathcal{C}$ contain all the credentials disclosed in $\mathcal{S}_{\mathcal{M}}$.
$m = \emptyset$.
For every local credential $C$ that is unlocked by $\mathcal{S}_{\mathcal{M}}$
  $m = m \cup \{C\}$.
For every local locked credential $C$
  Let $C$'s policy graph be $G_C$.
  if $((C, traversal(G_C, \mathcal{C})) \notin \mathcal{S}_{\mathcal{M}})$
  then $m = m \cup \{(C, traversal(C, \mathcal{C}))\}$.
For every most recent traversal $G'$ for a remote credential $C'$ in $\mathcal{S}_{\mathcal{M}}$
such that $no\_more\_extensions(C', G') \notin \mathcal{S}_{\mathcal{M}}$
  Find an extension $\mathcal{C}'$ to $G'$ such that for all $\mathcal{C}'' \subseteq \mathcal{C}'$, $extension(\mathcal{C}'', C', G') \notin \mathcal{S}_{\mathcal{M}}$.
  if (there is no such $\mathcal{C}'$) then $m = m \cup \{no\_more\_extensions(C', G')\}$.
  else $m = m \cup \{extension(\mathcal{C}', C', G')\}$.
$m = m - \mathcal{S}_{\mathcal{M}}$.
if (the other party cannot expand $m \cup \mathcal{S}_{\mathcal{M}}$)
then return $\{\emptyset\}$.
else return $\{m\}$.

(a)

**The TrustBuilder2-Relevant Strategy**
$\mathcal{S}_{\mathcal{M}} = \bigcup_{1 \leq i \leq k} m_i$.
Let $\mathcal{C}$ contain all the credentials disclosed in $\mathcal{S}_{\mathcal{M}}$.
$m = \emptyset$.
For every most recent traversal $G'$ for a remote credential $C'$ in $\mathcal{S}_{\mathcal{M}}$
such that $no\_more\_extensions(C', G') \notin \mathcal{S}_{\mathcal{M}}$
  if ($C'$ is relevant to $R$) then
    Find an extension binding $\mathcal{C}'$ for $G'$
    such that for all $\mathcal{C}'' \subseteq \mathcal{C}'$, $extension(\mathcal{C}'', C', G') \notin \mathcal{S}_{\mathcal{M}}$.
    if (there is no such $\mathcal{C}'$) then $m = m \cup \{no\_more\_extensions(C', G')\}$.
    else $m = m \cup \{extension(\mathcal{C}', C', G')\}$.
For every local credential $C$ syntactically relevant to $R$
  Let $G_C$ be $C$'s policy graph.
  if ($C$ is unlocked by $\mathcal{S}_{\mathcal{M}}$) then $m = m \cup \{C\}$.
  else if $((C, traversal(G_C, \mathcal{C})) \notin \mathcal{S}_{\mathcal{M}})$
    then $m = m \cup \{(C, traversal(G_C, \mathcal{C}))\}$.
$m = m - \mathcal{S}_{\mathcal{M}}$.
if ($m = \emptyset$) then //other party might be very unintelligent
  $m'$=TrustBuilder2-Simple($M, L, R$).
  if ($m' \neq \emptyset$) then
    let $m$ be a minimal non-empty subset of $m'$ such that the other party can expand $m \cup \mathcal{S}_{\mathcal{M}}$.
return $\{m\}$.

(b)

Fig. 11. Pseudocode for two strategies in the BTS family

## 10. SUMMARY AND FUTURE WORK

This paper has focused on guaranteeing interoperability between different strategies. We first propose a very simple trust negotiation protocol for the TrustBuilder trust negotiation architecture. Then we studied strategies that adhere to this protocol. We introduced the concepts of strategy families and closed sets of strategies. If two strategies are in the same strategy family, then they will always correctly interoperate with each other. Closure expresses the maximality of a strategy family, i.e., if we add another strategy to a closed family, the resulting set of strategies

is no longer a family. In practice, we want to identify closed families of strategies because they give negotiation participants maximum freedom in choosing the strategies appropriate for them.

In the first half of the paper, we modeled access control policies as propositional logic formulas, introduced the concept of disclosure trees, and identified the natural mapping between full disclosure trees and safe credential disclosure sequences. We then proposed a strategy called the disclosure tree strategy (DTS), and proved that all the strategies that are no more cautious than DTS form a closed strategy family.

To overcome the limited expressiveness of policies based on propositional logic, in the second half of the paper, we extended our discussion to credentials with internal structure and policy graphs that protect sensitive policies. We proposed TrustBuilder Protocol 2 to accommodate this new model and identified a closed strategy family, the BTS family, which possesses almost all the desirable properties of the DTS family.

The TrustBuilder project has both a theoretical component (e.g., the results presented in this paper) and a practical component. As the theory must precede the implementation, many of the results reported in this paper are not yet included in the TrustBuilder prototypes. The current implementations utilize the IBM Trust Establishment (TE) system [Herzberg et al. 2000; Herzberg and Mass 2001] to create X.509v3 certificates. Attributes associated with credential owners are stored in user-defined extensions of X.509v3 certificates. The TE system supports XML role-based access control policies that TrustBuilder uses to govern access to sensitive credentials, policies, and services. The TE runtime system includes a compliance checker that TrustBuilder uses to verify whether a set of certificates satisfies an access control policy and to determine which credentials satisfy a policy. We have implemented prototypes of TrustBuilder using two existing communication protocols: on top of HTTPS [Rescorla 1998], and as an extension to the TLS handshake protocol [Hess et al. 2002]. The HTTPS prototype supports trust negotiation between a web client and server, with negotiation confidentiality provided by SSL, while confidentiality in the TLS version is provided by placing the negotiation in the context of a TLS rehandshake. Recent work has focused on incorporating policy disclosures into the prototypes. TrustBuilder prototypes for other environments are also planned; our goal is to create a scalable, reusable trust negotiation component that can be used with a variety of communication protocols.

Though in this paper we focus our efforts on interoperable strategies for trust negotiation, there are many related issues that interest us. In particular, we are interested in formally modeling information flow during trust negotiation and investigating privacy preservation. Also, we are interested in ways for parties to avoid, detect, and deflect denial-of-service attacks launched through time-wasting trust negotiations. Further, we believe that there exist many strategy families, shaped in part by their underlying protocols. We plan to formally identify the desirable properties that can guide us in designing protocols and strategy families.

## ACKNOWLEDGMENTS

This research was sponsored by DARPA through Space and Naval Warfare Systems Center San Diego grant number N66001-01-18908 (BYU) and AFRL contract

numbers F33615-01-C-1805 (BYU) and F30602-97-C-0336 (NAI Labs). We would like to thank Jim Gray and Shanghua Teng for their constructive discussions and suggestions on an earlier version of this paper. We would also like to thank Michael Halcrow, Ryan Jarvis and Bryan Smith for their help during our preparation of this paper.

## REFERENCES

APT, K. R., WARREN, D. S., AND (EDITOR), M. T. 1999. *The Logic Programming Paradigm: A 25-Year Perspective*. Springer-Verlag.

BLAZE, M., FEIGENBAUM, J., IOANNIDIS, J., AND KEROMYTIS, A. 1999. The KeyNote Trust Management System Version 2. In *Internet Draft RFC 2704*.

BLAZE, M., FEIGENBAUM, J., AND KEROMYTIS, A. D. 1998. KeyNote: Trust Management for Public-Key Infrastructures. In *Security Protocols Workshop*. Cambridge, UK.

BONATTI, P. AND SAMARATI, P. 2000. Regulating Service Access and Information Release on the Web. In *Conference on Computer and Communications Security*. Athens.

DIERKS, T. AND ALLEN, C. 1999. The TLS Protocol Version 1.0. IETF.

FARRELL, S. 1998. TLS Extension for Attribute Certificate Based Authorization. IETF.

FRIER, A., KARLTON, P., AND KOCHER, P. 1996. *The SSL 3.0 Protocol*. Netscape Communications Corp.

HERZBERG, A. AND MASS, Y. 2001. Relying Party Credentials Framework. In *The Cryptographer's Tract at RSA Conference*. San Francisco, CA.

HERZBERG, A., MIHAELI, J., MASS, Y., NAOR, D., AND RAVID, Y. 2000. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *IEEE Symposium on Security and Privacy*. Oakland, CA.

HESS, A., JACOBSON, J., MILLS, H., WAMSLEY, R., SEAMONS, K., AND SMITH, B. 2002. Advanced Client/Server Authetication in TLS. In *Network and Distributed System Security Symposium*. San Diego, CA.

IETF 2001. *Simple Public Key Infrastructure (SPKI)*. IETF.

IETF 2002. *Public-Key Infrastructure (X.509) (pkix)*. IETF.

ISLAM, N., ANAND, R., JAEGER, T., AND RAO, J. R. 1997. A Flexible Security System for Using Internet Content. *IEEE Software 14*, 5 (Sept.).

JOHNSON, W., MUDUMBAI, S., AND THOMPSON, M. 1998. Authorization and Attribute Certificates for Widely Distributed Access Control. In *IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*.

LI, N., GROSOF, B., AND FEIGENBAUM, J. 2000. A Practically Implementable and Tractable Delegation Logic. In *IEEE Symposium on Security and Privacy*. Berkeley, California.

LI, N., WINSBOROUGH, W., AND MITCHELL, J. 2001. Distributed Credential Chain Discovery in Trust Management. In *Conference on Computer and Communication Security*. Philadelphia, PA.

RESCORLA, E. 1998. HTTP Over TLS. IETF.

SAGONAS, K., SWIFT, T., AND WARREN, D. 1994. Xsb as an efficient deductive database engine. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*. ACM Press, Minneapolis, MN, 442–453.

SEAMONS, K., WINSLETT, M., AND YU, T. 2001. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Network and Distributed System Security Symposium*. San Diego, CA.

W3C 2002. *Platform for Privacy Preferences (P3P) Specification*. W3C.

WINSBOROUGH, W., SEAMONS, K., AND JONES, V. 2000. Automated Trust Negotiation. In *DARPA Information Survivability Conference and Exposition*. Hilton Head Island, SC.

YU, T., MA, X., AND WINSLETT, M. 2000. PRUNES: An Efficient and Complete Strategy for Automated Trust Negotiation over the Internet. In *Conference on Computer and Communication Security*. Athens, Greece.

YU, T., WINSLETT, M., AND SEAMONS, K. 2001. Interoperable Strategies in Automated Trust Negotiation. In *Conference on Computer and Communication Security*. Philadelphia, PA.

ZIMMERMAN, P. 1994. *PGP User's Guide*. MIT Press.

## A. EXAMPLE TRUST NEGOTIATION WITH STRUCTURED CREDENTIALS AND POLICY GRAPHS

This example involves a request to view Susan Jones's psychiatric record at Busey Hospital. Busey Hospital's policy graph for that record has multiple levels of nodes. The first important level says that the requester must be a physician at Busey Hospital, or else be the patient. The second important level says that the only doctor who can view the record is Jane Smith. The hospital also has an institutional ID that it is willing to show to anyone.

The requester, Susan Jones, has two credentials of interest. She is a patient at Busey Hospital, and she also works there. She will show her employee ID only to Busey Hospital. Her first important level of policies for her patient ID says that it can only be shown to people who work at Busey Hospital, Blue Cross Blue Shield of Illinois, or to Busey Hospital itself. Her second important level of policies adds the restriction that Busey Hospital employees must be staff physicians, staff nurses, or be a customer service representative in the delinquent accounts department. Figure 12 summarizes the structure of the relevant policy graphs, and the exact policy contents are given below.

**Server (Busey Hospital) policies:**

institutionID
  policy 1: *true*

psychRecord
  policy 2: *true*
  policy 3:
    $x$.type = "patient ID" $\wedge$ $x$.issuerPublicKey = publicKey("Busey Hospital")
    $\wedge$ $x$.patientNumber = "12345" $\wedge$ requesterAuthenticatesTo($x$.ownerPublicKey)
  policy 4:
    $y$.type = "employee ID"
    $\wedge$ $y$.issuerPublicKey = publicKey("Busey Hospital")
    $\wedge$ $y$.jobtitle = "Staff Physician" $\wedge$ requesterAuthenticatesTo($y$.ownerPublicKey)
  policy 5: $y$.employeeName = "Jane Smith"

**Client (patient Susan Jones and employee Dr. Sue Jones) policies:**

employeeID
  policy 6:
    $x$.type = "institutional ID"
    $\wedge$ $x$.issuerPublicKey = publicKey("Busey Hospital")
    $\wedge$ $x$.ownerPublicKey = publicKey("Busey Hospital")
    $\wedge$ requesterAuthenticatesTo(publicKey("Busey Hospital"))

patientID
  policy 7: *true*
  policy 8:
    $x$.type = "employee ID"

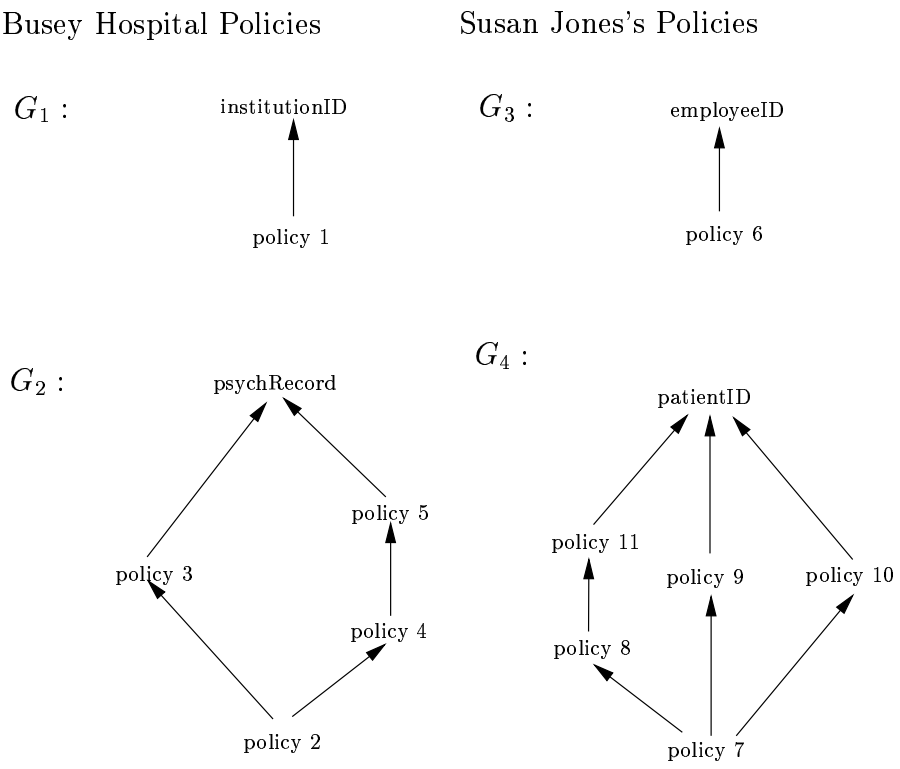



Fig. 12. Policy graphs for credentials of the client and the server.

$\wedge$ $x$.issuerPublicKey = publicKey("Busey Hospital")
$\wedge$ requesterAuthenticatesTo($x$.ownerPublicKey)

policy 9:
$y$.type = "employee ID"
$\wedge$ $y$.issuerPublicKey = publicKey("Blue Cross Blue Shield of Illinois")
$\wedge$ requesterAuthenticatesTo($y$.ownerPublicKey)

policy 10:
$z$.type = "institutional ID"
$\wedge$ $z$.issuerPublicKey = publicKey ("Busey Hospital")
$\wedge$ requesterAuthenticatesTo(publicKey("Busey Hospital"))

policy 11:
($x$.jobtitle = "Staff Physician" $\vee$ $x$.jobtitle = "Staff Nurse"
$\vee$ ($x$.jobtitle = "Customer Service Representative"
$\wedge$ $x$.department = "Delinquent Accounts"))

We have given descriptive names to the credentials in this example, to help the reader. In the real world, the credential and policy names must not reveal useful information. Also for realism, we have included the *RequesterAuthenticatesTo* predicate, whose interpretation in a model of a formula is determined at run time

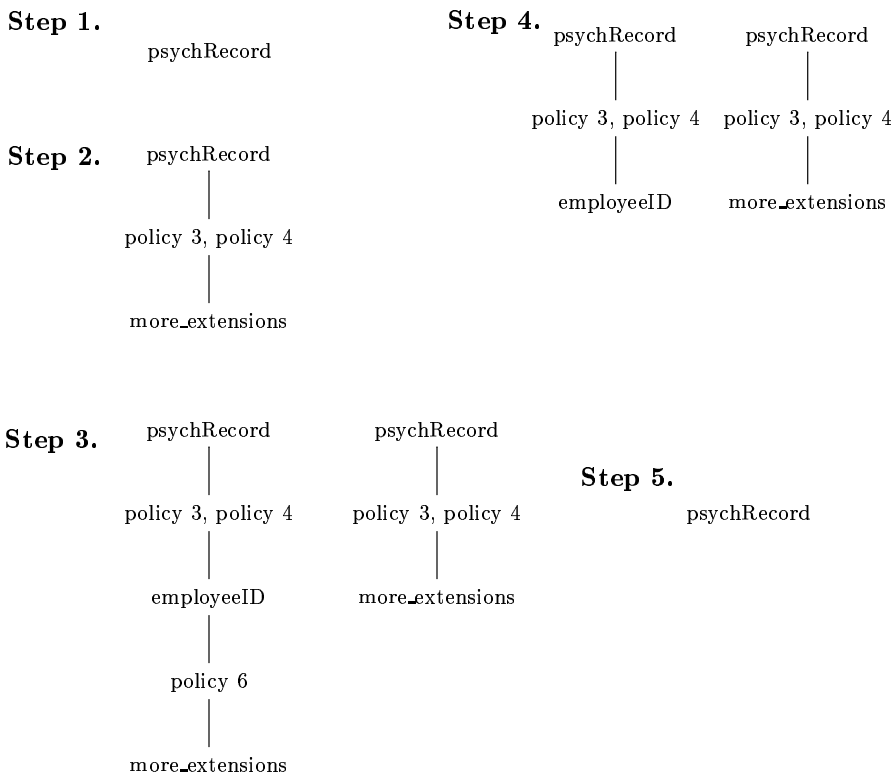



Fig. 13. Binding trees for each step in the example negotiation.

by whether or not the requester can demonstrate knowledge of the private key associated with a particular public key. Finally, we have included a function *publicKey*, whose interpretation is, in practice, supplied by a call to a certification authority that publishes public keys.

A trust negotiation is described below. Here Busey Hospital is using the TrustBuilder2-Relevant strategy while Susan Jones is using TrustBuilder2-Simple. The binding trees associated with the end of each step of the message exchange sequence are shown in figure 13. In figure 13 and our discussion, for conciseness, we use the sink nodes of a traversal to represent the whole traversal.

(1) The negotiation begins when Susan Jones requests access to *psychRecord*. The only binding tree at this point contains a single node representing *psychRecord*.

(2) Busey Hospital returns the current traversal (policy 3 and policy 4) of the *psychRecord* resource. The message is $\{(psychRecord, (policy\ 3, policy\ 4))\}$

(3) Susan Jones's strategy determines that her employee ID satisfies policy 4 for *psychRecord*, and discloses the policy for *employeeID*. Since she's using TrustBuilder2-Simple, she also discloses the traversal of *patientID*'s policy graph, the traversal of her *driversLicense*'s policy graph and her unprotected credential *libraryCard*. (The policy graphs for *driversLicense* and *libraryCard* are not shown in the figure). The message Susan Jones sends to Busey Hospital will be $\{extension(\{employeeID\},$

*psychRecord*, (*policy* 3, *policy* 4)), (*employeeID*, *policy* 6), (*patientID*, (*policy* 8, *policy* 9, *policy* 10)), (*driversLicense*, (*policy* 101)), *libraryCard*}. Note that policy 3 and policy 4 are the two sink nodes of *psychRecord*'s current traversal. At this point, Susan's strategy has no way to realize that the extension disclosure will lead to a dead end.

(4) Busey Hospital determines that its institutional ID satisfies policy 6, and discloses *institutionID*. Since *institutionID* is syntactically relevant to *psychRecord* and Susan Jones can expand the resulting set of disclosures, Busey Hospital will not make any irrelevant disclosures. The message only contains the disclosure of *institutionID*.

(5) Susan Jones's strategy determines that *institutionID* satisfies the policy for *employeeID* and for *patientID*, and discloses both credentials.

(6) At this point, Busey Hospital receives Susan Jones's *patientID*, which context satisfies policy 3. The *psychRecord* is unlocked and Busey Hospital grants access to *psychRecord*. Note that Susan Jones cannot tell whether she was granted access to *psychRecord* because it is her own medical record, or because she works for Busey Hospital, because policy 5 was never disclosed to her. If she had used TrustBuilder2-Relevant, policy 5 would have been disclosed, and her strategy would have seen that the disclosure of her *employeeID* led to a dead end.