Obligations in Trust Negotiation

Jeff Puchalski
jpuchalski@mitre.org

Vipin Swarup
swarup@mitre.org

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102

ABSTRACT

Trust negotiation frameworks allow communicating parties to incrementally establish trust in one another to achieve security goals. The goals we focus on deal with sharing sensitive information that is protected by a disclosure policy. Traditionally, such policies are satisfied through the use of signed credentials that express role memberships or attributes. The requirements in these policies are known as provisions, and represent past and present state. Although useful, provisions can sometimes prove too rigid, are susceptible to schema-matching problems, and cannot provide assurances on how information is used once it has been shared.

In this paper, we propose a means of augmenting trust negotiation frameworks to support obligations, which are commitment-based requirements to perform certain actions in the future. We provide a metamodel for such a framework along with a method of converting provisions into sets of obligations. We analyze the complexity of this conversion, and then provide a study of obligation optimality during the negotiation.

1. INTRODUCTION

Trust management enables two or more principals to establish trusted channels (for providing information services), using policies based on attributes such as the principals' identities, authorizations, environments, and communication medium. Sometimes the attributes or policies themselves may be sensitive, and hence require additional policies to regulate how they are exchanged—trust negotiation enables the conditional exchange of policy and attribute credentials so that principals can establish mutual trust in an incremental manner.

In many cases, principals may not be able to establish trust in other principals or environments, but yet need to be able to share critical information and provide information services so that those principals can perform mission-critical operations. Existing trust management and trust negotiation methods are fundamentally inadequate to deal with this need to provide information services in non-trustworthy situations. Firstly, their policies are static and rigid; a party either has the necessary credentials to satisfy the provisions in a policy or it does not. Second, they make no guarantees on the future usage of the information being shared. Once information has been shared, it could be misused by providing it to unauthorized parties that don't have the necessary credentials. A third weakness of provision-based policies is that a party may have the credentials needed to satisfy some policy, but they may be in a form unrecognizable to the other party. Thus, a credential exchange would result in failure.

The key reason for these shortcomings is that existing methods are concerned with protecting information, rather than sharing information. Hence, trust decisions are based on the risks of sharing information and exclude consideration of the risks of not sharing information. While adequate protection of information services is necessary, the failure to provide an information service to the right entity at the right time can be as (or more) disastrous as the inadvertent disclosure of information.

We address these problems in this paper by introducing obligations to trust negotiation. Obligations allow one to specify that certain actions should take place in the future. Another party can then enter into a binding agreement, or commitment, to fulfill those actions within a specified time bound. Our key observation is that parties can substitute obligations for provisions in trust negotiation. That is, when permitted by policy, a party can commit to obligations in lieu of satisfying some provisions. This provides a high degree of flexibility that does not rely on rigidly expressed credentials. Obligations can be constructed (or negotiated) by the involved parties as required, and formulated in such a way that a party who is a complete stranger could make sense of them and take appropriate action.

Obligations provide a weaker form of assurance than provisions because there is no guarantee that their actions will be fulfilled. It is thus not always possible to ensure that obligations are satisfied. However, penalties for violation, along with a suitable enforcement mechanism, can help to mitigate this risk by encouraging behaviors that will satisfy obligations. Further, obligations can be used together with provisions to provide mandated assurance.

In this paper, we develop a metamodel for including obligations in trust negotiation. We then explore ways in which provisions can be dynamically converted to obligations in order to achieve mission goals. We also examine how parties commit to obligations, specifically ways of ensuring that a party does not commit to more obligations than are necessary. Finally, we conclude with a thorough treatment of the complexity and decidability of determining whether an optimal (i.e., minimal) set of obligations has been used to achieve the negotiation goals. To our knowledge, this is the first treatment of obligations with regards to trust negotiation.

Our paper proceeds as follows. We begin in Section 2 by presenting a motivating example for why obligations are needed. Section 3 introduces several key concepts we use, such as trust negotiation, provisions and obligations. In Section 4, we describe a metamodel of a trust negotiation system containing support for obligations, and develop a method of converting provisions to obligations and ensuring that a minimal set of obligations is being used. Section 5 contains our analysis of optimal usage of obligations in trust negotiation. Finally, Section 6 presents related work, and Section 7 concludes.

2. MOTIVATING EXAMPLE

In order to make the case for why obligations are needed in trust negotiation and why provisions should be converted into obligations, we present here a realistic example scenario. Situations similar to this one often arise in the real world whenever there are security or privacy concerns over the disclosure of information, especially between unfamiliar parties. We will revisit this example throughout the paper.

Consider a hypothetical situation where a United States federal agency suspects that a terrorist cell is stockpiling hazardous materials in a factory, some of which might be flammable or combustible. The federal agency is in contact with other agencies, such as the Centers for Disease Control and Prevention (CDC), that have detailed knowledge about the hazardous materials. A fire suddenly breaks out in a warehouse adjacent to the factory, and local firefighters respond to the emergency. The fire grows rapidly and is about to spread to the factory. Meanwhile the firefighters are preparing to enter the warehouse. The CDC has a document that contains detailed information about the hazardous materials in the factory, but the document is classified. Further, the classification guidelines state that in addition to the clearance requirement, a party must have demonstratable need-to-know in order to access the document. This includes being a member of Hazardous Materials Community of Interest (COI) and being at the location where the materials are present. The local firefighters do not have any security clearances, are not members of the COI, and cannot provide verification of their current location. This information is relevant to their current mission because lives, theirs and possibly others, are at stake. Unfortunately, traditional access control methods would disallow release of the document because the firefighters do not satisfy the CDC's hypothetical disclosure policy.

This presents a conundrum. In the real world, people with operational responsibilities are sometimes compelled to violate access control policies if they believe that the benefit of sharing information (e.g., saving lives) outweighs the risk (e.g., disclosing sources and methods). This often gets done ad-hoc without any formal procedure, which can result in a lot of headaches after the fact. It also can result in damaging unchecked information leakages with little to no control over what happens to the information once it gets out.

The goal here is to be able to permit sharing the information to people who have a demonstratable need for it while managing the level of risk incurred by doing so. By just handing out the information freely, the level of risk is clearly high because there are no controls to restrict (a) who can get the information, and (b) what they can do with it once they have it. Traditional access control policies have focused on and done well at satisfying (a), but do not deal with (b). While usage control policies address both, they do not analyze the trade-off between (a) and (b). In the remainder of this paper, we will outline how information sharing can occur by incurring a modest increase in risk in cases where parties, such as the firefighters, cannot satisfy all of the requirements of a static security policy. In doing so, we will also allow parties to dictate how their information gets used after it has been released.

3. CORE CONCEPTS

In this section we introduce the fundamental components of interest. They will be used in the next section to construct a metamodel for sharing information via trust negotiation with obligations.

3.1 Trust Negotiation

In electronic trust negotiation, two parties incrementally establish trust in one another by selectively exchanging digitally signed credentials that contain attribute and role membership information. One common purpose of trust negotiation is to share information between the parties or users involved. The information is usually contained in an electronic medium such as a file or document. We refer to the party requesting the information as the *requester* and the party holding the information as the *provider*.

As long as the information being shared is not sensitive, then no negotiation is necessary to obtain it. In many cases however, the information to be shared is sensitive, and a provider may only want to share it with trusted requesters that satisfy some disclosure policy. An access control (AC) policy consists of rules that specify what roles need to be satisfied in order for sensitive information to be shared. Role memberships are usually satisfied through the use of signed credentials that represent a statement by some authority that a user belongs to a particular role. In a trust negotiation, these signed credentials are exchanged between the parties in order to satisfy rules in disclosure policies and therefore enable sharing of information.

Sometimes the credentials themselves that are being exchanged during a trust negotiation are sensitive. To safeguard their disclosure, AC policies can be used in the same manner as for information. Sometimes, the knowledge of whether a party even has a particular credential or not must be protected. Several types of disclosure policies have been proposed to handle this problem. Parties can use Acknowl-

edgment (Ack) policies [14] to guard knowledge about the presence or absence of sensitive credentials. Requesters will be required to satisfy the Ack policy before the party reveals whether or not they have the sensitive credential. Note that the Ack policy is always presented to the requester regardless of whether a party has the sensitive credential or not. A similar scheme called hidden credentials [8], controls the disclosure of sensitive information by encrypting messages such that only a party who has the necessary credentials can decrypt the message and retrieve the information.

3.2 Provisions

The credentials discussed thus far express memberships in roles that hold at the present time. The role memberships required to satisfy a policy are called *provisions*. Provisions are a representation of the past state of the world since they rely on actions performed or credentials obtained at some prior point in time for satisfaction. Provisions are unambiguous; a party either has or lacks the credentials necessary for satisfaction. Given that there is usually a high level of security inherent to credentials, provisions offer relatively strong assurance.

One key drawback of provisions though is that they must be known to both parties to be effective in the negotiation. Further, each party may have different ways of representing the same provisions, resulting in a schema matching problem that could prevent the negotiation from succeeding. In a structured trust negotiation environment, such problems can be avoided through advance planning, but in an environment where unfamiliar parties attempt to share information, the difficulties faced can cause the process to breakdown. In our firefighters example from earlier, imagine a case where the firefighters are responding to a blaze and request information from a government agency. The agency presents a policy stating that the firefighters must have a security clearance in order to access the information. The firefighters present their clearance credentials, only to get rejected because their clearance was a different type that the agency does not understand or accept.

Another problem with provisions is that they do not allow future data protection requirements to be met. Once information has been shared, there are no controls offered by provisions that regulate its usage. Going back to the firefighters example, let us assume that the information is sensitive but does not require a clearance and that the firefighters are able to obtain it. They then decide to share it with the local police department, the mayor's office, and several news outlets. The government agency almost certainly did not intend on such uses occurring, but could do little to prevent it without some formal means of defining usage control over the information.

3.3 Obligations

Our solution to these problems is to use *obligations* as a companion to provisions in trust negotiation. Obligations specify actions that must occur after information disclosure. More formally, obligations represent a party's *commitment*, or binding agreement, to abide by a temporal policy over future states by satisfying the actions specified by the policy. Obligations can be used alongside provisions in disclosure policies [1]. Future data protection and sharing requirements can thus be encoded as obligations that describe how the information may be used.

Obligations provide flexibility that cannot be achieved with provisions by allowing parties to agree to perform certain actions instead of presenting credentials. In this manner, a party that is qualified to receive information in a trust negotiation can do so even if they cannot satisfy the provisions that would otherwise be necessary. In the firefighters example, if the information is sensitive and requires a clearance, then the agency could waive the clearance requirement by requiring the firefighters to commit to one or more obligations, and possibly some weaker provisions, as an alternative.

3.3.1 Obligations as Two-Sided Agreements

In trust negotiation, a requester would have little motivation to make commitments to obligations without getting something in return. Hence, an underpinning of obligations is that, upon commitment, the requester will receive from the provider the information being protected. Obligations represent a *two-sided agreement* that implicitly places a requirement on the provider to deliver the information being protected upon commitment by another party. Note that sharing of the information needs to occur upon commitment to, not satisfaction of, the obligation. If the information is not shared, then the agreement between the two parties is nullified. The provider does retain first option however, as they do not have to present an obligation to the requester unless they are presently satisfied with the risk involved in doing so. The provider's knowledge about the requester will likely play a part in making this decision.

3.3.2 Assurance and Risk

The primary drawback of obligations is that there are no guarantees that their specified actions will be performed, which could result in violations. By allowing parties to break commitments, obligations offer a weaker form of assurance than provisions. Research has been done to show how commitment violations can lead to a decrease in a quantifiable trust value [4]. It is assumed that the benefit of sharing information outweighs the risk involved in using obligations. Without delving into a deep discussion of this trade-off, we will state that parties can be diligent in attempting to determine if another party is not only able to perform the actions required to satisfy an obligation, but also likely to do so. Given the reduced assurance of obligations, it seems straightforward that given a choice between using provisions or obligations, the former should be preferred as long as the cost to both parties of dealing with provisions and credentials is less than the cost of dealing with obligations and commitments. We expect this to be true in all but a small number of cases.

To increase the likelihood of obligations being satisfied, parties can be given incentives to fulfill their commitments by creating a reward/penalty system where satisfaction of an obligation results in a reward and violation results in a penalty, or *sanction*, being applied. In order for sanctions to hold merit, a suitable method of enforcement must be present. In the context of obligations, there are two primary characteristics to consider when dealing with enforcement: controllability and observability [7]. Obligations that are controllable will always be satisfied because it is not possi-

CDC Policies:		
p_1 : CDC.HazardInfo	$\leftarrow$	CDC.NeedToKnow
p_2 : CDC.NeedToKnow	$\leftarrow$	IAFF.Firefighter ! DOD.Location
p_3 : CDC.NeedToKnow	$\leftarrow$	DOD.Clearance(level >= 'Secret') ! COI.Member(group = 'HazMat')

Figure 1: The policies of CDC

ble for violation to occur. Obligations that are observable can be monitored to tell when they are satisfied or violated, but it may or may not be possible to prevent violation from occurring. Obligations that are neither controllable nor observable offer little assurance and should be avoided. There has been work performed in the area of contract negotiation to build an enforcement framework in which trusted third parties enable negotiating entities to trust that contract commitments will be monitored and enforced [11]. We will not be addressing sanctions or enforcement further in this work, but will instead focus on the way obligations integrate with trust negotiation.

4. TRUST NEGOTIATION WITH OBLIGATIONS METAMODEL

In this section we define a metamodel for trust negotiation that includes support for obligations. Before doing so though, we return to the firefighters example to formally express several characteristics of the scenario. We express the CDC data protection requirements from Section 2 as a set of policy rules shown in Figure 1.

4.1 Components

Several key components can be identified by looking at the policy rules for our example. Here we will describe each of those components in turn. We take a bottom-up approach, starting with the fundamental building blocks: provisions and obligations.

Provisions Specify requirements for membership in roles. Take the form $B.R$ used in the RT_0 language [15], where B is some principal or authority and R is a role. They are satisfied by digitally signed credentials that make authoritative statements about entities. From the policies shown in Figure 1, the single provision of the policy in p_1 is CDC.NeedToKnow, meaning that a party trying to gain access to the CDC's hazardous materials information first needs to demonstrate that they have a need-to-know for that information. As another example, the provision DOD.Location in p_2 means that a party needs to show that they are at a DOD-verifiable location that corresponds to the location of the fire. This role would likely have several attributes that express things like exact street address or GPS coordinates, but we have left them off for simplicity.

Obligations Specify future time-bounded actions, and are satisfied by commitments to fulfill those actions. Take the form ω(*action*, subject, `time-bound`), where the specified *action* must be applied with regards to the subject within the `time-bound` provided. From our example, assume that Fred is a firefighter, but cannot verify or prove his location in order to satisfy the CDC.NeedToKnow role. Further, the CDC has several obligations available that it can use in place of provisions, shown in Figure 2. The CDC converts the provision for the location attribute to obligation o_1, which states that Fred must install monitoring software on his machine in order to satisfy the need-to-know requirement. The time-bound of *now* means that as soon as Fred commits to the obligation he will be expected to fulfill it by performing the action of installing the software. We use shorthand such as `48h` in o_2 to represent a time-bound of 48 hours.

CDC Obligations:
o_1 : ω(*install*, monitoring software, `now`)
o_2 : ω(*submit*, background check, `48h`)

Figure 2: The CDC's available obligations

Our description here of the obligation for monitoring software is a bit terse. We assume that the obligation would come with a more complete description of what systems the software needs to be installed on, what types of behaviors will be monitored, and so forth. Since these details only affect the decision-making process of the parties involved and not the model itself, we have omitted them in this paper.

Policy Rules Represent conditional disclosure of some information based upon the satisfaction of provisions and obligations. The information (or credential) contained in the head of the rule will be released if the provisions and obligations contained in the body are satisfied. We use the syntax of the role-based trust language RT_0 [16, 15] for writing policy rules. We also use the concept of policy preconditions [13] to protect sensitive policies. A policy is written as $D \leftarrow precond$! $B_1.R_1 \cap \ldots \cap B_k.R_k$, where D is the information or role to be granted upon satisfaction of the policy, *precond* is the optional precondition that needs to be satisfied before the policy itself is disclosed, and the $B_i.R_i$ represent roles in the policy body.

Policy p_2 shown in Figure 1 is sensitive and has the precondition IAFF.Firefighter. Therefore, a party must first prove that they are a firefighter before they can see p_2. The body of policy p_2 contains a single role, DOD.Location, that must be satisfied.

Trust negotiations have the intention at each step of sharing some sensitive information. The information is presented as the head of a policy rule that has one or more provisions and obligations in its body. The information could be the negotiation goal, or it could be a sensitive credential, role, or policy. Once the provisions and obligations in the policy are satisfied via credential exchanges and commitments, the information can be shared. This entire agglomeration

CDC Conversion Functions:		
f_1 : IAFF.Firefighter	$\rightarrow$	US.Citizen $\wedge$ o_1
f_2 : DOD.Location	$\rightarrow$	o_1
f_3 : DOD.Clearance(level >= 'Secret')	$\rightarrow$	US.Citizen $\wedge$ o_2
f_4 : COI.Member(group = 'HazMat')	$\rightarrow$	o_1

Figure 3: CDC's provision to obligation conversion functions

consisting of sensitive information, a policy rule, credential exchanges and commitments is called a *decision point*.

Definition 1. We define the set of *decision points* D in a trust negotiation as the points where conditional release of information occurs based upon the satisfaction of rules comprised of provisions or obligations. Each decision point $d \in D$ has a rule that contain one or more provisions $d.p_i$ or obligations $d.\omega_i$.

We model trust negotiation as a *proof tree*, where the nodes represent decision points. The root node of the tree contains the goal in a policy head. For a policy rule at some node, the provisions contained in the rule body can be expanded out into child nodes, where each provision serves as the head of the rule in a child node. Figure 4 shows the initial proof tree for our firefighters example. Thus far, CDC has added the goal and their policies to the tree.

When referencing decision points in a negotiation, we typically refer to the policy nodes that are protecting the information to be released once credentials are exchanged and commitments are made. In our firefighters example, the decision points in the proof tree of Figure 4 occur at the nodes on the second level of the tree containing policies p_1 and p_2.

The proof tree for a trust negotiation could have many possible expansions. The number of possible proof trees is d^b, where d is the tree's depth and b is the node branching factor. When a negotiation actually happens, the proof tree that gets constructed is called a *proof tree instance*. Each proof tree instance represents one possible expansion of the nodes, which is equivalent to saying one possible event trace of the negotiation.

It is worth noting here that any trust negotiation framework, such as the Trust Target Graph (TTG) [25, 14, 13] and TrustBuilder(2)/Traust [22, 12], can be extended with support for obligations in order to implement this metamodel.

4.2 Converting Provisions into Obligations

The initial state of an provider's policy rule is the way it is represented when its decision point is first encountered. The requester always first attempts to satisfy the rule in its initial state. If they cannot, then the provider has the option to change the rule. A policy rule can undergo change by having a provision converted into one or more obligations expressed in disjunctive normal form (DNF). Each time this occurs only a single provision gets converted. Once a change has been made, the requester again tries to satisfy the rule. If they cannot, then more rounds of conversion occur until either the rule is satisfied or no more conversions are possible. The choice of which provision to convert at each turn rests with the provider. It is presumed that one would want to convert the least important provisions first.

We write the conversion function for provisions to obligations in product of sums form as $f : p \rightarrow \Omega_1 + \ldots + \Omega_n$, where $\Omega := o_1 \wedge \ldots o_m$ and o_k is an obligation. Each conjunction of obligations in the resulting DNF formula, or *monom* [24], represents a set of obligations that results in satisfaction of the requirement originally specified by the provision. The requester is free to choose any of the monoms. Figure 3 shows several conversion functions that the CDC can use in our firefighters example to convert various provisions from the policy rules in Figure 1 to obligations from Figure 2.

For the sake of completeness, we assume a partial ordering of obligations $o_i \prec o_j$ where o_j is deemed as more stringent than o_i. This means that it would be beneficial for the requester to choose o_i over o_j. We can extend this notion to sets of obligations $\Omega_i \prec \Omega_j$, where the set Ω_j is more stringent than the set Ω_i. Note that this does not imply $o_a \prec o_b$ for any $o_a \in \Omega_i$ and $o_b \in \Omega_j$. A complex scheme for comparing obligations could be used, but for simplicity in this paper we rank sets of obligations based solely on cardinality:

$$\Omega_i \prec \Omega_j \leftrightarrow |\Omega_i| < |\Omega_j| \quad (1)$$

In our firefighters example, say there is a firefighter named Fred who can satisfy the IAFF.Firefighter role, but cannot prove his location. Therefore, the CDC converts the provision DOD.Location into obligation o_1. Additionally, Fred does not have a security clearance, so the provision DOD.Clearance(level >= 'Secret') gets converted into the provision US.Citizen and obligation o_2. Finally, Fred is not a member of the Hazardous Materials Community of Interest (COI), so the role COI.Member(group = 'HazMat') gets converted into obligation o_1. Figure 5 shows what the proof tree looks like after these conversions have been made.

4.3 Minimizing Obligation Formulae

The policy rule at a decision point can have multiple provisions, each of which can be converted to a different DNF formula. This can result in redundancies and inefficiencies because it allows the requester to satisfy more obligations that may be necessary to satisfy the rule. To alleviate this problem, we present an algorithm that can be used to find the minimal set of obligations needed to satisfy the policy.

Imagine a trust negotiation is in progress between two parties. A decision point is reached that initially has a policy rule $g \leftarrow p_1 \wedge p_2 \wedge p_3$, where g is the goal and the p_is

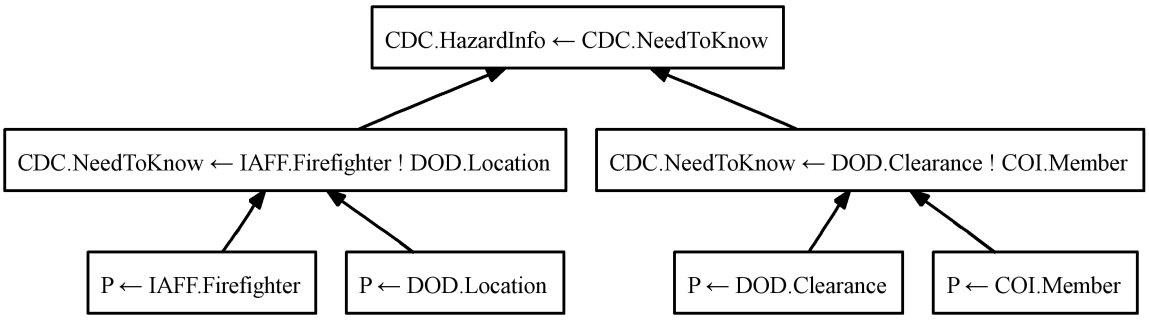

Figure 4: Initial proof tree for CDC example

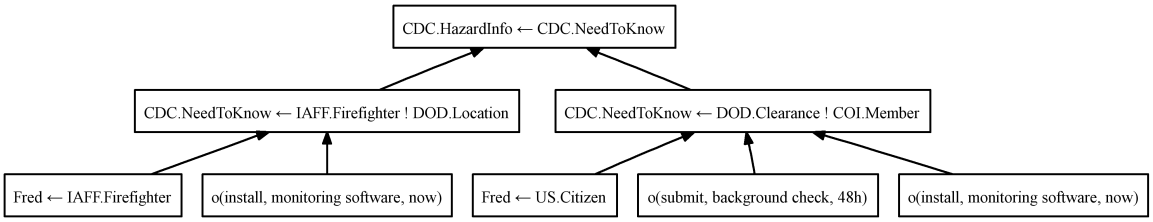

Figure 5: Proof tree for CDC example after conversions

are provisions. Assume that the provider converts p_1 and p_2 into obligation formulae using the conversion function $f : p \rightarrow \Omega_1 + \ldots + \Omega_n$ as follows:

$$\begin{aligned} f_1 : p_1 &\rightarrow o_1o_2o_3 + o_2o_4o_5 + o_1o_4o_5 \\ f_2 : p_2 &\rightarrow o_1o_6 + o_1o_4o_8 \end{aligned}$$

In order to determine the minimal set of obligations, we first need to take the cross product of all the formulae. This generates all possible combinations of obligations. Taking the cross product of f_1 and f_2 yields the following:

$$\begin{aligned} f_1 \times f_2 = o_1o_2o_3o_6 + o_1o_2o_4o_5o_6 + o_1o_4o_5o_6 + \\ o_1o_2o_3o_4o_7 + o_1o_2o_4o_5o_7 + o_1o_4o_5o_7 \end{aligned}$$

The next step is to simplify this formula. In this case we can take advantage of the fact that the second and third monoms and fifth and sixth monoms only differ by one term, so using the cardinality ordering from (1) above we can throw out the second and fifth monoms because they have an extra term. We can then combine like terms and eliminate one monom by the cardinality ordering to end up with the following:

$$f_1 \times f_2 = o_1o_2o_3o_6 + o_1o_4o_5o_6 + o_1o_4o_5o_7$$

In more complex cases, a variation of the Quine-McCluskey (Q-M) algorithm [19, 17], also known as the method of prime implicants, can be used for function minimization. The Q-M method has been shown to have a runtime of $O(N^{\log_2 3} \log_2^2 N)$, where $N = n^2$ and n is the number of variables [24]. Taking the cross-product in the first step of this minimization algorithm takes $O(m^n)$, where m is the number of rules and n is the number of obligations. This runtime dominates the runtime of the Q-M algorithm, so the upper bound on the runtime of the entire algorithm is $O(m^n)$;

5. OBLIGATION OPTIMALITY

The minimization performed in the previous section is useful because it prevents a requester from committing to more obligations than are necessary to satisfy a policy. Here we will formalize this notion of minimality by introducing the concept of optimality.

Definition 2. A decision point is *optimal* if the minimum set of obligations is used that allow its rule to be satisfied.

We assume that a minimal set is selected based on the cardinality ordering of obligations shown in (1) in Section 4.2. Optimality also minimizes the level of risk to the provider because, in order to achieve optimality, provisions are converted to obligations one at a time. Another way to describe optimality is to say that there is no smaller set of obligations that would allow the negotiation to make progress.

THEOREM 1. *Given a decision point that converts provisions $p_1, \ldots, p_n$ to obligations $o_1, \ldots, o_n$ according to the function $f : p \rightarrow \Omega_1 + \ldots + \Omega_m$, the problem of computing the optimal solution is decidable with complexity $O(m^n)$.*

PROOF. Sketch. Using the algorithm from Section 4.3, we can construct the minimum set of obligations for a policy rule, which by definition results in the optimal solution for a decision point. Taking the cross product of m rules

with n terms has complexity $O(m^n)$, which dominates the complexity of Q-M. □

It is far more ambitious to determine optimality over the entire proof tree (i.e., across all decision points) for a negotiation. It is in fact possible to achieve optimality at each individual decision point and not have optimality over the entire proof tree. The primary difficulty in attaining optimality over the entire tree is caused by the *horizon problem*. In this case, the horizon results from not being able to "'see"' unexpanded nodes further down the tree from any given state. Without such lookahead knowledge, it becomes impossible to determine the correct selection of a minimal set of obligations at the current node's decision point.

Imagine having a choice between two or more equally minimal sets of obligations at a decision point. Without any other knowledge, the choice seems to be arbitrary and irrelevant. However, if one of the choices has an intersection with a set of obligations from a node that is later expanded, then that choice would be the correct one in order for proof tree optimality to be possible.

THEOREM 2. *During a trust negotiation, the problem of determining whether the proof tree instance is optimal is undecidable.*

PROOF. Sketch. We will use our firefighters example to show the horizon problem. Assume that the proof tree for our negotiation looks as it does in Figure 5, except that Carol, who is a citizen but not a firefighter, is on the scene instead of Fred. Since Carol is not a firefighter, she does not satisfy the IAFF.Firefighter provision. Therefore, the CDC uses conversion function f_1 from Figure 3 to convert the requirement into the provision and obligation pair US.Citizen $\wedge$ o_1. At this point Carol can either satisfy US.Citizen $\wedge$ o_1 on the left branch or US.Citizen $\wedge$ o_2 on the right branch. However, if she chooses the latter and commits to o_2, she will eventually also have to commit to o_1, resulting in one more obligation than was necessary and a sub-optimal proof tree. The CDC will not know in advance what provisions need to be converted, and so they are unable to prevent this situation from occurring. Because neither party can "'see"' the nodes further down the tree that are not yet expanded, they cannot determine what choices should be made in order to achieve optimality. □

Once the negotiation has terminated either in success or failure, the proof tree will not be expanded further. Given complete knowledge of the nodes that were expanded, it becomes possible to determine if proof tree optimality was achieved.

THEOREM 3. *After a trust negotiation has concluded, it is decidable to determine if optimality was achieved for the proof tree instance.*

PROOF. Sketch. Going back to our firefighters example with Carol, assume that we have a proof tree instance from a completed negotiation in which Carol committed to both o_1 and o_2. This proof tree instance will not be expanded further. Assuming that the minimization algorithm from Section 4.3 was applied during the negotiation to achieve optimal efficiency at each decision point, we can now take the minimized formula from each decision point and re-apply the minimization algorithm to determine the optimal solution, QED. □

After the negotiation, both parties will have a final version of the proof tree instance that was generated. If the parties see that an optimal solution was not reached (e.g., due to the horizon problem), they may be able to renegotiate the set of obligations that were committed to in order to achieve optimality. In Carol's case, both she and the CDC will know after the negotiation that she could have just used o_1, so they could potentially renegotiate the terms in order to nullify o_2.

6. RELATED WORK

Trust negotiation and obligations have both been well studied, albeit independently, in the literature. Hilty et al. [7] present a framework, based on Distributed Temporal Logic (DTL), for expressing obligations. In particular, they focus on temporal and observability aspects in developing a classification system for obligations, including a method of transforming non-observable obligations into observable ones. Irwin et al. [9] present a metamodel for obligations and define a notion of accountability based on secure system states. In this paper, we have modeled an obligation as a set of actions in a bounded time range. Bettini et. al. [1] make a distinction between provisions and obligations in access control policies—the trust negotiation literature has focused on access control policies involving provisions, whereas in this paper, we consider access control policies with both provisions and obligations.

Winslett, Yu, et. al. [26] and Smith et. al [22] developed a system called TrustBuilder, which is an implementation of a trust negotiation framework that includes a description language for writing policy rules and provides an automated compliance checker that can determine satisfying sets of credentials. In [25] Winsborough et al. present two automated trust negotiation (ATN) strategies, one of which is a parsimonious strategy that aims to achieve a successful negotiation with a minimal exchange of credentials. Our work complements this approach in that we use obligations to replace provisions in cases where the necessary credentials are not available. Winsborough, Li, et. al. [14, 13] designed the (Extended) Trust Target Graph ([E]TTG) protocol and ATNL specification language to represent trust negotiation as a tree structure with typed nodes and edges. They leverage earlier work on a role-based rule language called *RT* [16, 15], which we also use in this paper. They also introduce Acknowledgment policies as a method of handling disclosure of sensitive information.

Yu and Winslett introduce policy migration [28] to protect sensitive credentials by integrating the policy of a sensitive credential into the policies of other credentials so that a user's behavior will be the same regardless of whether they have the sensitive credential. Yu et. al. [29, 30] present a

formal model of negotiation protocols and strategies, with a focus on allowing two parties with different strategies to communicate. They also discuss ways of negotiating in the presence of sensitive policies and credentials. Seamons et. al. [20] discusses ways to dynamically modify policy statements to prevent sensitive information leakage from probing attacks.

Techniques have been proposed for integrating measures of opponent trust into negotiation. Sierra and Debenham [21] define a negotiation model and language in which negotiating entities can employ trust measures into their negotiation strategy. Wooldridge and Parsons [27] present a formal language for negotiations and shows how negotiation protocols written in this language can be evaluated to show that agreement can be both identified when it occurs and guaranteed to occur. Hanson and Milosevic [6] define a number of negotiation patterns in terms of conversation policies that are suitable for automated negotiation by agents.

7. CONCLUSION AND FUTURE WORK

In this paper, we have shown that obligations are a valuable companion to provisions in trust negotiation. Provisions and obligations each have their strengths and weaknesses, but without obligations one loses not only flexibility, but also the ability to offer information usage assurances in the future. We have shown a way in which a provider's provisions can be converted into obligations in a way that offers more flexibility by providing multiple options to the requester, and how to minimize the options presented so as to avoid spurious obligations from being necessary. Finally, we have analyzed the complexity and decidability of determining whether various parts of the trust negotiation are optimal with regards to the usage of obligations.

As a continuation of our work, we would like to create a better model of obligations that could potentially automate some or all of the commitment process. We anticipate that this could include negotiating the terms of the obligations themselves, such as their actions, time bounds, violation penalties, and enforcement mechanisms. We would also like to explore the enforcement piece in greater detail to develop better methods of dealing with obligations that are monitorable but not enforceable. Another area to focus on is formalizing the ordering of obligations. In this paper we assumed that one exists, and for sets of obligations we used cardinality, but it would be useful to have a more tangible way of comparing two obligations.

8. REFERENCES

[1] Claudio Bettini, Sushil Jajodia, Xiaoyang Sean Wang, and Duminda Wijesekera. Provisions and obligations in policy rule management. *Journal of Network and Systems Management*, 11(3):351–372, 2003.

[2] Guido Boella and Leendert van der Torre. A game theoretic approach to contracts in multiagent systems. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews.*, 36(1):68–79, January 2006.

[3] Guido Boella and Leendert W. N. van der Torre. Negotiating the distribution of obligations with sanctions among autonomous agents. In de Mántaras and Saitta [5], pages 13–17.

[4] Jan Broersen, Mehdi Dastani, Zhisheng Huang, and Leendert W. N. van der Torre. Trust and commitment in dynamic logic. In *EurAsia-ICT '02: Proceedings of the First EurAsian Conference on Information and Communication Technology*, pages 677–684. Springer-Verlag, 2002.

[5] Ramon López de Mántaras and Lorenza Saitta, editors. *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*. IOS Press, 2004.

[6] James E. Hanson and Zoran Milosevic. Conversation-oriented protocols for contract negotiations. In *EDOC '03: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing*, page 40. IEEE Computer Society, 2003.

[7] Manuel Hilty, David Basin, and Alexander Pretschner. On obligations. In *10th European Symposium on Research in Computer Security (ESORICS 2005)*, volume 3679 of *LNCS*, pages 98–117. Springer-Verlag, 2005.

[8] Jason E. Holt, Robert W. Bradshaw, Kent E. Seamons, and Hilarie Orman. Concealing complex policies with hidden credentials. In *CCS '04: Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[9] Keith Irwin, Ting Yu, and William H. Winsborough. On the modeling and analysis of obligations. In *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 134–143. ACM Press, 2006.

[10] David Keppler, Vipin Swarup, and Sushil Jajodia. Redirection policies for mission-based information sharing. In *SACMAT '06: Proceedings of the 11th ACM Symposium on Access Control Models and Technologies*, pages 210–218. ACM Press, 2006.

[11] Martin J. Kollingbaum and Timothy J. Norman. Supervised interaction: creating a web of trust for contracting agents in electronic environments. In *AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 272–279. ACM Press, 2002.

[12] A. Lee, M. Winslett, J. Basney, and V. Welch. Traust: a trust negotiation-based authorization service for open systems. In *SACMAT '06: Proceedings of the Eleventh ACM Symposium on Access Control Models and Technologies*, 2006.

[13] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In *CCS '05: Proceedings of the 12th ACM Conference on Computer and Communications Security*, pages 46–57. ACM Press, 2005.

[14] N. Li and W. Winsborough. Towards practical automated trust negotiation. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 92–103. IEEE Computer Society, 2002.

[15] Ninghui Li and John Mitchell. A role-based trust-management framework. In *Third DARPA*

Information Survivability Conference and Exposition (DISCEX III), pages 201–, 2003.

[16] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society, 2002.

[17] E. McCluskey. Algebraic minimization and the design of two-terminal contact networks. *Bell System Technical Journal*, 35:1417–1444, 1956.

[18] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed usage control. *Communications of the ACM*, 49(9):39–44, 2006.

[19] W. V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59:521–531, 1952.

[20] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, and J. Jacobsen. Protecting privacy during on-line trust negotiation, 2002.

[21] Carles Sierra and John Debenham. An information-based model for trust. In *AAMAS '05: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 497–504. ACM Press, 2005.

[22] B. Smith, K.E. Seamons, and M.D. Jones. Responding to policies at runtime in trustbuilder. *Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pages 149–158, 2004.

[23] Vipin Swarup, Len Seligman, and Arnon Rosenthal. Specifying data sharing agreements. In *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 157–162, 2006.

[24] Ingo Wegener. *The Complexity of Boolean Functions*. B. G. Teubner, and John Wiley & Sons, 1987.

[25] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. *DARPA Information Survivability Conference and Exposition*, 1:88–102, 2000.

[26] Marianne Winslett, Ting Yu, Kent E. Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.

[27] M. Wooldridge and S. Parsons. Languages for negotiation. In *Proceedings of the Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*. John Wiley & Sons, 2000.

[28] Ting Yu and Marianne Winslett. Policy migration for sensitive credentials in trust negotiation. In *WPES '03: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 9–20. ACM Press, 2003.

[29] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *ACM Conference on Computer and Communications Security*, pages 146–155, 2001.

[30] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, 2003.