2005-08-29

Challenging Policies That Do Not Play Fair: A Credential Relevancy Framework Using Trust Negotiation Ontologies

Travis S. Leithead
Brigham Young University - Provo

CHALLENGING POLICIES THAT DO NOT PLAY FAIR: A CREDENTIAL RELEVANCY FRAMEWORK USING TRUST NEGOTIATION ONTOLOGIES

by

Travis Scott Leithead

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

August 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Travis Scott Leithead

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Kent E. Seamons, Chair
Date	David W. Embley
Date	Phillip J. Windley

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Travis Scott Leithead in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Kent E. Seamons Chair, Graduate Committee

Accepted for the Department

Parris Egbert
Graduate Coordinator

Accepted for the College

G. Rex Bryce
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

CHALLENGING POLICIES THAT DO NOT PLAY FAIR: A CREDENTIAL RELEVANCY FRAMEWORK USING TRUST NEGOTIATION ONTOLOGIES

Travis Scott Leithead

Department of Computer Science

Master of Science

This thesis challenges the assumption that policies will "play fair" within trust negotiation. Policies that do not "play fair" contain requirements for authentication that are misleading, irrelevant, and/or incorrect, based on the current transaction context. To detect these unfair policies, trust negotiation ontologies provide the context to determine the relevancy of a given credential set for a particular negotiation.

We propose a *credential relevancy framework* for use in trust negotiation that utilizes ontologies to process the set of all available credentials C and produce a subset of credentials C' relevant to the context of a given negotiation. This credential relevancy framework reveals the credentials inconsistent with the current negotiation and detects potentially malicious policies that request these credentials. It provides a general solution for detecting policies that do not "play fair," such as those used

in credential phishing attacks, malformed policies, and malicious strategies.

This thesis motivates the need for a credential relevancy framework, outlines considerations for designing and implementing it (including topics that require further research), and analyzes a prototype implementation. The credential relevancy framework prototype, analyzed in this thesis, has the following two properties: first, it incurs less than 10% extra execution time compared to a baseline trust negotiation prototype (e.g., TrustBuilder); second, credential relevance determination does not compromise the desired goals of trust negotiation—transparent and automated authentication in open systems. Current trust negotiation systems integrated with a credential relevancy framework will be enabled to better defend against users that do not always "play fair" by incorporating a credential relevancy framework.

ACKNOWLEDGMENTS

This work is made possible thanks to great sacrifice by my wife Erin for proofing and all her support. Special thanks to Mark Vickers for his help, references, and insight into the ontology world, and to Timothy W. van der Horst, Kent Seamons, and members of the Internet Security Research Lab for encouragement and friendship.

Table of Contents

1 Introduction **1**

1.1 Trust negotiation paradigms 4

1.2 Challenging the assumption 6

1.2.1 Irrelevant credentials scenario 7

1.2.2 Credential phishing scenario 9

1.2.3 Malicious strategy scenario 11

1.3 Solution Overview 12

2 Relevancy Determination Obstacles **15**

2.1 Context 15

2.1.1 Available contexts 15

2.1.2 Current contexts are too narrow 16

2.2 User discretion 19

2.2.1 Users uncertain of relevance 19

2.2.2 User discretion prohibitive in complex negotiations 19

2.2.3 A violation of automation and transparency principles 20

2.2.4 A uniquely client-side approach 21

2.3 Negotiation context location 21

2.4 Universal agreement 24

3 Introduction to Ontologies **25**

3.1 Understanding ontologies 25

3.2 Using ontologies in trust negotiation 29

4 The Credential Relevancy Framework 33

4.1 Overview 33

4.2 Credential relevancy in trust negotiation 33

4.2.1 Irrelevant or malicious policy detection 34

4.2.2 Predisposed negotiation 36

4.2.3 Relevant credentials first 37

5 The Ontology Context Channel 41

5.1 Trust negotiation ontology structure 41

5.1.1 Inheritance and Subsumption 43

5.2 Channel availability 44

5.3 Context alignment 45

5.3.1 User interaction 46

5.3.2 Content extraction 46

5.3.3 Semantic web 47

5.4 Example summary 47

6 The Decision Module 49

6.1 Mapping methods 49

6.2 Handling irrelevance 50

6.2.1 Providing feedback 52

7 Implementation 53

7.1 Fundamental systems 53

7.1.1 TrustBuilder 53

7.1.2 Protégé 55

7.1.3 Jena 55

7.2 Design decisions 58

8 Analysis 61

8.1 Scenario setup 61

8.2 Results 64

8.3 Execution time profile 66

8.4 Generalizations 69

9 Conclusions and Future Work 71

9.1 Contributions 72

9.2 Future work 73

9.2.1 Maturing technologies 73

9.2.2 Aspects of the credential relevancy framework 74

References 83

A OWL Ontology Source 85

A.1 Header information 85

A.2 Supplementary attribute ontologies 85

A.2.1 HospitalStaff 85

A.2.2 University 86

A.2.3 Certification 87

A.2.4 Provider 96

A.2.5 USGovernmentDepartment 96

A.2.6 PoliticalDivisions 97

A.2.7 Payment . . . 97
A.2.8 PersonalInformation . . . 100
A.2.9 Store . . . 102
A.3 Trust negotiation ontologies . . . 102
A.3.1 BusinessDiscount . . . 102
A.3.2 MedicalInformationExchange . . . 104
A.3.3 DMV . . . 105

B TPL Policies 109

B.1 DMV negotiation server policy . . . 109
B.2 DMV negotiation client policy . . . 112
B.3 Medical records negotiation server policy . . . 113
B.4 Medical records negotiation client policy . . . 115
B.5 Bookstore negotiation server TPL policy . . . 115
B.6 Bookstore negotiation client TPL policy . . . 115

List of Tables

1.1 Current security techniques are based on what you know, what you have, and who you are. 1

1.2 Credentials potentially collected by the DMV for a driver license renewal. 8

2.1 The pros and cons of four configurations for negotiation context source. 22

3.1 Summary of ontology features. 30

7.1 The RDQL query that retrieves attributes from a DriverLicenseRenewal. 58

7.2 The prototype design decisions for creation of the credential relevancy framework extension to TrustBuilder. 59

8.1 Comparison of total execution time for TrustBuilder with and without utilizing the credential relevancy framework (CRF). 65

List of Figures

1.1 A simple trust negotiation example. 3

2.1 Areas of context in a classical trust negotiation. 17

2.2 Four examples of current trust negotiation policy languages. 20

2.3 Four configurations for locating the negotiation context source. 23

3.1 Two illustrative ontologies. 26

4.1 An overview of the credential relevancy framework. 34

4.2 The credential relevancy framework's primary application. 35

4.3 An illustrative example of predisposed negotiation. 37

4.4 An example of hidden credential decryption—the third application of the credential relevancy framework. 38

5.1 An illustrative DMV trust negotiation ontology. 42

7.1 The TrustBuilder authentication agent. 54

7.2 Protégé 3.0 with the OWL plugin. 56

7.3 The class hierarchy of the prototype trust negotiation ontologies. . . . 57

8.1 Three baseline scenarios and their negotiation progress. 63

8.2 The profiling architecture. 64

8.3 Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the DMV scenario. 67

8.4 Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the medical records scenario. 67

8.5 Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the bookstore scenario. . . . 68
8.6 Averages over the DMV, medical records, and bookstore scenarios for each of the five modules in TrustBuilder. 68
8.7 Trend graph showing decreasing execution time for negotiations of increased complexity. 69

B.1 The negotiation progress for the semi- and none-relevant cases in the DMV scenario. 110
B.2 The negotiation progress for the semi- and none-relevant cases in the medical records scenario. 114
B.3 The negotiation progress for the semi- and none-relevant cases in the bookstore scenario. 116

Chapter 1 — Introduction

The Internet has made interactions between people easier, faster and closer than ever before. However, increased automation and information access often neglects security, an integral aspect of interpersonal transactions. As transactions move into the digital domain, security measures must be in place to give people the peace of mind necessary to have confidence when interacting online.

Many security techniques are in place today (refer to Table 1.1). All of these have some basis in human trust (a connection to an individual or group of people) and a vehicle for conveying that trust. Most of the systems that implement these techniques are *closed*—meaning that in order to participate, a centralized authority requires registration. Closed systems are analogous to living in a gated community—arrivals and departures are authenticated because the member is recognizable at the gate. While closed systems are ideal for many applications, they do not scale very effectively—especially to Internet-wide proportions—nor are they suitable for applications desiring to customize their services for users possessing specific credentials. For such services, security in an open system is desirable.

Online security techniques

Basis for trust	**Vehicle of trust**	**Example systems**
What you know	shared secrets, passwords, personal information	username/password
What you have	keys, digital certificates, smart cards, tokens	role-based access control, Kerberos [21], Keynote [4]
Who you are	biometrics (fingerprint, eye, voice recognition), credential attributes	biometric scanners

Table 1.1: Current security techniques are based on what you know, what you have, and who you are.

One relatively new approach to open system security attempts to emulate the social method of establishing trust. It is based on observing that people who have never met frequently invest a small amount of initial trust in the other by sharing relevant information about themselves in an effort to foster a mutual increase in trust and confidence. In conversation with a stranger, for example, people often build a foundation of trust with non-essential conversation before addressing more sensitive subjects. Some interactions require only a small amount of trust. For instance, in an exchange with a supermarket cashier, simply swiping a credit card or writing a check is sufficient to form a trusting relationship because the supermarket trusts the credit card issuer or bank. In more expensive or sensitive exchanges, the supermarket may want additional proof of identity or may impose strict check policies requiring further information exchange.

The types of personal information mentioned often take the form of credentials. Generally, a credential is a document containing assertions of attributes or properties about a person and is endorsed by a third party. Common credentials include credit cards, driver's licenses, passports, auto insurance, diplomas, etc. When two people trust a common third party, by extension they should also trust the assertions made by that party.

Digital credentials are the online analogue of physical credentials and serve as a building block for establishing trust in open systems. A trusted third party creates a digital credential by placing attributes about a subject into a document and cryptographically signing that document in such a way that preserves its integrity and authenticity [13]. Digital credentials have been implemented using X.509v3[1] [17] and signed XML [29] standards.

[1]X.509 certificates did not support the addition of user-defined attributes until version 3.

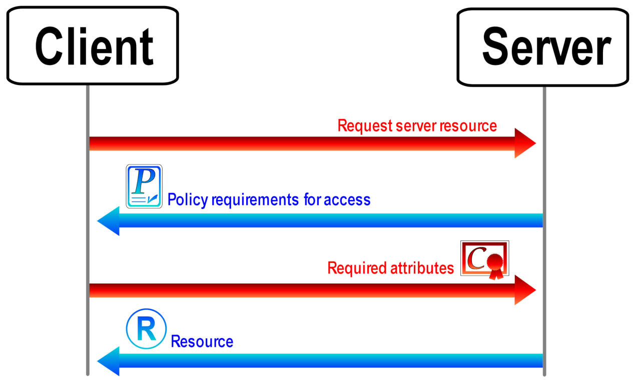


Figure 1.1: Simplified overview of a policy and credential exchange in a negotiation.

When building trust online using digital credentials, a naïve method involves disclosing all non-sensitive credentials at once in hopes of providing the required credentials to the other party. In the supermarket example, this is equivalent to removing the sensitive credentials and then giving the remaining contents of a wallet to the cashier. Obviously, more discretion should be taken. Access control policies are documents that allow both parties to indicate the specific credentials that should be presented in an exchange [32]. Using policies, each participant may indicate the credentials required to build a sufficient level of trust in order to access a resource or complete a transaction.

With digital credentials and policies, two entities may engage in a negotiation for a desired resource. For example, a negotiation might proceed as illustrated in Figure 1.1. The client requests access to a resource. The server responds with its policy. To satisfy the requirements of the server's policy, the client must prove that it possesses the required attributes as certified in digital credentials. The client evaluates the policy, selects the necessary credentials to release, and discloses them to the server, whereupon the server trusts the client sufficiently to provide the requested resource.

The process of exchanging credentials and policies as previously described is known as *trust negotiation*. Various trust negotiation systems exist [2, 3, 5, 19, 24, 26, 34, 39], each with differing policy languages and credential syntax. Using trust negotiation, entities in open systems build trust gradually until they have sufficient confidence to perform online transactions.

Trust negotiation does not always involve an iterative exchange of messages as shown in Figure 1.1. One recent approach to trust negotiation involves using *hidden credentials* to perform an entire negotiation in a single round [6, 8, 15] without the need to disclose credentials or policies explicitly. Section 4.2.3 illustrates an example of trust negotiation using hidden credentials—otherwise, the remainder of this thesis presents trust negotiation using the iterative exchange model due to its understandability and simplicity.

1.1 Trust negotiation paradigms

Trust negotiation research explores the use of automated and transparent negotiation agents, which represent a user's preferences. The goals of automation and transparency allow trust negotiation agents to be seamlessly integrated into a ubiquitous computing world.

Automated trust negotiation implies that an agent makes credential and policy release decisions automatically and that the entire authentication process proceeds without any manual input. To facilitate this, users obtain credentials and define policies for their own protected resources. Policies may also be defined to protect other sensitive policies and credentials. In addition, a user must select an appropriate negotiation strategy [43] that contains the logic for unlocking and releasing policies and credentials according to the user's preferences. Other considerations for automated operation will not be covered in this thesis, including interoperable strategies [24], adequate privacy protection [31, 33], credential and policy storage and discovery [32]. Overcoming the difficulties inherent in automated operation remains an open research question.

The second goal of trust negotiation allows authentication to occur transparently to users. Transparency suggests that the user is unaware that a trust negotiation is occurring. Ideally, after requesting a protected resource, the user notices only a minimal delay while the authentication protocol runs "behind the scenes."

To meet these automation and transparency goals, current trust negotiation systems rely exclusively upon a negotiation strategy and access control policies to guide negotiations to a successful conclusion.

A negotiation strategy is similar to an automated bargaining strategy. Negotiation strategies evaluate policies and select the relevant credentials that satisfy them. The negotiation strategy also determines how credentials will be released. Two disparate strategies include the *eager* and *parsimonious* strategies [38]: the eager strategy releases all non-sensitive credentials every round of negotiation; the parsimonious strategy only releases a minimal set of credentials that satisfy access control policy requirements. Further research has proven that *safe* and *complete* negotiation strategies allow a negotiation to succeed if possible [31, 35].

Access control policies describe the requirements necessary for release of sensitive information or resources. Many varieties of access control policies have been explored in the literature [2, 3, 5, 6, 14, 19, 26, 34, 36]. Requirements for trust negotiation policies are outlined by Seamons et al. [32]. Essentially, a policy asks for the digital credentials containing specific attributes about a user. Trust negotiation halts if a policy cannot be satisfied [35].

Negotiation strategy and policy language research make a significant assumption: that malicious entities will not tamper with or modify strategies and policies.

However, for trust negotiation systems to reach their full potential—deployment in a ubiquitous computing world—this assumption must be challenged.

According to Seamons et al. [31], current negotiation strategies assume that both participants "bargain in good faith." In other words, pre-packaged strategies will not be tampered with or modified from their original form and trust negotiation agents will adhere to certain ethical guidelines. They suggest certification by an appropriate inspection service as a way to enforce bargaining in good faith. However, a certified negotiation strategy is insufficient protection from malicious entities with complete control of a trust negotiation agent.

Policies offer a more vulnerable avenue for attack by malicious entities. Negotiation agents currently assume that a policy authoritatively defines the credentials relevant to completing a transaction. This assumption is valid if the other negotiating party can be trusted not to alter or author malicious policies. As will be shown in the next section, malicious policies have a disastrous effect on trust negotiation.

1.2 Challenging the assumption

Seamons et al. [33] briefly mention the problems that could occur when policies received from a negotiating partner are assumed to be relevant:

> During trust negotiation, one [or both] of the parties may request credentials that are not absolutely necessary or relevant to the trust requirements of the transaction, even though that party may be entitled to see them. A policy could require the disclosure of an inordinate amount of information during a trust negotiation, beyond what is truly needed to protect its resource. An unscrupulous party could use seemingly legitimate credential requests to gather extraneous information, thus violating the privacy of the other party.

The authors conclude that a secure channel is necessary to prevent a malicious eavesdropper from modifying policies in transit, and that digital signatures are necessary to ensure policy integrity on disk. However, these solutions overlook a critical fact; a negotiator may purposefully author a malicious policy and digitally sign it.

The rest of this chapter examines the problems caused by malicious strategies and policies in greater depth. These include the privacy violations mentioned previously by Seamons et al. (i.e., the collection of unnecessary or irrelevant information about a user), as well as the devious collection of sensitive information (i.e., credential phishing), and the malicious tampering of negotiation strategies, potentially leading to denial of service. Each problem is illustrated by example within a scenario that concretely illustrates the potential risks of policies that do not "play fair."

1.2.1 Irrelevant credentials scenario

Some policies enable the broadest set of possible satisfying credentials to grant access to a resource. However, all of the credentials requested may not be relevant to the specific negotiation. For example, when renewing a driver's license from the Department of Motor Vehicles (DMV), various forms of acceptable personal identification may be collected. Table 1.2 contains a set of attributes required by a fictional DMV policy—all of which the DMV is authorized to collect[2].

In this scenario, Pat sends a request to the online DMV requesting a driver license renewal. The DMV server replies to Pat's request by starting a trust negotiation and returning a policy containing requirements similar to those listed in Table 1.2. Pat's trust negotiation agent evaluates the policy and returns the relevant credentials. Pat is disabled and possesses a permit to carry a concealed weapon—credentials that the trust negotiation agent discloses as proof of identity in order to satisfy the

[2]All attributes listed in Table 1.2 were collected from actual state or county DMV licensing requirements (see `http://www.onlinedmv.com`).

Fictional DMV Policy

Please provide two of the following identity proofs		
Social Security Card	State ID	Insurance
US Birth Certificate	Employee ID	CDL Certificate
Vehicle Title	Vision Exam	Vehicle Registration
I/M Test Results	Disabled Status	Pilot's License
Safety Test Results	Immigration Certificate	Passport
Adoption Certificate	Certified Name Change	GED or equivalent
Certified Marriage License	Canadian Birth Certificate	ID Card issued by the DoD
Certificate of Naturalization	Permanent/Resident Alien Status	Permit to carry a concealed weapon or firearm
Certificate of completion of drivers training course		

Table 1.2: Credentials potentially collected by the DMV for a driver license renewal.

policy. The driver license renewal negotiation policy requires only *minimal* (two) proofs of identity and *any* two proofs will do. While the DMV server is authorized to accept all these forms of proof, Pat may not want knowledge of the disabled status and concealed weapons permit credentials unnecessarily proliferated on the Internet, especially when other more common credentials will satisfy the policy. In this case, the decision of which credentials to release was arbitrary.

Due to the complexity and expressiveness of policy languages [2, 14, 24, 26, 36], it is easy to imagine that policies can be written in error. Because most policy languages contain logical constructs (e.g., conjunctions and disjunctions) for grouping credentials, sets of *required* credentials can erroneously be assigned as *optional* and vice-versa. The DMV policy expressed previously could easily have contained an error specifying that Pat must release *all* of the requested credentials rather than any *two* of them. One simple error could lead to irrelevant credential requests. Because most policy creation and modification is performed manually, policies are highly error-prone.

Whether the DMV policy was written in error or not, Pat's trust negotiation agent was unable to recognize that it was releasing extraneous or irrelevant credentials based only on the policy received. If their detection were possible, Pat might have prevented, altered, or justified their automatic release and provided valuable feedback to the DMV server, allowing them to correct their policy.

Pat's privacy concerns are justified because disclosed information is often harvested to analyze buying patterns, collect demographic information, buy or sell personal information, etc. This scenario considers the release of irrelevant information to a server that is justified in obtaining it. The next scenario presents a more potentially damaging irrelevant policy.

1.2.2 Credential phishing scenario

If policies can be written to petition any attribute, then policies in the hands of malicious entities can be disastrous. One strategy used by a malicious entity is to set up a server that looks and feels the same as a legitimate web server. This technique, known as a *phishing attack*[3], extends to trust negotiation where the user is much more vulnerable.

Lynn requests medical records from a HIPAA-compliant[4] hospital website. Unfortunately, Lynn has actually requested information from an impersonating phishing website. The malicious website hosts a trust negotiation phishing service designed to steal credit card information. After Lynn requests the medical records, the malicious trust negotiation agent returns a carefully-crafted policy requesting

[3] `http://www.antiphishing.org` reports that "Phishing attacks use 'spoofed' e-mails and fraudulent websites designed to fool recipient into divulging personal financial data such as credit card numbers, account usernames and passwords, social security numbers, etc. By hijacking the trusted brands of well-known banks, online retailers and credit card companies, phishers are able to convince up to 5% of recipients to respond to them." Policies designed for phishing attacks are a natural extension to current phishing schemes.

[4] HIPAA law (2004) states that costs for copies of personal medical information should only cover the cost of materials, postage, and prepared summary forms, and that entities "may not charge any fees for retrieving or handling the information, or for processing the request for copies" (see `http://www.hipaadvisory.com/action/legalqa/law/Legal47.htm`).

credit card information. Lynn's automated trust negotiation strategy determines that a VISA and Mastercard credential will satisfy the malicious policy. Depending on Lynn's privacy preferences, these credentials may be considered sensitive. If not, they are disclosed to the fraudulent hospital website, resulting in the automated theft of Lynn's credit cards. Alternately, Lynn replies to the credit card-phishing policy with another policy protecting the credit card credentials. The phishing agent must be able to satisfy Lynn's policy in order to successfully steal the credit cards. Phishing attacks will be successful to the extent that the attacker can impersonate the real server and its credentials.

Because of automation and transparency, users like Lynn may unknowingly fall victim to malicious policies because of their own security software. If Lynn were able to detect that the malicious policy was requesting credit card information during an apparent medical records request, the attack might have been subverted. Li et al. [24] present a similar scenario in which a malicious server is able to detect that a user possesses arbitrarily specific sensitive credentials. However, the attacker cannot infer the exact content of those credentials without their disclosure.

Credential phishing is a potentially serious problem for automated trust negotiation, because policies are trusted and trust negotiation agents are designed to disclose whatever credentials are petitioned or the policies that guard their release. Current trust negotiation systems have no mechanism for detecting that a policy's required credentials are not relevant to the context of the original request.

Malicious policies are not limited to usurping sensitive information like credit cards. They may be used in a variety of ways to attack the very core of the trust negotiation protocol itself. For example, extremely large and complicated policies can be designed to cause a denial of service by requiring the compliance checker to spend extra computational cycles attempting to satisfy a policy. The final scenario presents another possible attack based on malicious policies.

1.2.3 Malicious strategy scenario

Recently, Ryutov et al. [30] mentioned many possible avenues for attacking trust negotiation. One such method is to cause a denial of service attack by forcing a negotiation to continue without progressing toward a successful conclusion. The final scenario presents a denial of service attack caused by a malicious negotiation strategy.

Borders bookstore has set up a trust negotiation-enabled web server that offers discounted books to students of accredited universities. Terry, a disgruntled ex-employee decides to take revenge on the company and launch a denial of service attack on the discount service. Terry's trust negotiation strategy is carefully altered so that it randomly generates policies that do not lead to a successful negotiation. To begin the attack, Terry generates a request for a student discount. When the server responds with a policy stating that the client must be a student, Terry's negotiation strategy ignores the policy and replies with a random policy. A random policy is returned each time the server responds with a policy. Terry then begins many simultaneous requests for the discount, continuing until Border's trust negotiation server is rendered useless because legitimate requests cannot get through.

The Borders trust negotiation server was vulnerable to this type of denial of service because Terry modified a trust negotiation strategy, and the server could not detect the irrelevant policies. Ryutov et al. also discuss an approach for responding to denial of service attacks by adapting policies according to a server suspicion level and adjusting various server time-outs. Alternatively, this thesis considers a context-based approach for detection of policy irrelevant information requests.

1.3 Solution Overview

These three scenarios depict a range of negative consequences provoked by policies and negotiation strategies that do not “play fair.” To mitigate the potential risk to users of trust negotiation and to make trust negotiation a robust authentication protocol for open systems, previous assumptions that policies and negotiation strategies will “play fair” must be challenged.

This thesis leverages a new source of context to fortify trust negotiation systems. As detailed in Chapter 2, context is essential when negotiating trust—however, current contexts are insufficient to provide the capacity to establish the relevancy of an unfamiliar policy. Other authentication systems [9, 40] similarly incorporate external contexts to buoy their information sources allowing a *smarter* authentication to occur. However, our solution is unique in that it is the first to use context to infer the relevance of access control policies themselves and to assert their relevance to the negotiation.

The context used to protect trust negotiation policies will be expressed using ontologies as outlined in Chapter 3. Ontologies possess many features that make them ideal for context sharing, reuse, and distribution to agents in an open system. Ontologies containing negotiation context are called *trust negotiation ontologies*. Chapter 4 introduces the credential relevancy framework; one solution that uses trust negotiation ontologies to thwart attacks by malicious policies and negotiation strategies. The credential relevancy framework is designed as an extension to current trust negotiation systems lacking sufficient context to protect themselves. Chapters 5 and 6 explore the framework’s two main components, the *ontology context channel* and *decision module*.

This thesis also contains a prototype implementation of the credential relevancy framework. The implementation is intended to validate the feasibility of the framework and to ensure that the overhead incurred by the framework is negligible (i.e., less than 10% of the total computational overhead). Chapter 7 presents the implementation of the credential relevancy framework, which is subsequently analyzed in Chapter 8. Finally, Chapter 9 presents the conclusions drawn from this work and discusses future work.

Chapter 2 — Relevancy Determination Obstacles

There are many obstacles that must be considered when designing a credential relevancy framework for trust negotiation. Four primary obstacles are: context, user discretion, context source, and universal agreement. Understanding these obstacles provides a foundation for designing a credential relevancy framework.

2.1 Context

As described in Chapter 1, policies are the sole context for deriving appropriate satisfying credentials considered by current trust negotiation systems. This section outlines other currently available contexts and identifies a new context that makes credential relevance detection possible.

2.1.1 Available contexts

Trust negotiation systems use many existing contexts as bases for determining relevancy. Beyond the contexts explicitly utilized in credential verification (issuer, subject, attributes, expiration, signature, and type[1]), other existing contexts are available but often overlooked. Four available contexts include: *time*, *frequency*, *location*, and *session-based information*.

Time is a beneficial context for geographically-based trust negotiations. For example, a trust negotiation server hosting national resources might use *time* to determine connection relevancy. Connections occurring during the middle of the night (relative to the client's time zone) might be ignored or flagged as suspicious. *Time* becomes less useful as geography becomes less relevant. Very localized systems, such as the *Aware Home*, have successfully integrated a useful time context [9].

[1]Generally, credential types are analogous to credential identifiers such as *credit card*, *driver license*, etc. IBM-based Trust Establishment (TE), reserves a specific OID *type* value within an X.509v3 credential allowing TE credentials to be mapped to profiles describing their attributes.

Frequency and *location* are two good contexts for determining suspicious behavior. *Frequency* (i.e., how often a credential is submitted or a negotiation occurs) and the physical location or IP address of a connection may indicate denial of service attacks on trust negotiation systems. Ryutov et al. [30] recently leveraged these contexts in their adaptive trust negotiation and access control (ATNAC) framework. Wullems et al. [40] grant or deny access to protected resources after validating a user's location context via GPS locators and iButtons.

Session-based context maintains the current state of a trust negotiation and also provides limited relevance determination. Prominent session-based context includes: negotiation history, policies and credentials received, trust requirements met, and round count. TrustBuilder, a trust negotiation prototype, uses session-based context to conclude that a negotiation has failed when no new policies or credentials are disclosed and all possible negotiation paths have been exhausted [42].

Other than the four previously discussed, useful contexts such as credential sensitivity, individual policy/credential frequency, credential submission count, credential and policy context groups (groups of related credentials), and credential attribute granularity, may contribute additional robustness and intelligence to future trust negotiation agents. The use of these contexts is not explored in this thesis.

2.1.2 Current contexts are too narrow

Unfortunately, none of the preceding contexts provides sufficient information to detect irrelevant credential solicitations or malformed policies because they do not express any relevant information about the nature of the negotiation. How can a trust negotiation agent deduce that a policy is irrelevant with respect to the *negotiation* when a *negotiation context* is unavailable? Such a negotiation context is required before a credential relevancy framework can be designed.

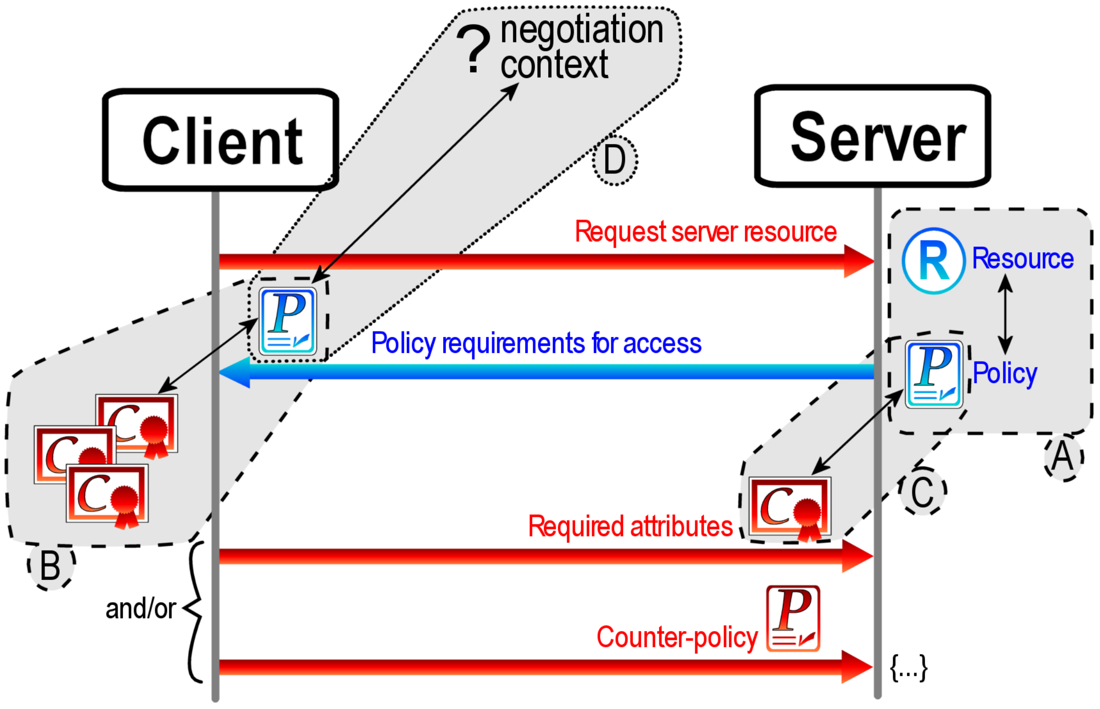


Figure 2.1: A sample trust negotiation annotated with three areas enclosing currently understood *relevant* contexts, labeled A, B, and C. An additional negotiation context is necessary to provide relevancy determination for policies (label D).

Trust agents depend on context information to make relevancy decisions critical to negotiation progress. Figure 2.1 graphically depicts three uses[2] of existing context that define how the term *relevant* is currently understood in trust negotiation literature [32, 39]. The dashed areas in the sample negotiation show the three instances of current relevancy determination during a trust negotiation. A dotted line surrounds the fourth area depicting a necessary, but missing, context for determining policy relevance.

The first area (Figure 2.1-A), encompasses all of the context that associates the requested resource R with its protecting policy P. This context is unique and

[2]Omitted from the figure is a fourth use of relevant context that relates to credential chain verification—the use of one credential's context to assert the relevancy of the next, and so on.

totally self-contained on the server—no external context is necessary to relate R to P because both are owned and maintained by the server. However, when P is released, all *implicit context*[3] associated with it is lost.

Area 2.1-B represents all the context used by the client to discover the credentials C necessary to satisfy P. Contexts from Section 2.1.1 can be applied in this area to improve client confidence or protect against denial of service, but these existing contexts fail to establish relevancy between P and the current negotiation context (area 2.1-D). Area 2.1-B is represented on the side of the client in the figure, but also applies to the server when a counter-policy is sent from the client.

Finally, area 2.1-C depicts the context that relate received credentials C to policy P. For the server, the policy is trusted but not the credentials, which is the inverse of area 2.1-B on the client. Contexts currently applied in area 2.1-C are necessary for trust negotiation because credentials are implicitly *irrelevant* until proven relevant[4]. Area 2.1-C occurs on both the client and server, depending on the current round of negotiation.

A new area of context is necessary to mitigate the problems of trusting potentially malicious policies received in area 2.1-B. Area 2.1-D must include context that relates a policy P or its required credentials C to a *negotiation class*[5] and is understood by each negotiating agent. Provided with such a context, policy P can be declared relevant, allowing credentials C to be selected with confidence.

Developing a context that associates negotiation classes with credential require-

[3] *Implicit context* refers to the server's intended usage, assumptions, and trust in the policy, while *explicit context* entails the predicates, rules, and trust requirements specified directly in the policy.

[4] Relevancy proof for credentials include: credential ownership verification, certificate integrity checking (signature checking), certificate revocation checking, valid credential proof chain—including a trusted root authority, attribute verification, and policy compliance.

[5] A *negotiation class* is a convenient shorthand for expressing the purpose, goal, or reason for a trust negotiation.

ments (policies) is a difficult task—primarily because negotiation types are linked to human perceptions, which are not easily expressible. Requiring users to specify policy relevance according to their perceived negotiation class may seem plausible, but leads to other obstacles as outlined in the next section. As will be shown, a more concrete context that formalizes negotiation classes is necessary.

2.2 User discretion

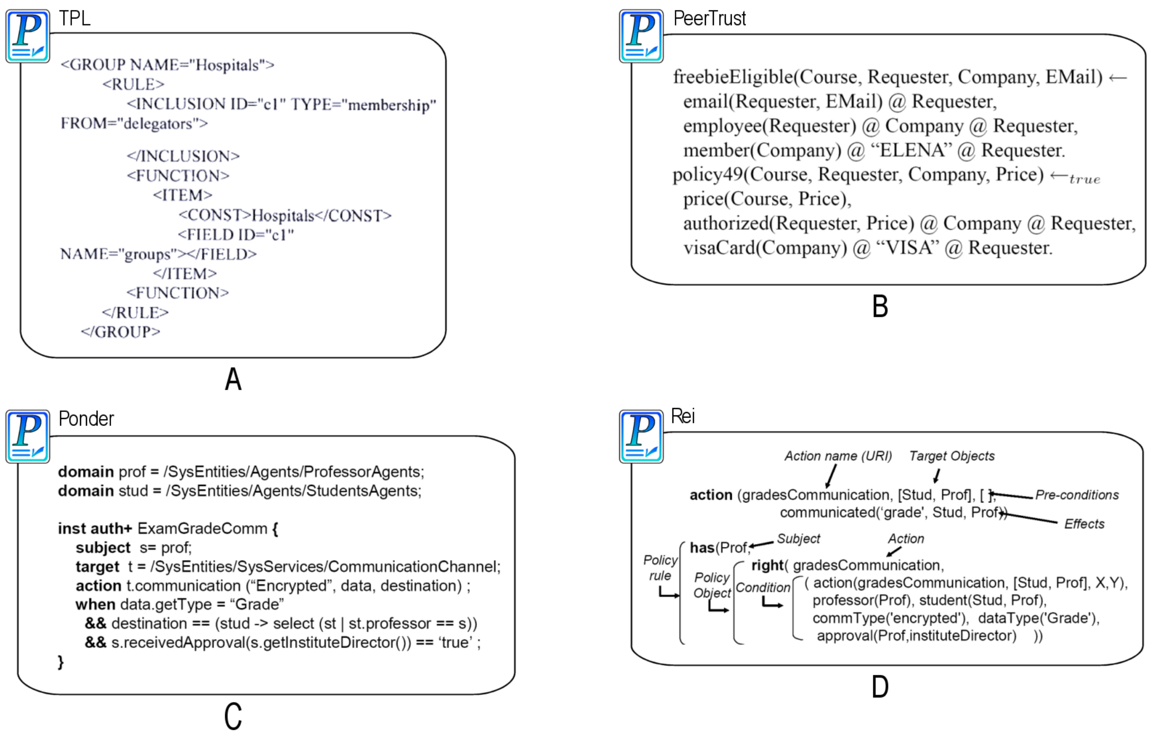
An easy solution that evades quantifying negotiation classes invokes user discretion concerning the relevancy of petitioned credentials. The user is interrupted and verifies that the credentials selected for release are relevant to the negotiation, as perceived by the user. While apparently ideal, this solution is impractical for four reasons: first, common users are poor judges of credential or policy relevance; second, user interruptions become prohibitive as trust negotiations get more complex; third, user interruptions violate the principals of automation and transparency; fourth, this solution is unilateral—it fails for servers.

2.2.1 Users uncertain of relevance

Requiring common users to make critical relevancy decisions regarding policy requirements or credential disclosures is impractical. Policy languages are designed for automated machine processing and are expressed in many differing syntaxes and semantics. Figures 2.2-A, 2.2-B, 2.2-C, and 2.2-D, show four current policy languages, TPL [14], PeerTrust [26], Ponder, and Rei [36], respectively. As yet, no standard trust negotiation policy language exists, and none of the current languages are easy for people to understand nor provide a translation to natural language.

2.2.2 User discretion prohibitive in complex negotiations

In simple negotiations (involving few policies and credentials), users are able to quickly verify the relevance of objects received or released by their trust agent.



```
<GROUP NAME="Hospitals">
        <RULE>
             <INCLUSION ID="c1" TYPE="membership"
FROM="delegators">

             </INCLUSION>
             <FUNCTION>
                  <ITEM>
                       <CONST>Hospitals</CONST>
                       <FIELD ID="c1"
NAME="groups"></FIELD>
                  </ITEM>
             <FUNCTION>
        </RULE>
    </GROUP>
```

```
freebieEligible(Course, Requester, Company, EMail) ←
  email(Requester, EMail) @ Requester,
  employee(Requester) @ Company @ Requester,
  member(Company) @ "ELENA" @ Requester.
policy49(Course, Requester, Company, Price) ←true
  price(Course, Price),
  authorized(Requester, Price) @ Company @ Requester,
  visaCard(Company) @ "VISA" @ Requester.
```

```
domain prof = /SysEntities/Agents/ProfessorAgents;
domain stud = /SysEntities/Agents/StudentsAgents;

inst auth+ ExamGradeComm {
   subject  s= prof;
   target  t = /SysEntities/SysServices/CommunicationChannel;
   action t.communication ("Encrypted", data, destination) ;
   when data.getType = "Grade"
     && destination == (stud -> select (st | st.professor == s))
     && s.receivedApproval(s.getInstituteDirector()) == 'true' ;
}
```

```
action (gradesCommunication, [Stud, Prof], [ ],
          communicated('grade', Stud, Prof))

has(Prof,
     right( gradesCommunication,
          ( action(gradesCommunication, [Stud, Prof], X,Y),
            professor(Prof), student(Stud, Prof),
            commType('encrypted'),  dataType('Grade'),
             approval(Prof,instituteDirector)    ))
```

Figure 2.2: Four examples of current trust negotiation policy languages.

However, actual trust negotiations may become very large, consisting of many negotiation rounds, policies, and credentials. For example, actual policies[6] are exponentially more complicated, involving thousands of predicates. Such policies necessitate hundreds of credential disclosures. Requiring user intervention to declare policy or credential relevance in a timely and efficient manner becomes prohibitive as policy complexity and credential numbers increase.

2.2.3 A violation of automation and transparency principles

Requiring user permission for credential release violates the principles of automation and transparency described in Section 1.1—trust negotiation should occur automatically and not be intrusive to the user. A relevancy detection framework that

[6]The Cassandra trust negotiation system [2] uses an Electronic Health Record policy, designed for the UK's National Health Service, for managing access to heath records (see `http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-628.pdf`).

relies completely on user intervention directly opposes these principles. However, a completely automated and transparent framework risks making improper decisions in sensitive, critical, or ambiguous situations. Prudence suggests interrupting the user in these situations, while otherwise respecting the principles of automation and transparency. This thesis presents a framework permitting user interruption only in cases of irrelevancy. Ongoing research efforts will determine the ideal balance between trust negotiation principles and user interruption.

2.2.4 A uniquely client-side approach

Finally, a relevancy detection framework driven by user intervention is a uniquely client-side approach—trust negotiation servers must adhere to the principles of automation and transparency more strictly because users cannot be present to oversee server negotiations.

2.3 Negotiation context location

Requiring user discretion to evade formally defining trust negotiation classes is impractical in all but the simplest cases, therefore a formal context containing negotiation classes must be created and accessible to both client and server. Figure 2.3 illustrates four possible configurations for the location of a negotiation context—on the client, on the server, on both the client and server, and on a third party. Table 2.1 describes the pros and cons of these four configurations.

Locating relevancy source information on either the client or server alone is an impractical solution primarily because one agent must rely wholly upon the other for negotiation context. Such a configuration leads to problems of trust in the received context—the same problems that already exist with policies. The most plausible solutions provide a source of negotiation context either on *both* the client and server or on a third party.

Pros	**Cons**
Negotiation context on the client (see Figure 2.3-A)	
Client stores only pertinent negotiation classes—no need to scale.	Requires additional user configuration.
Fully controlled and customizable to client preferences.	Information interoperability problems with servers that may request client relevancy information.
Client need not trust server context.	Client configuration and customized preferences may be error-prone.
	A server without relevancy context cannot protect itself from malicious clients.
Negotiation context on the server (see Figure 2.3-B)	
Client needs no extra configuration—all management is handled on the server.	Client cannot trust server negotiation context because both the policy and context belong to the untrusted server.
Server need not trust the client.	Server must determine one or more standard relevancy preferences for all clients.
	Client cannot leverage predisposed negotiation (see Section 4.2.2) or other trust negotiation acceleration algorithms based on negotiation context.
Negotiation context on both client and server (see Figure 2.3-C)	
Client and server can customize their preferences independently.	Possible exchange of negotiation class preferences between servers or clients is freeform—leading to parsing or translation failures.
Client and server are not co-dependent for sources of negotiation context.	Increase in percentage of relevancy determination false positives due to misconfigured server or client preferences.
Negotiation context need not be exchanged or standardized.	Proliferation of errors and inconsistencies between negotiation class preferences.
Negotiation context on a third party (see Figure 2.3-D)	
Third party certifies the correctness of negotiation context.	Client and server must trust the third party.
Third party mediates between client and server, ensuring consistency.	
Standardizes negotiation context syntax and semantics.	

Table 2.1: The pros and cons of four configurations for negotiation context source.

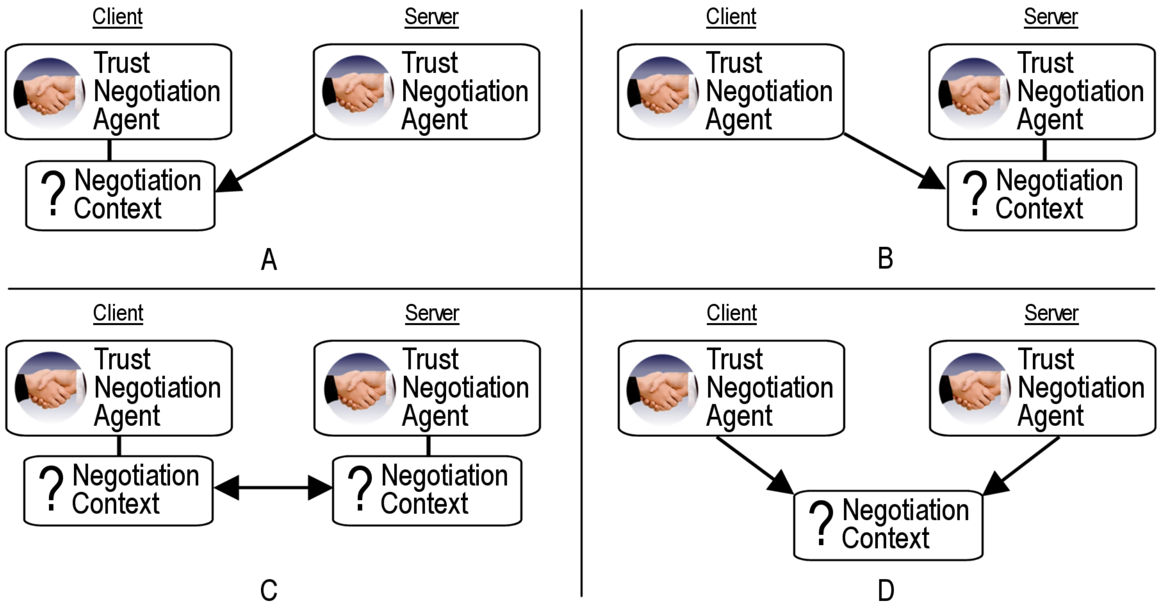


Figure 2.3: Four configurations illustrating the location of negotiation context relative to a client, server, and optional third party. Negotiation context may be only on the client (A), only on the server (B), on both client and server (C), or on a third party (D).

Ad-hoc trust exchange systems, known informally as a *web of trust* [20, 41], could provide an adequate solution to the problems of locating relevancy context on both clients and servers. Web of trust systems use majority vote to sway the consensus of what is considered trustworthy by exchanging ad-hoc *trust metrics*. These systems could easily be modified to exchange negotiation context in some form of *relevancy metric* associated with specific negotiation classes. Because web of trust systems leverage human perceptions of trust, they are well suited to convey negotiation class preferences. However, the quantization, summarization, and expression of relevancy preferences as relevancy metrics remains open to specification.

The most significant obstacle to hosting negotiation context on a third party is that of trusting the third party. One acceptable solution is for the third party to digitally sign the context. Thus, negotiation contexts become nearly analogous to digital credentials and principles of trust for digital credentials can naturally be extended to signed context.

This thesis explores the possibility of hosting negotiation context on a third party, rather than exchanging and retrieving context from a web of trust.

2.4 Universal agreement

Standardized negotiation classes, certified to be syntactically correct and digitally signed to be trustworthy, are possible when the context source is provided by third parties. In addition, these parties are responsible for defining the semantic meaning of negotiation classes. Unfortunately, the inevitable diversity of available third parities will result in heterogeneous semantic meaning. While a universal agreement of semantic meaning is desirable, it is also impractical.

Hypothetically, universal agreement involves one or more unified third parties that provide *the only source* for semantic meaning, thus defining all possible negotiation classes. One authoritative source is impractical considering the diversity of conceivable trust negotiations. Furthermore, one source for relevancy is also a single point of failure and attack.

Realistically, many diverse third parties will offer potentially competing negotiation contexts. Third parties may specialize within a certain domain of negotiation classes, providing specific solutions to specific users. From diverse third parties, clients and servers may select those that best suit their needs.

Chapter 3 — Introduction to Ontologies

This chapter introduces ontologies and describes the features that make them ideal for structuring negotiation contexts.

3.1 Understanding ontologies

The term ontology has roots in the discipline of philosophy and means *the study of being or existence*[1]. In computer science, the word also describes the terms and relationships that exist within a given knowledge domain. Ontologies are conceptually a formal structure for specifying the vocabulary of possible language, grammar rules, and relationships within a particular domain of interest.

Ontologies are composed of relationships between objects, often in a hierarchal structure. Figure 3.1 illustrates two different valid ontology examples for a book, using both tree and graphical notation[2]. In Figure 3.1-A, the objects represent different *types* of books, as shown by the *is a* relationships between them, while Figure 3.1-B categorizes a book by its different attributes using the *has a* relationship. In general, an ontology is not limited to one type of relationship—the book ontology of Figure 3.1 might also include other relationships such as *cardinality*$\geq$*2*, *disjointWith*, *sameAuthorAs*, etc. The figure illustrates only two of an infinite variety of possible ontology designs; differing designs or classifications are known as *dimensions.* In principle, the number of distinct classifications for an object are unlimited because the number of possible dimensions along which to categorize it cannot be exhaustively specified [7].

[1] A quick overview of ontologies may be found in the article "Ontology (computer science)" available on the Wikipedia (see `http://en.wikipedia.org/wiki/Ontology_(computer_science)`).

[2] The graphical notation for representing ontologies is borrowed from Horridge [16], and is similar to Venn diagrams.

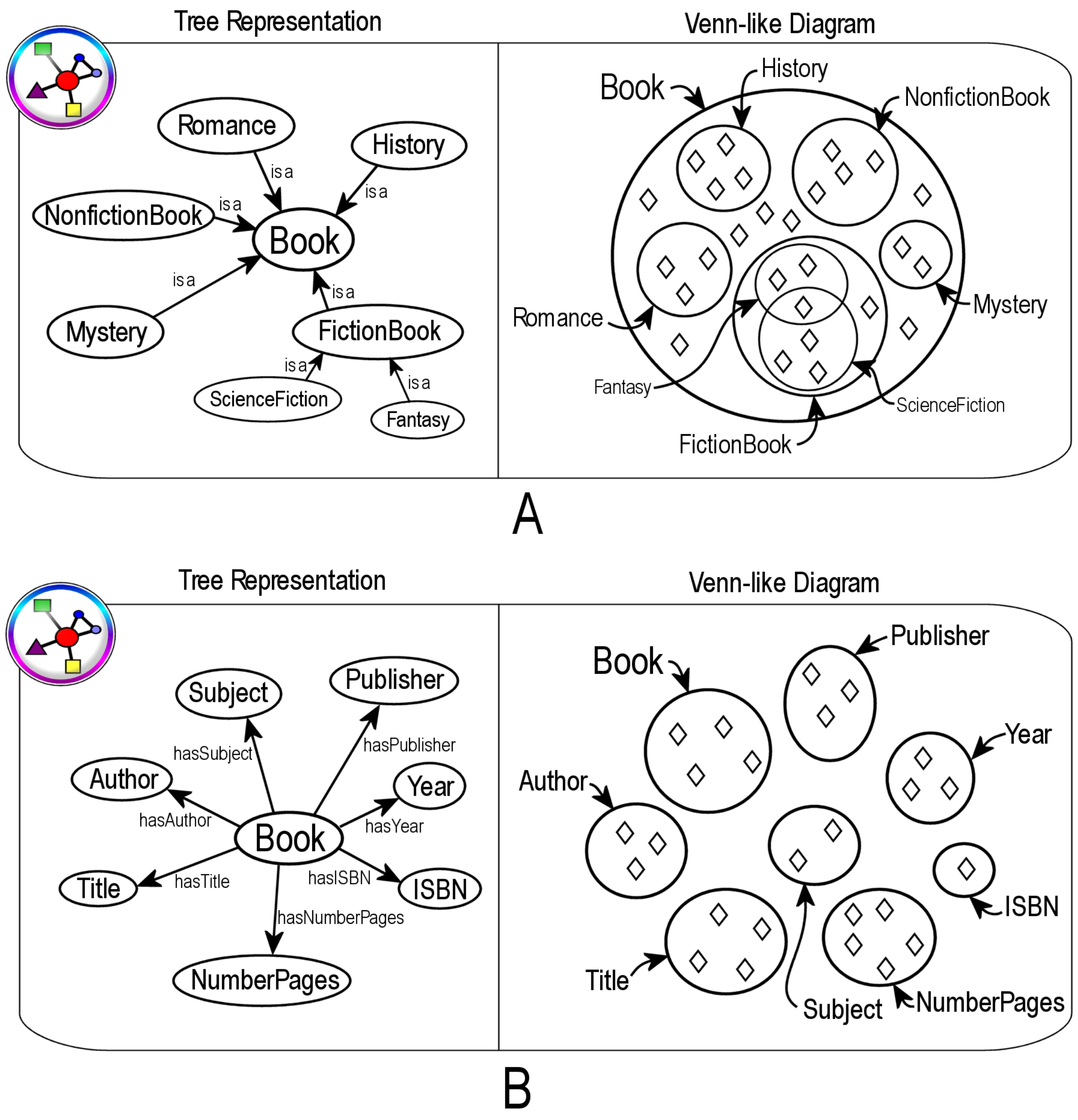


Figure 3.1: Two alternative ontology *dimensions* for describing a book. Each ontology is represented in both a tree and Venn-like diagram (see Horridge et al. [16]). Circles represent classes, arcs represent properties, and diamonds represent arbitrary instances. (A) describes a book using *is a* properties and (B) classifies a book by its component parts using the *has a* property.

Objects used in ontologies are formally called *classes*. A class is a set or grouping of similar items that share class relationships, attributes and restrictions. In Figure 3.1, classes are represented by the circles labelled *Book*, *FictionBook*, *Fantasy*, etc. The classes described in an ontology define the extent of the vocabulary used in a specific knowledge domain [37]. While no fixed rules exist for ontology design, classes are usually organized into *is a* relationships that relate specific classes to more general classes as the class hierarchy is traversed. Once the fundamental *is a* relationships are defined, more complex relations may be established.

The items contained in classes are known as *instances*. Instances are often omitted from the actual ontology specification and are included instead in databases. Instances are classified based on the properties they contain. For example, the instance *Star Wars*, would be included in the class *Book* because it *is a* book, *FictionBook* because it is fictional, and the *ScienceFiction* and *Fantasy* subclasses because it could belong to both.

Relationships between classes, formally known as *properties* or *slots*, act as rules for describing complex and dynamic associations between classes. Besides basic *is a* properties, two additional common properties include *restrictions* and *cardinality constraints*. Restriction properties limit the instances included in a class to the properties possessed by the instance, and cardinality constraints specify the number of instances that can be included. For example, the class *MultipleAuthors* described in a book ontology might include a restriction of $\exists$ *hasAuthor MultipleAuthors*[3] and a cardinality constraint of ≥ 2 that together limit the valid set of instances in that class to those that have an author property with two or more authors listed.

[3]This restriction is read: "at least one *Author* that is an instance from the class *MultipleAuthors*," meaning that if an instance has at least one *author* attribute whose value is an instance belonging to the class *MultipleAuthors*, then that instance belongs to the restriction.

Ontology research is divided into two veins, one theoretical and the other practical. Theoretical ontology research is concerned with the logical and systematic organization of general concepts in the world. Ontologies of this nature, designed for maximum reuse and ubiquity, are known as *foundation* or *upper* ontologies. Several examples of current foundation ontologies include CYC[4], GUM[5], WordNet[6], SUMO[7], and J. F. Sowa's *Knowledge Representation*[8]. Upper ontology research continues but is unlikely to yield a practical solution because the level of detail necessary for real world tasks are outside its scope [7].

The key to practical ontology research are ontologies tailored to a specific problem domain, rather than attempts to describe all possible domains. *Domain ontologies* refer to ontologies localized to a certain problem domain. Domain ontologies prohibit large scale information sharing and reuse, but make ontology design practical for applications. Ontology design tools like Protégé [12, 16, 28] facilitate rapid, accurate domain ontology creation and management.

The proliferation of domain ontologies necessitated the creation and standard-

[4]Derived from the word en*cyc*lopedia, CYC is a comprehensive ontology and database for enabling AI applications to perform human-like reasoning. Began in 1984 by Doug Lenat, CYC is used in many applications and has spawned an open-source derivative *OpenCYC* (see `http://www.cyc.com`).

[5]The *generalized upper model* is an ontology targeted at expressing information in natural language (see `http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/gum/index.htm`).

[6]WordNet, developed at the cognitive science laboratory at Princeton, is an upper ontology representing natural language (see `http://wordnet.princeton.edu`).

[7]The *suggested upper merged ontology* is part of the IEEE Standard Upper Ontology Working Group and targets meta, generic, abstract, or philosophical concepts [27] (see `http://ontology.teknowledge.com`).

[8]An ontology based on the book *Knowledge Representation* by J. F. Sowa with influence from philosophers Charles Sanders Peirce and Alfred North Whitehead, pioneers in symbolic logic (see `http://www.jfsowa.com/ontology/`).

ization of ontology languages such as DAML+OIL[9] and more recently OWL[10] [37] that formally describe the syntax used to specify ontologies. Formal specification allows ontologies to be machine understandable and thus processed by logic reasoners like *Racer*[11], when the expressiveness[12] of the ontology language supports it. Logic reasoners understand ontology rules (specified by properties), can infer new classes not explicitly specified, and verify ontology consistency. Formal specification and logic reasoners make ontologies very useful for automated systems and ideal for use on the *semantic web* [1, 25].

Additionally, ontologies facilitate knowledge sharing. Through formal specification, the structure and rules for processing internal information is available for automated systems to share. For instance, a business can simplify and automate its sharing of public database information by sharing the ontology that allows other businesses to automatically interpret and extract their information.

The features of ontologies previously discussed are summarized in Table 3.1. These features make ontologies ideal for use in trust negotiation as outlined in the next section.

3.2 Using ontologies in trust negotiation

The synthesis of ontology research and trust negotiation is a new research area. Resultant research found that ontologies allow convenient expression of policies as

[9]The *DARPA Agent Markup Language (DAML)*, initiated by US government research groups, was developed for realizing the vision of the semantic web. Later, the independently developed *Ontology Inference Layer (OIL)* specification by European researchers was combined into one standard language.

[10]*Web Ontology Language (OWL)*, the W3C's evolved version of DAML+OIL, contains additional language features that were not available in the old specification. OWL became an official W3C recommendation in 2004.

[11]The *Racer* project (see `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`).

[12]OWL-DL and OWL-Lite are two sublanguages of OWL that support automated reasoning because their expressiveness is limited to a decidable first order logic (algorithms that will terminate in *finite* time). OWL-Full provides the most expressive power at the expense of decidability [37].

Ontology Features
Natural hierarchal structure
Unique class definitions
Definition of arbitrary properties
Currently evolving research area
Fit for knowledge sharing and reuse
Standardized formal language
Property classification inference
Consistency verification

Table 3.1: Summary of ontology features.

demonstrated by the KaOS trust management system [36]. Policies expressed using ontologies enable constraint representation across diverse enforcement mechanisms [37]. These ontologies uniquely define the terms necessary for policy specification (in classes), the rules for policy satisfaction (properties), and are described in an application-neutral format. Policies described using ontologies also benefit from built-in consistency through the use of a logic reasoner that verifies an individual policy or the knowledge represented by a collection of policies as a whole.

Leithead et al. [23] show how policies can be used in conjunction with ontologies to provide additional ease of management, structure, and privacy. They describe ontologies containing hierarchies of digital credential attributes using *is a* properties. Policies protect privacy during early stages of trust negotiation by requesting only the most general attribute from an ontology class[13]. After more trust is established, policy requests for more specific credential attributes are obtained by locating a more specific ontology class.

Leithead et al. [23] laid the groundwork for this thesis by introducing *negotiation classes* (as referenced in Chapter 2) in domain-specific ontologies. These ontologies

[13]Leithead et al. are ambiguous when describing ontology structure. The representation of digital credential attributes as ontology *classes* or *instances* is a trivial implementation issue.

contain classes that provide hierarchical negotiation context and the properties that link them to digital credential attributes. In this thesis, ontologies containing these negotiation classes are referred to as *trust negotiation ontologies*. Chapter 5 describes the structure and use of trust negotiation ontologies.

Authorized third party providers that describe *typical* trust negotiation ontologies are realistic and feasible. Businesses today leverage third party guidelines and certifications to bolster trust by demonstrating that they adhere to general or typical operational procedures. For example, banks follow guidelines set forth by the FDIC[14], employment agencies claim *equal opportunity employer*[15], web sites follow privacy and security guidelines prescribed by *TRUSTe*[16], etc. Even big businesses strive to certify to minimum quality standards such as iso9001[17] and other certifications. Such agencies exist to augment user trust in the business' credibility. These or similar agencies will provide and maintain the trust negotiation ontologies that strengthen user trust in online negotiations by protecting them from irrelevant or malicious policies.

The use of ontologies in trust negotiation is an exciting new research area. Few trust negotiation systems currently utilize the benefits of ontologies. Future trust negotiation agents can harness the benefits of ontologies for a higher degree of automation and negotiation-specific knowledge acquisition (e.g., understanding diverse policy and credential languages in an open system).

[14]The FDIC regulates and examines banks, providing required examinations, laws, reports, training, etc (see `http://www.fdic.gov/regulations/index.html`).

[15]The US equal employment opportunity commission regulates cases of discriminations and provides various laws and guidelines for employers (see `http://www.eeoc.gov`).

[16]TRUSTe provides security and privacy guidelines for online business (see `http://www.truste.org`).

[17]The organizational structure, processes, and procedures necessary to ensure the quality of the overall intentions and direction of an organization [paraphrased] (see `http://www.iso9001qualityassurance.com`).

Chapter 4 — The Credential Relevancy Framework

This chapter presents the *credential relevancy framework*, an extension to trust negotiation. The credential relevancy framework utilizes the negotiation context necessary to detect irrelevant or malicious policies as described in Chapter 1. This chapter presents an overview of the framework, its applications and benefits to trust negotiation. Details of the framework's two main components—the ontology context channel and decision module—comprise Chapters 5 and 6.

4.1 Overview

The credential relevancy framework is an extension to current trust negotiation systems that lack sufficient context to determine the relevancy of credential requests. The two main components of the credential relevancy framework are the *ontology context channel* and the *decision module*, as illustrated in Figure 4.1. The ontology context channel queries and maintains trust negotiation ontologies describing relevant negotiation classes. The decision module produces a relevant subset of credentials based on input from the ontology context channel and local credential repository.

4.2 Credential relevancy in trust negotiation

As an extension to current trust negotiation systems, the primary application of the credential relevancy framework augments privacy and user protection from irrelevant or malicious credential requests. Two additional applications for the framework are *predisposed negotiation* and *relevant credentials first*. The following sections describe these three applications of the credential relevancy framework.

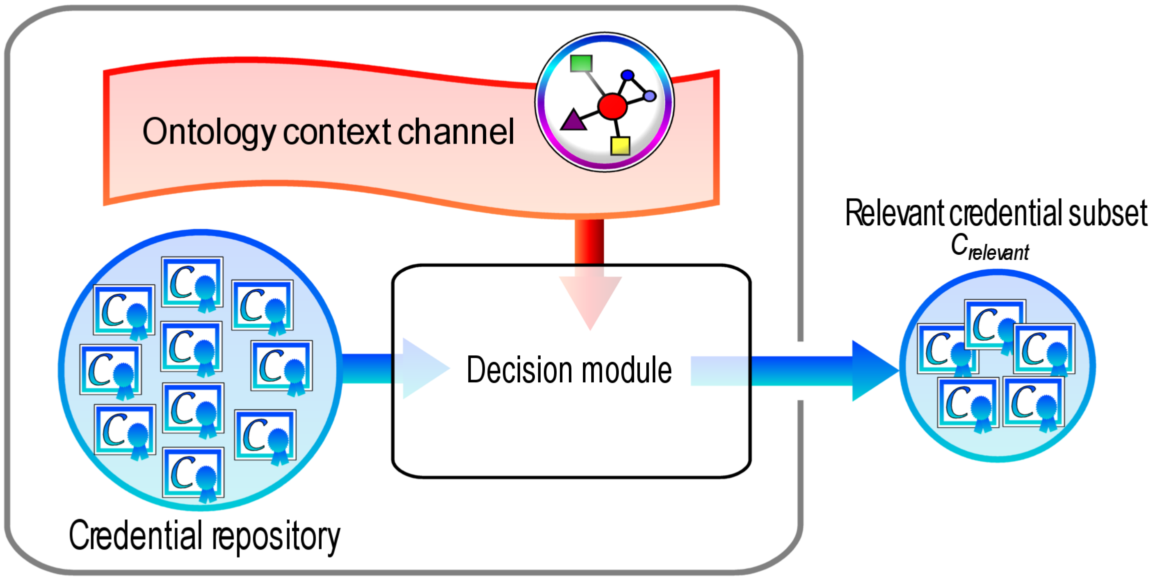


Figure 4.1: An overview of the credential relevancy framework. The *decision module* receives a user's credential repository and the *ontology context channel*, and produces a relevant credential subset as output.

4.2.1 Irrelevant or malicious policy detection

Primary application of the credential relevancy framework provides necessary protection from irrelevant or malicious policies. Figure 4.2 illustrates how the framework integrates with an existing trust negotiation system.

Recalling the scenario from Section 1.2.1, Pat makes a request to renew a driver license. This request is answered by a policy from the DMV server. Pat's trust negotiation agent uses that policy as context for deriving appropriate satisfying credentials $C_{derived}$. $C_{derived}$ includes Pat's private disabled and concealed weapon credentials. These credentials fulfil the policy's proof of identity requirements, but reveal information that is private and irrelevant to the negotiation.

Using the credential relevancy framework, the release of Pat's private and irrelevant credentials will be detected and prevented. Before $C_{derived}$ is disclosed, the ontology context channel consults a trust negotiation ontology specifying the rele-

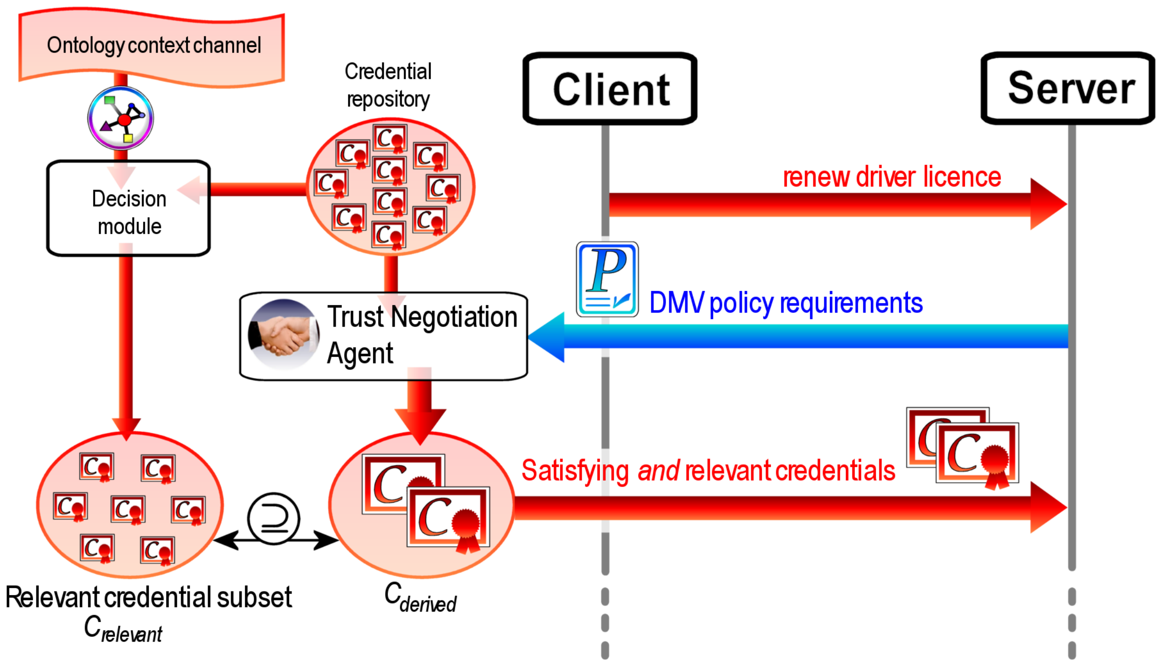


Figure 4.2: The primary application of the credential relevancy framework—detection and prevention of irrelevant and malicious policies. The client's credential repository is used both by the trust agent to evaluate the DMV policy and by the decision module to calculate $C_{derived}$ and $C_{relevant}$, respectively. $C_{derived}$ is relevant to the current negotiation if it is a subset of $C_{relevant}$.

vant credentials typically used in a driver license renewal negotiation. The decision module receives this input, together with the credential repository, and produces a subset of Pat's credentials $C_{relevant}$. If $C_{relevant}$ includes $C_{derived}$, then the set $C_{derived}$ are assumed relevant and released. In this scenario, the disabled and concealed weapon credentials are not found in $C_{relevant}$. Pat's trust agent may reevaluate the policy, trying different combinations of $C_{derived}$ and comparing it to $C_{relevant}$ until relevant credentials are released. Alternately, $C_{relevant}$ could replace the credential repository used by the trust agent to evaluate the policy, producing a subset $C_{derived}$ guaranteed to contain relevant credentials.

By using the credential relevancy framework, Pat's trust agent selects only the relevant credentials that satisfy the server's policy. The primary application of the framework is only invoked as needed (i.e., only on occasions when Pat receives a policy). Figures 4.2, 4.3, and 4.4 show use of the framework exclusively by one negotiating party (rather than both client and server) to simplify the illustrations.

4.2.2 Predisposed negotiation

An alternate application of the credential relevancy framework potentially expedites trust negotiation by pre-selecting and sending credentials that are relevant to the server in advance. This technique, termed *predisposed negotiation*, was first introduced by Leithead et al. [23] and is based on an optimized *eager* strategy [31]. Figure 4.3 illustrates predisposed negotiation using the credential relevancy framework.

The credential phishing scenario from Section 1.2.2 demonstrates the application of predisposed negotiation. Lynn's agent begins by collecting relevant context from a trust negotiation ontology that pertains to HIPAA medial record requests. Using this context and the current credential repository, the framework produces $C_{relevant}$.

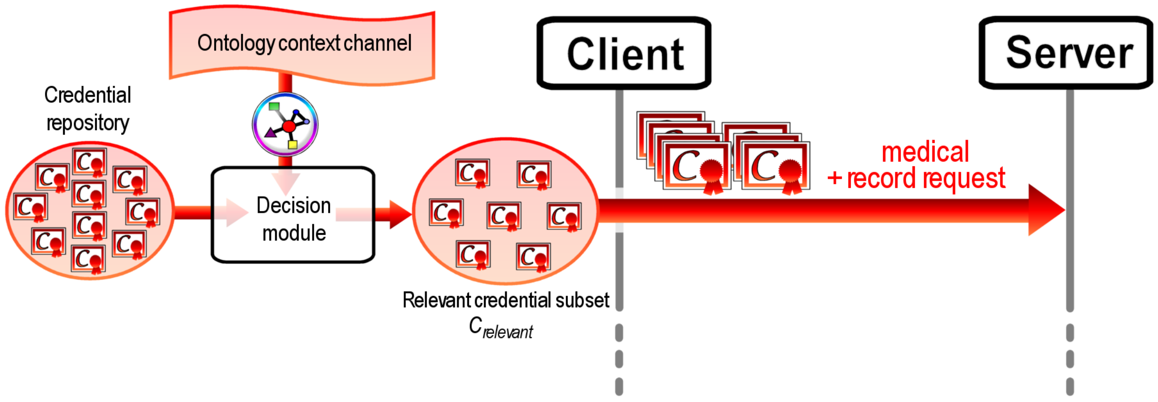


Figure 4.3: Predisposed negotiation produces a subset of relevant credentials to include with the request to access medical records.

$C_{relevant}$ is then sent to the server with the initial request for Lynn's medical records. Assuming that the hospital's server is prepared to receive predisposed negotiations, Lynn's request could be granted without further negotiation rounds, if $C_{relevant}$ satisfies the server's policy.

Through predisposed negotiation, Lynn gains access to the requested medical records in one round of negotiation. Predisposed negotiation yields a significant optimization when complex negotiations involve many rounds[1].

4.2.3 Relevant credentials first

A third application of the credential relevancy framework in trust negotiation involves creating a set of relevant credentials $C_{relevant}$ for faster searching. Clients and servers are likely to have many credentials stored in their repositories. Trust negotiation internal processes search credential repositories to select credential sets (e.g., in order to calculate $C_{derived}$), find all possible satisfying sets[2], attempt decryption,

[1]Predisposed negotiations should not release sensitive credentials, therefore not all negotiations will reduce to a single round.

[2]Smith et al. [35] note that satisfying set generation using a type 2 compliance checker has a complexity of $O(2^{|U|})$, where U is the union of satisfying sets selected from the local credential repository. As U becomes large (>15), the algorithm becomes impractical due to its exponential

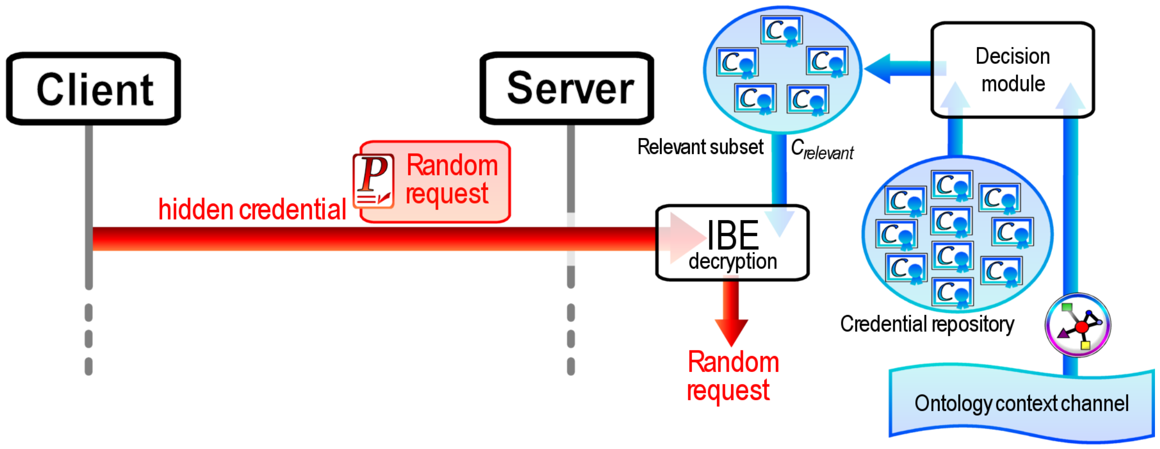


Figure 4.4: Another application of the credential relevancy framework is to calculate $C_{relevant}$ and use that subset instead of the credential repository when decrypting hidden credentials.

etc. For example, decryption of *hidden credentials*[3] [6, 8, 15] is computationally expensive and increases in complexity with the number of credentials in a repository. Figure 4.4 illustrates how the credential relevancy framework optimizes the decryption of hidden credentials by trying the most relevant credentials first.

Drawing from the scenario of Section 1.2.3, Terry stages a denial of service attack on a bookstore server by sending a random request encrypted in a hidden credential. The bookstore trust negotiation agent is unable to service the request before decrypting the hidden credential to verify the specific service requested. To protect itself from irrelevant requests and minimize the computational resources spent on this connection, the server derives $C_{relevant}$ based on its default student discount negotiation context and uses $C_{relevant}$ first to attempt decryption. If none of the credential subset from $C_{relevant}$ successfully decrypts the hidden credential, then the server rejects the request. Thus, Terry's attempt to exhaust the server's compu-

property.

[3]Hidden credentials are an encrypted package containing a resource encrypted using *Identity-Based Encryption (IBE)* in such a way that a recipient can decrypt it only if the correct credentials are used. For more information see Bradshaw et al. [6, 15].

tational resources by sending irrelevant requests encrypted with hidden credentials will prove ineffective. The server benefits by using $C_{relevant}$ first when decrypting hidden credentials.

Using the credential relevancy framework to form a subset of relevant credentials for early consideration is very useful. However, when sufficient negotiation context is not available, the credential relevancy framework fails to produce a beneficial $C_{relevant}$ subset. This might occur when hidden credentials (or policies) are received from an unknown source. Negotiations should rarely occur without any context—in such situations, the user should be wary of malicious behavior.

Chapter 5 — The Ontology Context Channel

The ontology context channel connects the credential relevancy framework with trust negotiation ontologies and provides the knowledge base for relevant negotiation classes. This chapter describes the structure of a trust negotiation ontology, an analysis of channel trust and availability, and the methods by which negotiation contexts are aligned with users' perceptions of those contexts. The chapter concludes with a summarizing usage example of the ontology context channel.

5.1 Trust negotiation ontology structure

Trust negotiation ontologies, introduced in Section 3.2, contain taxonomies of negotiation classes with linking properties to pertinent credential attributes. To illustrate, suppose the US government provides a DMV third party trust negotiation ontology. While state DMV policies vary in their specific requirements, the ontology shown in Figure 5.1 describes general (typical) negotiation classes for DMV transactions online. These might include transactions such as *LicensingNegotiation*, *RegistrationNegotiation*, or *CarHistoryNegotiation*. These negotiation classes are organized into a taxonomy using *is a* properties that relate more specific to more general negotiation classes. Thus, according to Figure 5.1, a *DriverLicenseRenewal-Negotiation* is more specific than a *LicensingNegotiation*, which is more specific than the most general *DMVNegotiation*.

Trust negotiation ontologies rely on existing credential attribute definitions. As introduced in Chapter 1, attributes contained in digital credentials describe properties or characteristics of an entity as asserted by the credential provider. Trust agents that exchange and process digital credentials must have a method for understanding and deriving meaning from unfamiliar credential attribute names and

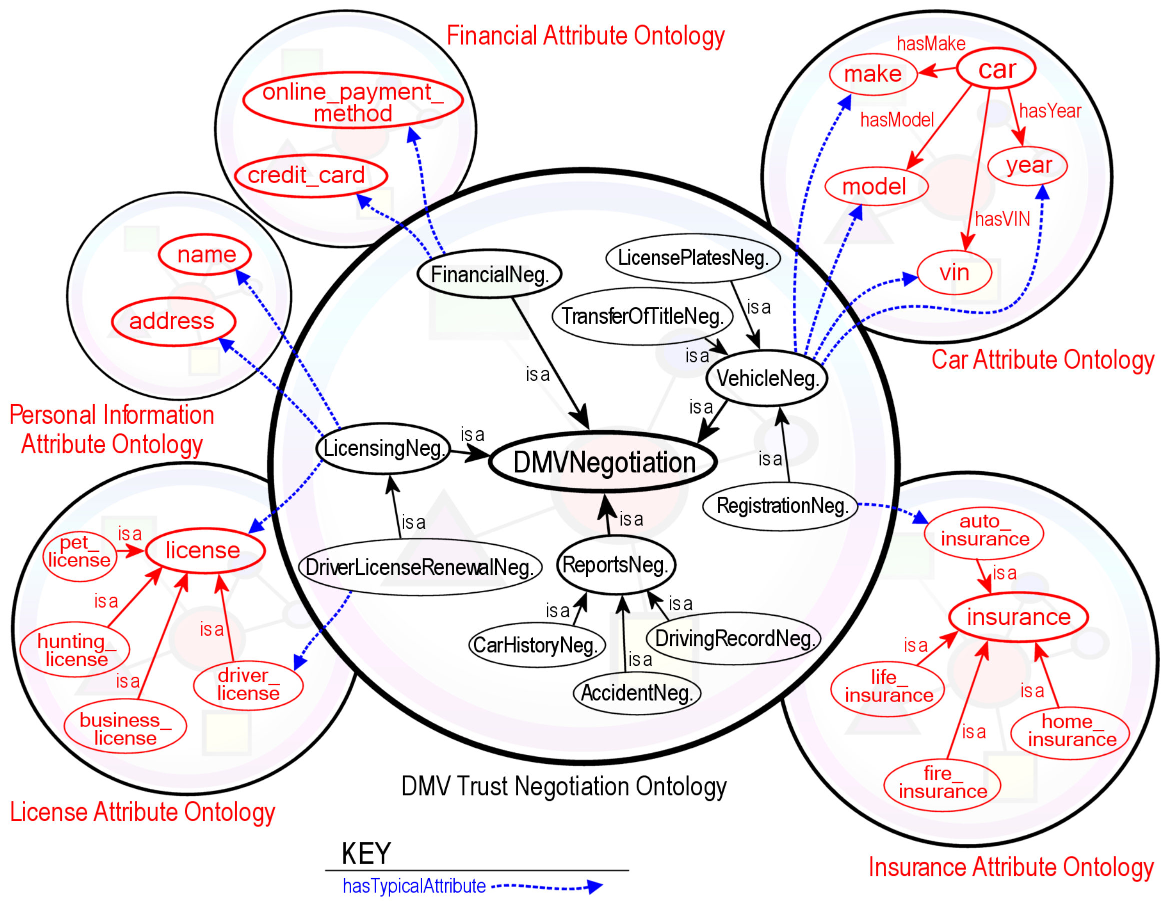


Figure 5.1: An example DMV trust negotiation ontology, illustrating the *hasTypicalAttribute* property that links negotiation classes to credential attributes within attribute ontologies.

values. Ontologies conveniently allow the expression, derivation of meaning, and organization of credential attribute names and legal values. *Attribute* ontologies[1], postulated by Leithead et al. [23], provide a diverse corpus of knowledge referenced or included in trust negotiation ontologies.

Properties in trust negotiation ontologies that specify typical credential attributes bridge negotiation classes with attribute ontologies. The binary property *hasTypicalAttribute* relates the domain of negotiation classes to the range of credential attributes. Figure 5.1 illustrates four *hasTypicalAttribute* properties linking the class *VehicleNegotiation* to typical attributes: *make*, *model*, *year*, and *vin*.

5.1.1 Inheritance and Subsumption

Trust negotiation ontology classes linked by the *is a* property inherit all the properties of their parents in the hierarchy. For example, a *RegistrationNegotiation* class with a *hasTypicalAttribute* property specifies that an *auto_insurance* credential attribute is typical of such negotiations. Additionally, each credential attribute of its parent negotiation class (*make*, *model*, *year*, *vin* of the class *VehicleNegotiation*) is inherited and thus also typical of a vehicle *RegistrationNegotiation*.

Inherited credential attributes are subsumed by a negotiation subclass when the *hasTypicalAttribute* property on that subclass refers to a more specific attribute. Subsumption only occurs for credential attributes that share an *is a* property within an attribute ontology. For example, Figure 5.1 shows the *license* credential attribute, inherited from the *LicensingNegotiation* class, being subsumed by the *driver_license* attribute for a *DriverLicenseRenewalNegotiation* class. Subsumption occurs because the *DriverLicenseRenewalNegotiation* is a subclass of *LicensingNegotiation* and the inherited *license* attribute is overridden by *driver_license*.

[1]Other equivalent structures that organize and provide meaning for credential attributes may be substituted.

Generally, when traversing a trust negotiation ontology from the most general to the most specific class, the number and specificity of credential attributes increases using inheritance and subsumption. Therefore, to identify the credential attributes most typical of a current transaction, the most specific negotiation class that best describes the current transaction should be selected.

However, before negotiation classes can be selected for use, the trust negotiation ontology itself must be available to the context channel. The next section describes the accessability of the ontology context channel, including the trust, location and caching of trust negotiation ontologies.

5.2 Channel availability

The ontology context channel provides trusted access and availability of trust negotiation ontologies to the credential relevancy framework despite possible limited connectivity of third party providers.

As suggested in Section 2.3, third parties provide negotiation context through trust negotiation ontologies. Therefore, the significant problem of trust in ontology integrity and correctness must be addressed. Digital signatures provide integrity and non-repudiation, but do not verify the ontology's correctness. In practice, the reputation of the third party ontology provider will dictate the general correctness of the ontology, as the reputation of digital credential providers like VeriSign[2] is trusted today. Web-of-trust systems, used together with digital signatures, furnish additional trust in ontology correctness by ensuring a general consensus of trust.

Digitally signed ontologies only convey trust in a single static instance of the ontology at a specific time; however, trust negotiation ontologies will change as general notions of relevancy shift, trust-agent feedback occurs, and new irrelevant

[2]A presumably trustworthy digital credential provider (see `http://www.verisign.com`).

or malicious policies appear. Signed trust negotiation ontologies will consequently need to be updated and re-issued. Expiration dates used in conjunction with digital signatures, dictate the frequency that the ontology context channel will re-acquire a trust negotiation ontology.

Once authenticity and integrity are verified, trust negotiation ontologies are cached because connections to third party providers are not always available. The ontology context channel uses a list of trusted ontology providers[3] to acquire and cache ontologies from multiple third party providers. Before a trust negotiation ontology expires, the context channel downloads an updated version from the corresponding provider. Periodic checking, acquisition, and caching of new ontologies occurs only for trusted providers on the list.

Cached trust negotiation ontologies are stored independently or merged into a local ontology. The local corpus additionally supports the saving of user preferences by replacing default typical attributes with those authorized by user override. Futhermore, the single local ontology allows easy exchange of user preferences with other agents—a valuable extension for future support of web of trust systems.

After downloading, verifying, and caching trust negotiation ontologies, the ontology context channel is invoked when a negotiation begins. A specific negotiation class must then be selected to provide the most appropriate context for that session.

5.3 Context alignment

Context alignment is the process of matching user perceptions of the current transaction to the most appropriate negotiation class within the ontology context channel. This process is different for clients and servers because clients *infer* negotiation contexts while servers have *implied* context.

[3]Similar to browsers that have a pre-selected list of trusted root authorities.

For servers, the ontology context channel is pre-configured to use one or more relevant negotiation classes that best describe the server's services. Context alignment occurs manually by the system administrator at the time the server is brought online or scheduled when ontology updates are retrieved.

For clients, context alignment involves inferring a negotiation class based on the user's perception of the trust negotiation context. Also, automation and transparency principals dictate that users should not be interrupted; therefore, the ontology context channel supports several methods for identifying user perceptions with varying degrees of automation. Three methods for context alignment on the client are identified below in order of increasing automation.

5.3.1 User interaction

For trust negotiations initiated with an unfamiliar server, the user interaction context alignment method prompts the user to select the most appropriate negotiation class from the ontology context channel[4]. The selected negotiation class is then queried and saved in the local ontology with server agent information (domain, identifiers, requested resource) to prevent subsequent user interruption.

5.3.2 Content extraction

A more automated alignment method extracts text or other available context from the protocol used to invoke trust negotiation, analyzes it[5], and identifies the closest matching negotiation class. For example, an HTML page invokes trust negotiation over HTTP presenting descriptions and other text that establish the context

[4]The range of available negotiation classes is determined by the root classes of each trust negotiation ontology in the channel's list of trusted ontology providers. While the exact selection method is implementation-specific, a navigation and selection widget similar to file system tree navigation would be appropriate.

[5]The exact method for analysis is left as an implementation detail. Possible methods include statistical probability, bayesian methods, or machine learning techniques. The ontology context channel may filter messages received from the server like a proxy, progressively determining negotiation context while the user navigates.

basis for a user's perception of the desired resource or purpose of the negotiation. HTML text is extracted and analyzed to find the closest matching negotiation class[6]. The user interaction context alignment resolves insufficient or ambiguous protocol context.

5.3.3 Semantic web

A machine understandable web, made possible through continuing development of the semantic web [1, 25], may be incorporated with the credential relevancy framework to indicate appropriate negotiation classes based on explicit meta data rather than derived content as previously discussed. Formalized algorithms for providing trust on the semantic web are yet in infancy. Currently, semantic web meta data is trusted if the indicated negotiation class is found within the list of trusted ontologies maintained by the context channel. The ontology context channel may support additional automated context alignment methods through the semantic web as further research yields new results.

5.4 Example summary

The following example summarizes the details of the ontology context channel presented in this chapter. Pat begins a trust negotiation to renew an expired driver's license with an online DMV. The list of trusted ontology providers in the context channel includes one that distributes a DMV ontology. The DMV ontology was previously updated and deemed trustworthy (after verifying its signature, etc.). Because the channel is configured for user interaction context alignment and the server is unknown, Pat is prompted to select the most appropriate negotiation class. According to the DMV trust negotiation ontology from Figure 5.1, the *DriverLicenseRenewalNegotiation* class is the most specific class that best describes

[6]A related but inverse method that uses ontologies to extract information from HTML is described by Embly [10].

the current negotiation. The ontology context channel queries the *hasTypicalAttribute* properties for this negotiation class and returns the *driver_license* credential attribute. Additionally, the *name* and *address* attributes inherited from *LicensingNegotiation* are returned. Credential attributes are provided to the decision module (discussed in the next chapter), which creates a subset of relevant credentials $C_{relevant}$.

Chapter 6 — The Decision Module

This chapter examines the decision module, the engine that uses the credential attributes provided by the ontology context channel together with the credential repository and produces a relevant credential subset. The following section outlines the process by which typical credential attributes are mapped to credentials in the repository. Section 6.2 considers the implications of handling irrelevant credentials.

6.1 Mapping methods

The decision module maps credential attributes from the ontology context channel to credentials available in the repository. The attributes returned by the context channel correspond with attribute names and identifiers within digital credentials. For clarity, the terms *ontology attribute* and *credential attribute* are used in this chapter to distinguish between attributes returned by the ontology context channel and attributes found in digital credentials, respectively.

The simplest mapping method is to match ontology attributes and credential attributes. If a credential attribute is found in the set of ontology attributes, then it is considered relevant and included in the result subset.

For trust negotiation systems that support privacy enhancements such as fine-grained selective disclosure[1] [18], simple matching is sufficient because only the relevant credential attributes will be disclosed. However, for credentials that include multiple attributes, the release of one credential could disclose both relevant and irrelevant attributes.

A minimum/maximum inclusion mapping method provides greater control over

[1]Selective disclosure allows individual credential attribute disclosure without revealing all the attributes contained in a credential.

the number of attributes per credential that must be considered relevant before its release. The minimum inclusion approach classifies an entire credential as relevant if one or more attributes are relevant—this is equivalent to the simple matching method. Conversely, the maximum inclusion approach classifies a credential as relevant if and only if *all* of its attributes are relevant.

Additional fine-grained mapping methods are possible when ontology attributes contain extra content, like value restrictions. For example, if the ontology context channel returns an *age* attribute with a value restriction of ≥ 17, credentials that contain the attribute *age* are only considered relevant if the *age* value is greater than 16.

Each of the previous approaches returns boolean relevancy decisions. However, other mapping methods returning continuous ranges and thresholds may also be used to select relevant attributes. Common classifiers, likely based on fuzzy logic or machine learning, provide the ability to generalize and adapt to user preference over time, given appropriate user feedback. These advanced mapping methods are beyond the scope of this thesis.

6.2 Handling irrelevance

After classifying the appropriate credentials as relevant, the subset $\mathrm{C}_{relevant}$ is returned by the decision module. The set of credentials derived from a received policy $\mathrm{C}_{derived}$, are deemed relevant if they are a subset of $\mathrm{C}_{relevant}$. Otherwise the set defined by the difference $\mathrm{C}_{derived} - \mathrm{C}_{relevant}$ contains the irrelevant credentials. This section suggests several possible actions to take when irrelevant credentials are detected (i.e., $\mathrm{C}_{derived} - \mathrm{C}_{relevant} \neq \emptyset$).

The worst case occurs when $\mathrm{C}_{relevant}$ and $\mathrm{C}_{derived}$ are disjoint (i.e., all credentials in the set $C_{derived}$ are irrelevant). This situation dictates that the strongest action be taken. The most drastic action is to terminate the negotiation. Other options include notifying the user, allowing the negotiation to continue at the user's discretion, or using the received policy to derive a different credential set[2].

If the context alignment method (see Section 5.3) selects too specific a negotiation class, the result set $C_{derived}$ may contain many irrelevant credentials. To reduce the number of irrelevant credentials caused by improper context alignment, the decision module utilizes the trust negotiation ontology's general-to-specific organization. By increasing the generality of the selected negotiation class, fewer irrelevant credentials occur at the risk of generalizing beyond the scope of the actual negotiation context. Negotiation class generalization is performed by traversing the *is a* property of the current class in the trust negotiation ontology. The process repeats, each time selecting a negotiation context more general than the last, until a threshold is reached on the number of irrelevant credentials, the steps of generalization, or the user is satisfied.

Lastly, the detection of irrelevant credentials may direct the trust negotiation agent to report to an external module. This approach removes responsibility to handle irrelevancy from the trust agent, allowing feedback and negotiation decisions to be made by another entity (e.g., an access control module that uses global system suspicion levels [30]).

In all cases, the detection of irrelevant credentials should trigger some form of notification to the user. The decision module ultimately places relevancy determination in the user's hands. Clients are only interrupted when dramatic (more than a threshold of) credential irrelevancy is detected. Servers log relevancy results allowing auditing to uncover trends or problems in their configuration.

[2]The backtracking algorithm presented by Smith [35], derives all of the alternate satisfying credential sets for a given policy.

6.2.1 Providing feedback

Feedback is important to both negotiating parties when trust negotiations fail due to policy irrelevancy. Both server and client may utilize negative feedback (failure codes) to fix errors or better comply with typical policy practices (as established by the trust negotiation ontologies). Positive feedback should reinforce negotiation class preferences. Because feedback presents privacy concerns, client and server may customize the content of feedback responses. Basic feedback responses should include the negotiating agent's identifier, time/date, the trust negotiation ontology and negotiation class applied, and possibly the irrelevant credential attribute identifiers (each depending on privacy preferences).

Third party ontology providers will also depend on feedback to update trust negotiation ontologies over time. Thus, similar feedback reporting irrelevant credential classifications, lack of appropriate negotiation classes, or user surveys should be furnished. The ramifications and a detailed analysis of possible feedback messages are left for future work.

Chapter 7 — Implementation

This chapter presents the prototype implementation of the credential relevancy framework, the technologies used to build it, and a summary of its design decisions. The goal of the prototype implementation is twofold—demonstrate the feasability of creating a credential relevancy framework by extending an existing trust negotiation system, and constrain the computational overhead of that system to no more than an additional 10% compared to baseline trust negotiation.

7.1 Fundamental systems

The prototype implementation extends the functionality of TrustBuilder, a trust negotiation system. The following sections present an overview of TrustBuilder, Protégé, and Jena, three fundamental technologies used in the integration, design, and implementation of the credential relevancy framework.

7.1.1 TrustBuilder

TrustBuilder[1] allows two strangers to authenticate by exchanging credentials and policies [39]. A graphical illustration of TrustBuilder is shown in Figure 7.1.

TrustBuilder makes authentication decisions based on exchanged X.509v3 digital credentials [17] and TPL policies[2] [14]. TrustBuilder's heart, the compliance checker, decides if remote access control policies are satisfied by local credentials, and if remote credentials satisfy local policies. Together the compliance checker, credential verification module, and vault (containing local credentials and policies) enable the negotiation strategy to release the resource or disclose further credentials or policies.

[1]TrustBuilder was jointly conceptualized at Brigham Young University and the University of Illinois–Champaign, and developed by the Internet Security Research Lab at BYU.

[2]The IBM Alphaworks project *Trust Establishment* provides the basic components currently used by TrustBuilder: a compliance checker, the TPL policy language and X.509v3 credentials with special extensions.

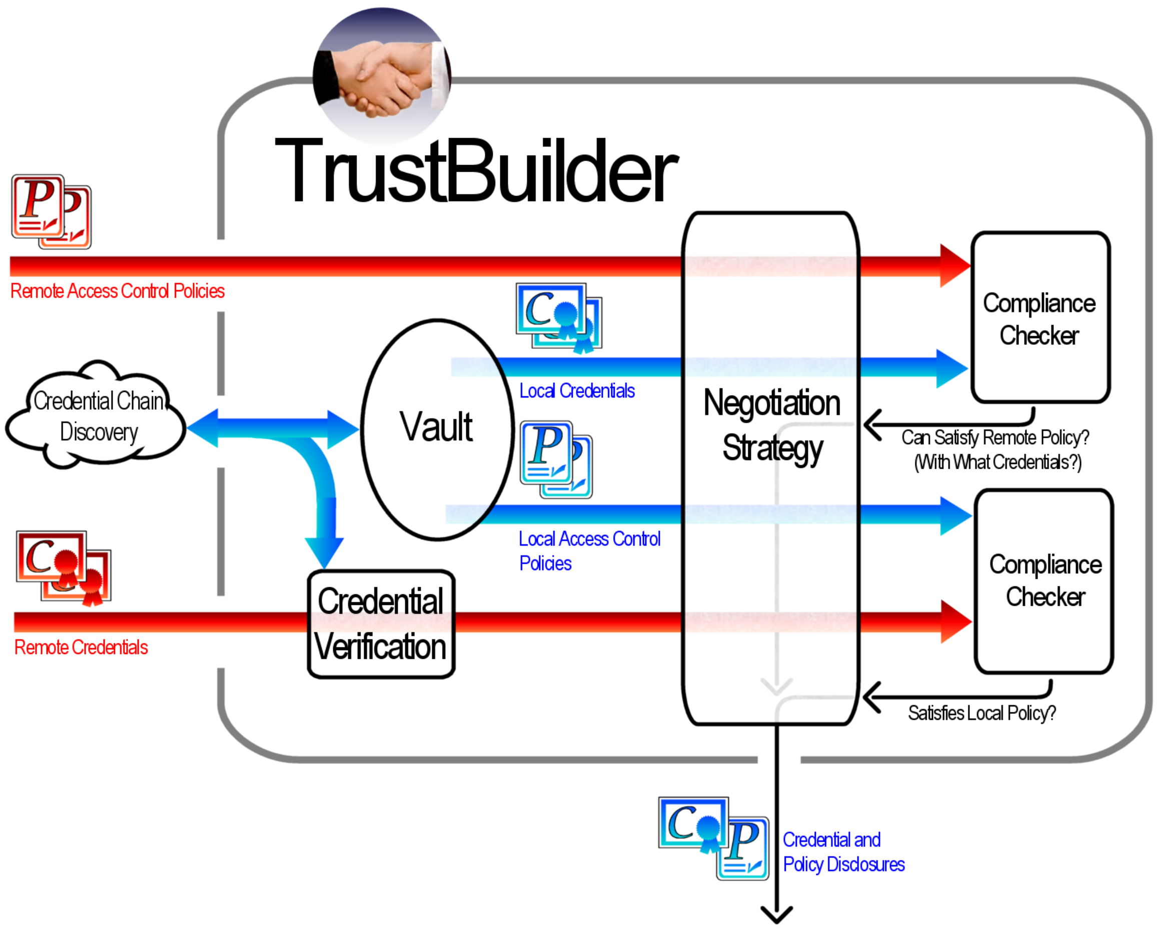

TrustBuilder
Remote Access Control Policies
Credential Chain Discovery
Vault
Local Credentials
Local Access Control Policies
Negotiation Strategy
Compliance Checker
Can Satisfy Remote Policy? (With What Credentials?)
Compliance Checker
Satisfies Local Policy?
Remote Credentials
Credential Verification
Credential and Policy Disclosures

Figure 7.1: The TrustBuilder authentication agent. Credentials and/or policies are received from another negotiation agent. A compliance checker receives local credentials to satisfy remote policies and remote credentials to satisfy local policies. Credential verification and compliance checker results are analyzed by the agent's negotiation strategy, which determines the local policies, credentials, or resources to disclose.

7.1.2 Protégé

Protégé[3] [12, 16, 28] is an ontology editor and knowledge-base framework created and maintained by Stanford Medical Informatics. Protégé provides a user interface for the graphical creation of classes, slots (properties), instances, and other supporting concepts. Modification of the ontology class hierarchy and its class properties (e.g., transitive, disjoint, functional, symmetric, etc.) are performed in the application's *classes* tab. Similar tabs make available the creation and modification of slots, introduction of instances, ontology visualization, etc. Protégé also includes a plugin for creating OWL ontologies that alters the default user interface as shown in the screen shot of Figure 7.2. The OWL ontology created for the prototype is graphically illustrated in Figure 7.3.

7.1.3 Jena

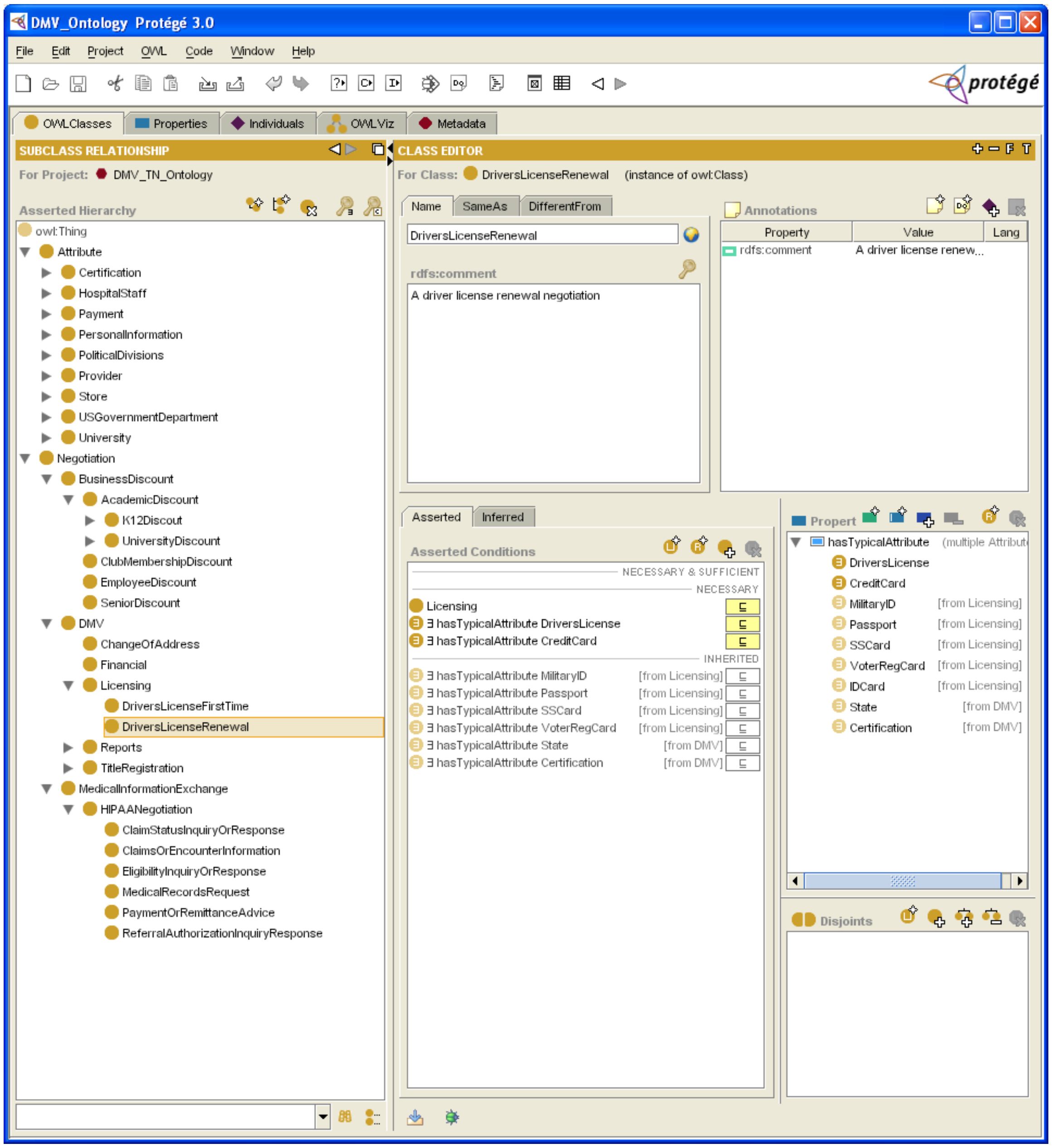
To query and retrieve meaningful information from the prototype OWL ontology, Jena[4], an open source semantic web framework, provides an API for loading, parsing, querying OWL ontologies with RDQL[5] and an inference engine for simple reasoning and information retrieval. Jena's API includes methods for working with RDF, OWL, and other semantic web languages in a programmatic environment.

RDQL, similar to OWL-QL [11] consists of collections of RDF statements (i.e., a subject–predicate–object triple) with optional variable substitution for any value in the triple. RDQL syntax is based loosely on SQL. For example, the query used to retrieve all the attributes for the negotiation class *DriverLicenseRenewal* in the prototype's OWL ontology is shown in Table 7.1.

[3]The ontology for this thesis was created using Protégé 3.0 with the OWL Plugin version 2.0 (see `http://protege.stanford.edu`).

[4]Jena 2.2 is an open source project, grown out of work from HP Labs' Semantic Web Programme (see `http://jena.sourceforge.net`).

[5]RDQL is an implementation of the SquishQL RDF query language, which itself is derived from rdfDB.



For Project: DMV_TN_Ontology

Asserted Hierarchy

```
owl:Thing
▼ Attribute
    ▶ Certification
    ▶ HospitalStaff
    ▶ Payment
    ▶ PersonalInformation
    ▶ PoliticalDivisions
    ▶ Provider
    ▶ Store
    ▶ USGovernmentDepartment
    ▶ University
▼ Negotiation
    ▼ BusinessDiscount
        ▼ AcademicDiscount
            ▶ K12Discout
            ▶ UniversityDiscount
          ClubMembershipDiscount
          EmployeeDiscount
          SeniorDiscount
    ▼ DMV
          ChangeOfAddress
          Financial
        ▼ Licensing
              DriversLicenseFirstTime
              DriversLicenseRenewal
        ▶ Reports
        ▶ TitleRegistration
    ▼ MedicalInformationExchange
        ▼ HIPAANegotiation
              ClaimStatusInquiryOrResponse
              ClaimsOrEncounterInformation
              EligibilityInquiryOrResponse
              MedicalRecordsRequest
              PaymentOrRemittanceAdvice
              ReferralAuthorizationInquiryResponse
```

For Class: DriversLicenseRenewal (instance of owl:Class)

DriversLicenseRenewal

rdfs:comment

A driver license renewal negotiation

Property	Value	Lang
rdfs:comment	A driver license renew...	

```
NECESSARY & SUFFICIENT
NECESSARY
Licensing                                                ⊑
∃ hasTypicalAttribute DriversLicense                     ⊑
∃ hasTypicalAttribute CreditCard                         ⊑
INHERITED
∃ hasTypicalAttribute MilitaryID      [from Licensing]   ⊑
∃ hasTypicalAttribute Passport        [from Licensing]   ⊑
∃ hasTypicalAttribute SSCard          [from Licensing]   ⊑
∃ hasTypicalAttribute VoterRegCard    [from Licensing]   ⊑
∃ hasTypicalAttribute State           [from DMV]         ⊑
∃ hasTypicalAttribute Certification   [from DMV]         ⊑
```

```
▼ hasTypicalAttribute   (multiple Attribut
    DriversLicense
    CreditCard
    MilitaryID          [from Licensing]
    Passport            [from Licensing]
    SSCard              [from Licensing]
    VoterRegCard        [from Licensing]
    IDCard              [from Licensing]
    State               [from DMV]
    Certification       [from DMV]
```

Figure 7.2: A screen shot of Protégé 3.0 with OWL plugin displaying a portion of the ontology class hierarchy developed for the prototype.

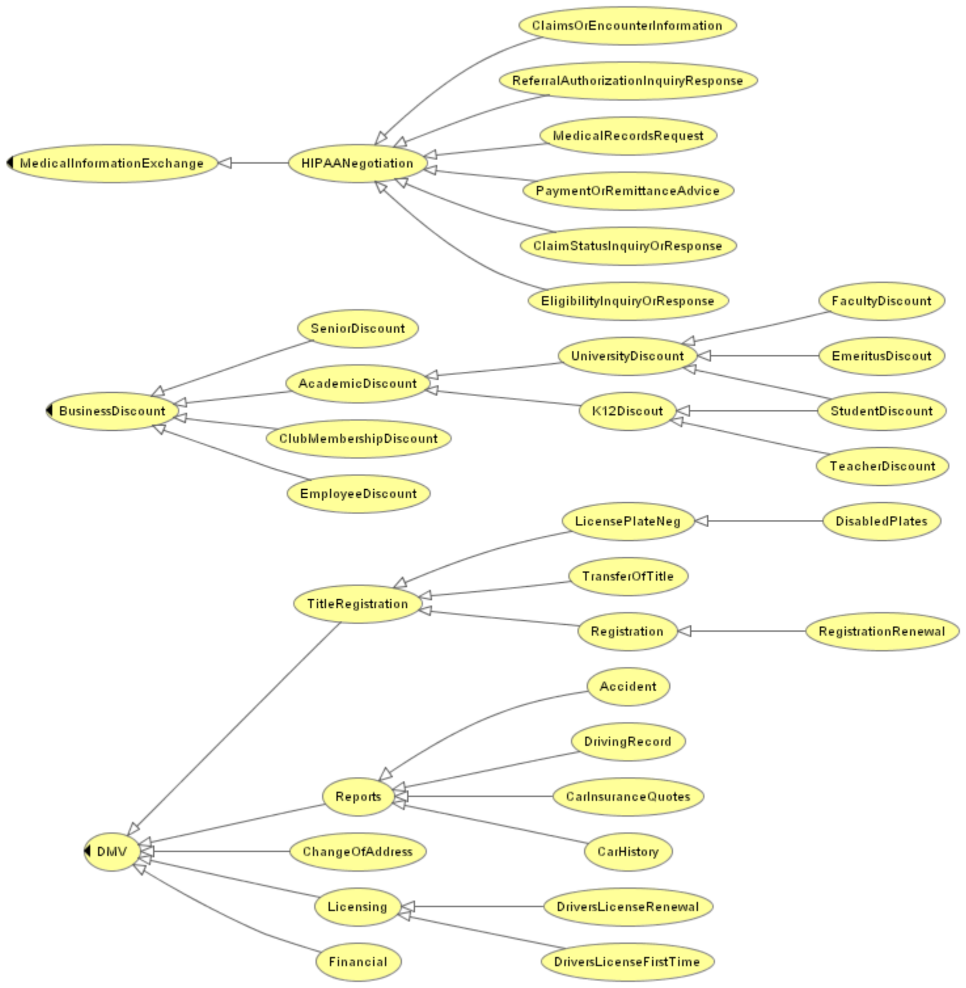


Figure 7.3: The class hierarchy graphically representing *is a* properties between classes for three trust negotiation ontologies. Other class interrelationships, such as the property *hasTypicalAttribute*, are not shown. The complete ontology description is included in Appendix A.

$$\begin{array}{ll}
\text{SELECT} & ?\mathit{attribs} \\
\text{WHERE} & \left(\begin{array}{l}
(?X,\ \text{rdf:type, owl:Restriction}) \\
(?X,\ \text{owl:onPropery},\ \mathit{tno}\text{:}\ \mathit{hasTypicalAttribute}) \\
(?X,\ \text{owl:someValuesFrom},\ ?\mathit{attribs}) \\
(\mathit{tno}\text{:}\ \mathit{DriverLicenseRenewal},\ \text{rdfs:subClassOf},\ ?X)
\end{array}\right) \\
\text{USING} & \mathit{tno}\ \text{FOR <http://isrl.cs.byu.edu/travisthesis/ontology.owl\#>}
\end{array}$$

Table 7.1: The RDQL query that retrieves the attributes for a *DriverLicenseRenewal* negotiation class along the property *hasTypicalAttribute*. The variable *?attribs* is assigned the query results based on each limiting conjunction in the WHERE clause. The USING statement associates a shortcut for the complete ontology namespace within the query.

Leveraging TrustBuilder, Protégé, and Jena, the credential relevancy framework can be implemented as an extension to TrustBuilder using Java.

7.2 Design decisions

Table 7.2 summarizes the implementation design decisions for the credential relevancy framework. Specific items are expounded below.

For simplicity, both client and server use the same ontology, which combines three trust negotiation ontologies and nine dependent attribute ontologies. The trust negotiation ontologies contain typical negotiation classes used for each motivating scenario from Chapter 1 (i.e., the DMV, medical records, and bookstore scenarios), focusing on the relevant negotiation described in each scenario (i.e., the driver license renewal, HIPAA medical record request, and student discount). Appendix A contains the complete OWL ontology specified in XML.

The optimization alluded to in Section 4.2.1 is employed to minimize the overhead of repeatedly computing $C_{relevant}$, hence the subset $C_{derived}$ is input to the decision module to determine the relevant subset.

Each scenario's policies are designed with varying levels of irrelevancy for per-

Design decisions	
Ontology design	To consolidate space and facilitate ease of development, all supporting attribute and trust negotiation ontologies were developed jointly and co-located in a single file.
Ontology language	OWL was chosen as the ontology language for the prototype because it is the most recent standard to be recommended by the W3C.
Ontology location	A third party webserver hosts the ontology, allowing each agent to query it through RDQL.
Ontology availability	For simplicity, both client and server reference the same ontology. The ontology is assumed previously authenticated with no expiration date.
Ontology caching	Jena loads the ontology entirely into memory at startup—therefore, all queries and results are cached. Local ontology caching (preservation of user preferences) is disabled.
Attribute ontology	Lack of global credential attribute definitions for TE credentials necessitates the creation of local definitions of attribute ontologies. Attribute ontology classes are attribute names that correspond to TE credential type values (see *mapping method* below).
Client context	The user interaction context alignment method allows for the client to select the most relevant negotiation class from any of the three trust negotiation ontologies at transaction commencement. This is done via the TrustBuilder configuration file.
Server context	Servers are pre-configured to use a specific negotiation class.
Mapping method	The decision module uses the minimal inclusion matching method to associate ontology attributes with TE credential types. (TE credential types are unique identifiers for each credential designed to have one-to-one correspondence to ontology attributes. Thus, the minimal inclusion matching method reduces to the simple matching method.)
Irrelevant credentials	For worst-case performance testing, the credential relevancy framework assumes that irrelevant or sensitive credentials are always disclosed. Therefore, no warning dialog is presented to the user. Irrelevant credential disclosures are automatically logged and negotiation continues.

Table 7.2: The prototype design decisions for creation of the credential relevancy framework extension to TrustBuilder.

formance testing (see Chapter 8). During a negotiation, the prototype assumes that the user always elects to release irrelevant credentials when presented with a choice. Thus performance measures in the next chapter represent a worst-case or upper bound on the execution time because trust negotiation always continues despite the detection of irrelevant credential requests. To simplify performance testing, irrelevancy is logged to a file instead of presented to the user for confirmation.

Chapter 8 — Analysis

This chapter presents the analysis of the prototype credential relevancy framework. Because the prototype is a proof-of-concept implementation, the analysis is not intended to showcase its efficiency or thoroughly examine the tradeoffs in design decision performance. Rather, it focuses on the feasibility of creating a credential relevancy framework, by demonstrating that it can be implemented with minimal impact to an existing baseline trust negotiation system. Minimal impact is defined as utilizing less than 10% of the total execution time for the combined system.

This chapter describes three scenarios used for performance testing, presents the total execution time results, illustrates an execution time profile of TrustBuilder, and estimates performance over larger and more complex trust negotiations.

8.1 Scenario setup

To validate the computational overhead of the credential relevancy framework, three trust negotiation scenarios were selected that span the range from large and complicated to small and simple. These three scenarios allow the generalization and extrapolation of the computational effects of the credential relevancy framework as negotiations get larger and more complex.

The first scenario is based on the DMV driver license renewal used throughout this thesis. The baseline DMV scenario[1] uses nineteen combined credentials in both repositories and nineteen policy groups[2] before granting the license renewal. The trust negotiation progression for this scenario is depicted in Figure 8.1-A.

[1]Each baseline scenario refers to a trust negotiation previously created in which all policies exchanged are assumed relevant to the negotiation.

[2]The number of policy groups within a TPL policy is a rough estimate of its complexity. Other contributing factors apply, such as role expression complexity, and number and complexity of possible combinations of satisfying sets.

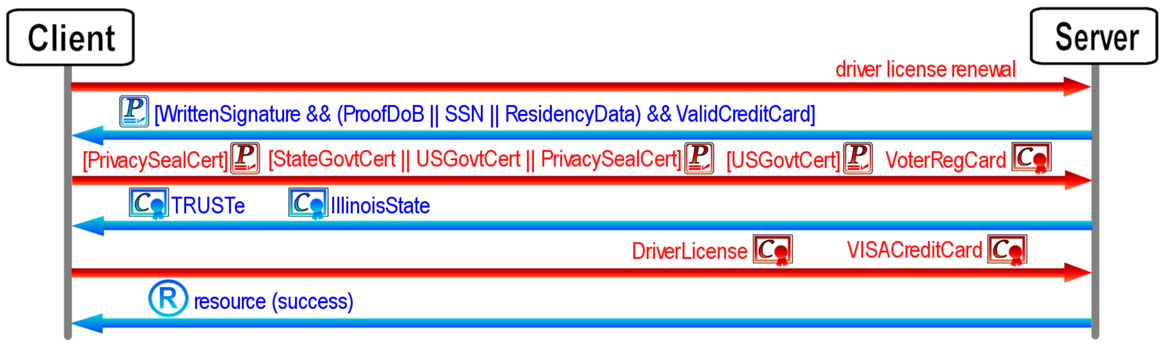
The second baseline scenario is a request for medical records. The medical records scenario employs fourteen credentials with eight policy groups. This baseline scenario progresses according to Figure 8.1-B.

Finally, Figure 8.1-C shows the baseline bookstore scenario—the simplest of the three scenarios. The student discount request is granted after three rounds of negotiation, in which nine total credentials and seven policy groups are considered.

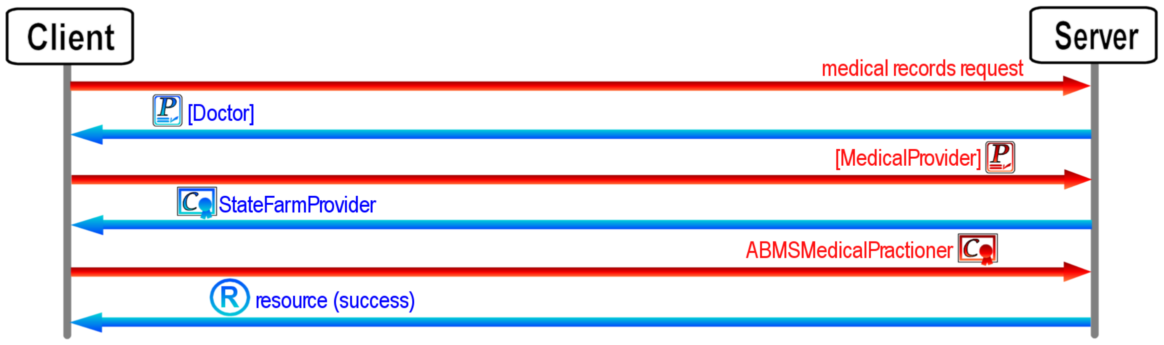
Three cases of potentially malicious negotiations supplement each of the three preceding scenarios. For each scenario, the existing baseline policies were modified to create two additional *semi* and *none* relevant cases. The third *all* relevant case (baseline) remained unchanged for comparison with the other two cases. These cases are used to extrapolate the effect that policy relevancy has on the credential relevancy framework, while the specific scenario remains constant. The modified semi- and none-relevant policies from each scenario along with illustrations describing the alternate negotiation progress are included in Appendix B.

To achieve accurate graphs of execution time, without factoring in time to send and receive trust negotiation messages, the JProfiler[3] tool was used to precisely analyze the quantity of time spent in each class and method of the credential relevancy framework extension and TrustBuilder. JProfiler was attached to both client and server such that only cumulative execution time on each negotiation agent applied to the results. The Java RMI protocol minimized communication overhead between the two agents during negotiation analysis. Figure 8.2 illustrates the profiling architecture.

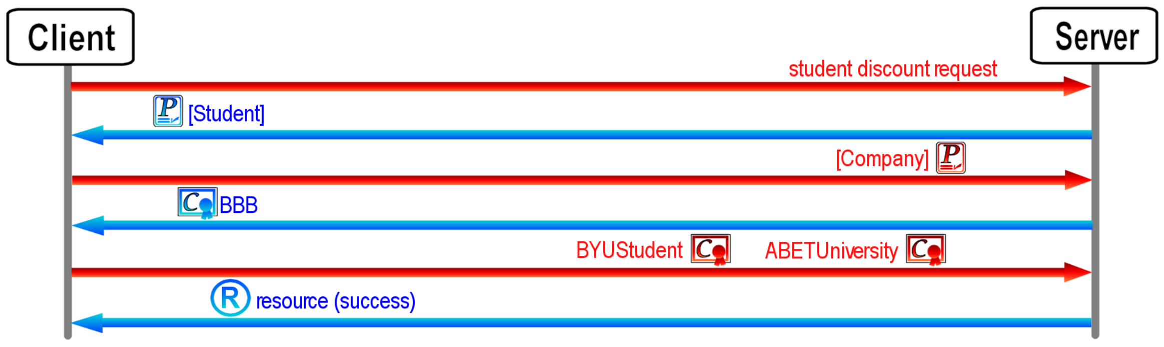
[3]JProfiler version 3.3.1, by *ej-technologies*, is an application and server profiling tool (see `http://www.ej-technologies.com/products/jprofiler/overview.html`).


Client
Server
driver license renewal
P [WrittenSignature && (ProofDoB || SSN || ResidencyData) && ValidCreditCard]
[PrivacySealCert] P [StateGovtCert || USGovtCert || PrivacySealCert] P [USGovtCert] P VoterRegCard C
C TRUSTe C IllinoisState
DriverLicense C VISACreditCard C
R resource (success)

(A) Baseline DMV driver license renewal negotiation



(B) Baseline medical records negotiation



(C) Baseline bookstore negotiation

Figure 8.1: Three baseline scenarios illustrating the negotiation progress for (A) the driver license renewal, (B) medical records, and (C) student discount negotiations. The policies exchanged include the role expressions that specify the logical combination of policy groups required to satisfy the policy. Role expressions are shown within square brackets.

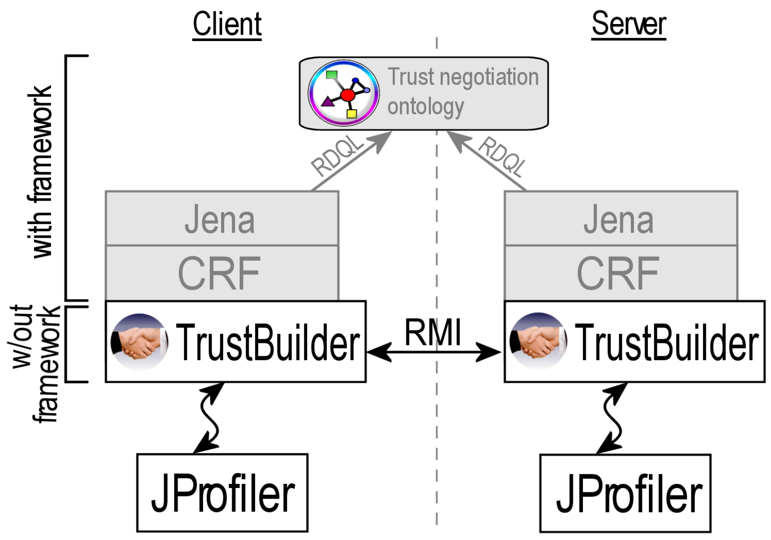


Figure 8.2: The profiling architecture for performance analysis. The profilers measured cumulative execution time on both client and server TrustBuilder (with and without the credential relevancy framework). The individual execution times for client and server are summed for the results of Table 8.1.

8.2 Results

Results were calculated locally on a single Pentium 4, 3.0 GHz processor with 1 GB of RAM running the Windows XP operating system.

Table 8.1 compares the total execution time of the three scenarios with their baseline, semi-relevant, and none-relevant cases to the total execution time of the same scenarios running the credential relevancy framework. Execution time measurements for TrustBuilder without the credential relevancy framework were gathered prior to any code modification. Additionally, the following list delimits other configuration details and settings used throughout the data collection process.

- Full debugging output ensured visual negotiation progress.
- Each execution time measurement is the sum of 50 interleaved negotiations. The server handled no more than ten simultaneous negotiations at a time, starting a new negotiation whenever another concluded. Thus, to obtain an estimate[4] of the average time per negotiation, divide the time by 50.

[4]The resultant average is the execution time for one trust negotiation given that nine others

Total execution time spent negotiating in milliseconds

Scenario name	Baseline w/out framework	Baseline with framework	Baseline percent increase	Semi-relevant w/out framework	Semi-relevant with framework	Semi-relevant percent increase
DMV	1,200,662	1,232,495	2.65%	402,528,908	402,738,766	0.05%
Medical records	561,363	576,508	2.7%	847,063	885,192	4.5%
Bookstore	445,039	486,766	9.38%	462,370	533,462	15.38%
Average over each scenario	**735,688**	**765,256**	**4.02%**	**134,612,780**	**134,719,140**	**0.08%**

Scenario name	None-relevant w/out framework	None-relevant with framework	None-relevant percent increase	Average over each case w/out framework	Average over each case with framework	Average over each case percent increase
DMV	1,126,793	1,163,724	3.28%	**134,952,121**	**135,044,995**	**0.07%**
Medical records	900,141	986,385	9.58%	**769,522**	**816,028**	**6.04%**
Bookstore	414,965	447,366	7.81%	**440,791**	**489,198**	**10.98%**
Average over each scenario	**813,966**	**865,825**	**6.37%**	**45,387,478**	**45,450,074**	**0.14%**

Table 8.1: Comparison of total execution time for TrustBuilder with and without utilizing the credential relevancy framework (CRF).

- Each measurement represents the sum total of the execution time spent in the client and server TrustBuilder agents (including the credential relevancy framework as indicated). No RMI message passing overhead is included.
- The percent increase is calculated as $\frac{(\text{with framework} - \text{w/out framework})}{\text{w/out framework}}$
- For all credential relevancy framework measurements, both client and server use the framework and reference the same trust negotiation ontology. A Tomcat 4 server, running locally, hosts the trust negotiation ontology.

From the results tabulated in Table 8.1, only the bookstore scenario in the *semi* relevant case incurred more than the goal of 10% additional computation overhead. This anomaly is explained due to the high variability of simple trust negotiations spending less time negotiating in general due to their simple policies and few credentials.

A more typical trend can be seen in the moderately complex medical records scenario. As policies become less relevant, execution time in the credential relevancy

are also executing concurrently.

framework increases from the baseline's 2.7% to 9.58% in the *none* relevant case. This is more typical of the expected trend.

Averages for the *semi* relevant case are largely skewed because of the exceptionally long time spent negotiating in the DMV scenario. This occurred because the client was forced to initiate the backtracking algorithm (see Smith et al. [35]) to generate all the possible combinations of satisfying credential sets[5]. This time-consuming algorithm minimizes the impact of the credential relevancy algorithm in this instance.

8.3 Execution time profile

In order to further visualize the impact of the credential relevancy framework on TrustBuilder, the total execution times from Table 8.1 are broken down by scenario to show the percentage of time spent in each of TrustBuilder's modules. Figures 8.3, 8.4, and 8.5, graph the percent of execution time spent in TrustBuilder's significant modules[6] for the driver license, medical records, and bookstore scenario, respectively. Figure 8.6 averages the percentages reported in the previous three figures to show a generalized summary.

As shown in Figure 8.6, the credential relevancy framework itself incurs less overhead on trust negotiation than indicated by the total execution time table. Averaging the baseline, *semi* and *none* relevant cases for the credential relevancy framework module of Figure 8.6 yields an average impact of 2.22% overall.

The cryptography processes involved in trust negotiation (i.e., certificate signing/verifying, certificate hashing, credential ownership verification, etc.) and logic contained in the negotiation strategy generally consume the most execution time

[5]The exact proceeding of the DMV *semi* relevant negotiation is illustrated in Figure B.1 from Appendix B.

[6]Total method and class execution time for less-significant modules are aggregated into a general module category for convenience in graphing.

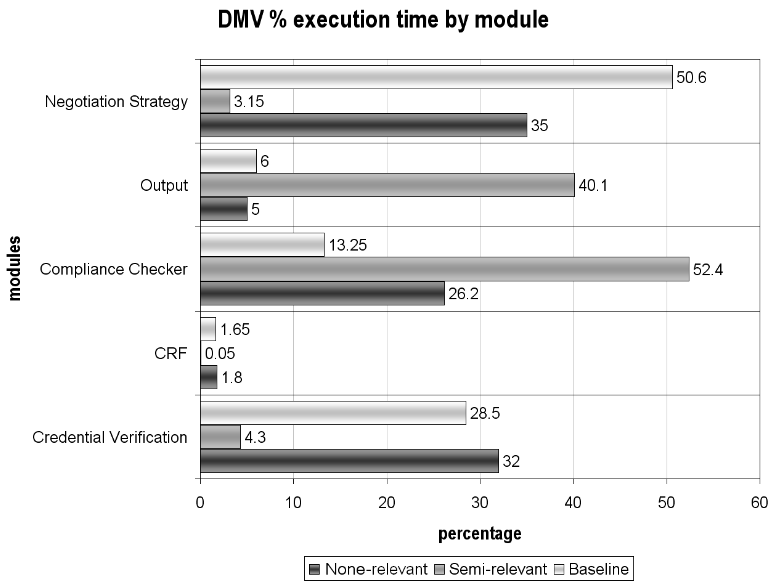


Figure 8.3: Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the DMV scenario.

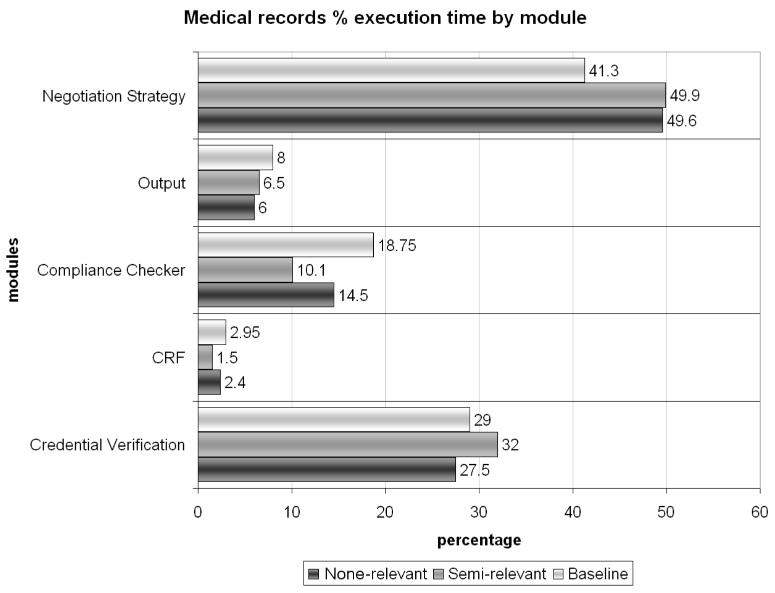


Figure 8.4: Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the medical records scenario.

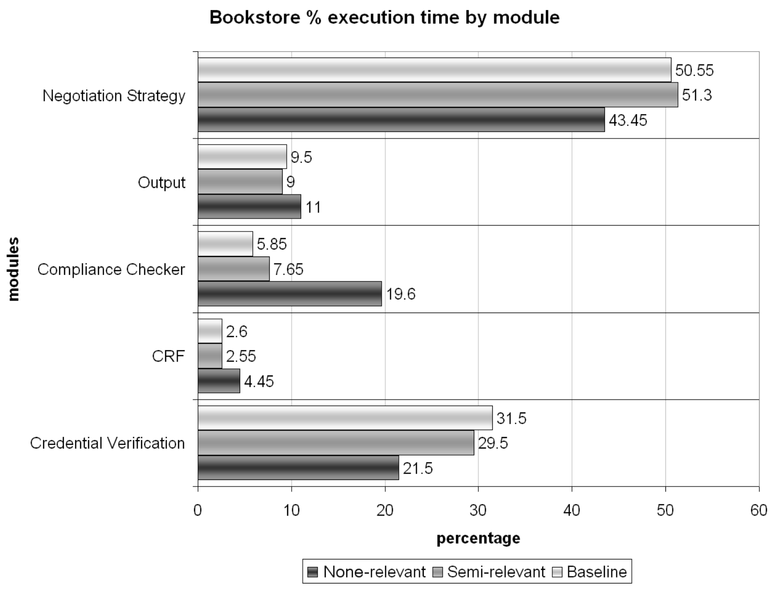


Figure 8.5: Comparison of five modules in TrustBuilder and their percentage of execution time for each relevancy case in the bookstore scenario.

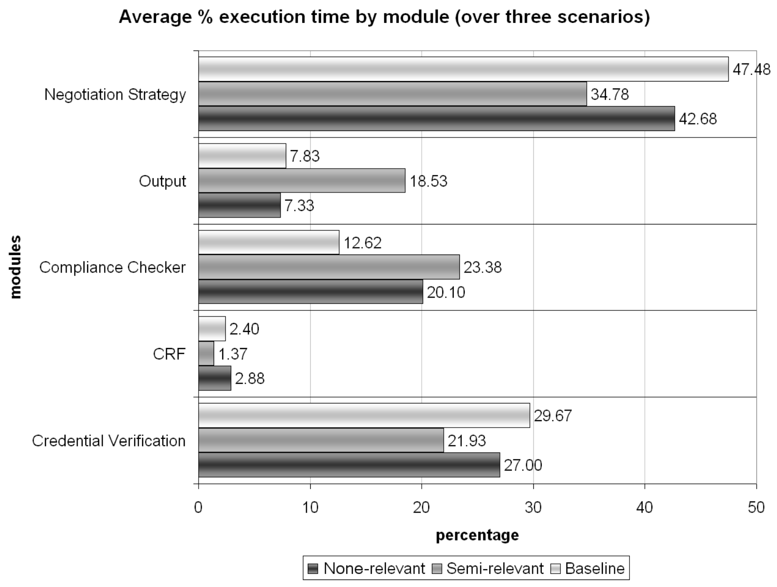


Figure 8.6: Averages over the DMV, medical records, and bookstore scenarios for each of the five modules in TrustBuilder.

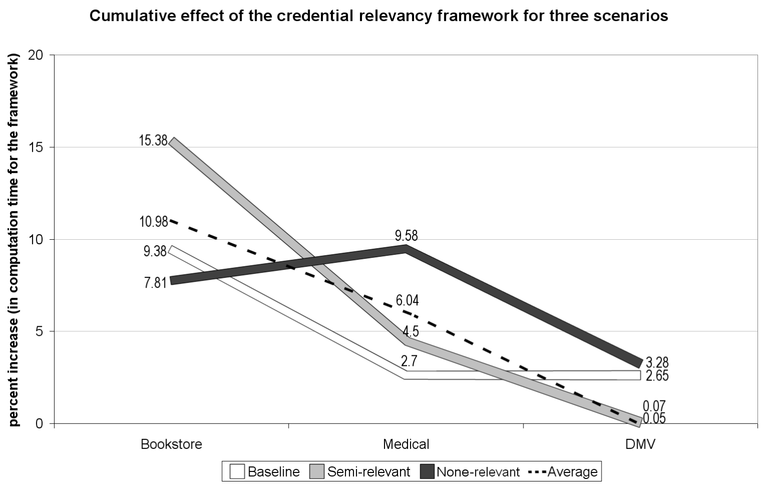


Figure 8.7: Trend graph based on data from Table 8.1, showing that as negotiations get more complex, the overall computational effect on the baseline trust negotiation decreases.

in TrustBuilder. As before, the DMV semi-relevant case in Figure 8.3, is largely skewed due to the backtracking algorithm. As shown, the backtracking algorithm spends the most time in the compliance checker (and also generating output when debugging is enabled).

8.4 Generalizations

An estimate of the performance of larger, more complex negotiations can be made based on previous results. Figure 8.7 illustrates the downward trend of the percentage of execution time spent in the credential relevancy framework as negotiations increase in complexity. This is not surprising considering the credential relevancy framework's small impact on the overall negotiation (see Figure 8.6). As negotiations get more complex more time is spent in the negotiation strategy, compliance checker, and credential verification modules.

The downward trend would seem to suggest that as negotiations become larger the effect of the credential relevancy framework would dwindle to nothing. However, this trend fails to consider an increase in trust negotiation ontology complexity.

With increased complexity, ontology queries take longer to complete and network overhead caused by those queries become a concern (e.g., because of the distributed nature of ontologies). Additional factors, such as the complexity of the context alignment method, can contribute to the overall impact of the credential relevancy framework. Contrary to the trend visible in Figure 8.7, it is expected that the effect of the CRF will hold relatively constant or slightly increase as negotiations become more complex, due to the other factors previously mentioned.

Chapter 9 — Conclusions and Future Work

Trust negotiation is the process of building trust between strangers in open systems through the exchange of digital credentials and policies. Negotiating agents are presently capable of verifying the relevance of received credentials, but lack sufficient context to verify policy relevance.

Previous research assumes that policies "bargain in good faith" by requesting credentials that further the negotiation progress. This thesis challenges that assumption by suggesting that policies cannot be guaranteed to "play fair" in actual open systems. Policies that do not "play fair" naively or intensionally request information that exploits automated trust negotiation by causing the disclosure of sensitive or unrelated credentials. Trust negotiation authentication systems cannot be trusted while this open avenue for attack exists. To ensure the security of trust negotiation, protection from irrelevant or malicious policies must exist.

Additional context is required to protect users from policies that do not "play fair." However, current policies lack sufficient context to establish their relevance to a negotiation. Presently, policies establish context for credentials, but no context currently exists to establish policy relevance.

This thesis presents the credential relevancy framework as one solution providing relevant external context. The framework utilizes trust negotiation ontologies that contain typical negotiation classes and attributes. Trust negotiation ontologies and credential repositories are used to derive a subset of relevant credentials.

Finally, a prototype demonstrates the framework's feasibility. An analysis of the prototype shows that its cumulative effect on trust negotiation is minor (less than 10% extra computational overhead).

9.1 Contributions

This thesis makes several important contributions to trust negotiation and computer science. It raises awareness of policies that do not "play fair," provides a prototype framework from which to base future systems, broadens the scope of what is considered relevant, conceptualizes a new approach to using ontologies to provide context, and promotes correct policy design.

This work identifies an area of trust negotiation that has been largely ignored—the consequences of irrelevant or malicious policies. Malicious policies cause *phishing* attacks (stealing sensitive information) or request information that is not relevant to the negotiation, compromising user privacy and security. By exposing this problem and proposing a solution, current and future trust negotiation systems may better protect themselves from irrelevant and malicious policies.

The prototype credential relevancy framework is the first to incorporate sufficient context information to allow discrimination between relevant and irrelevant credential solicitations in trust negotiation. The prototype respects the automation and transparency goals of trust negotiation by including several methods of context alignment for verifying policy relevance.

Trust negotiation literature previously used the term *relevant* only to relate a set of credentials to a policy. This work broadens the meaning of *relevant* to include policies and their relation to negotiation classes.

This thesis extends the published ideas of Leithead et al. [23] by formally defining trust negotiation ontology structure. Merging ontologies and trust negotiation opens a novel area of future research.

Trust negotiation ontologies encourage correct policy design by modeling the typical attributes requested during a trust negotiation, according to third party domain experts. Adherence to authoritative trust negotiation ontologies helps to standardize and promote correct policy design and encourages widespread policy unity.

9.2 Future work

This thesis pioneers the synthesis of trust negotiation and ontology research. As a relatively new combined research area, substantial future work remains. Future work falls into two categories—independent maturing technologies and the application of the credential relevancy framework.

9.2.1 Maturing technologies

Ontology research is a large and active field [7]. This thesis incorporates many innovations and techniques for ontology creation, maintenance, deployment, query, and knowledge encapsulation. Advances in ontology research may provide new solutions within the credential relevancy framework.

This thesis suggests the use of digital signatures to cryptographically sign and verify the content of an ontology as authored and issued from a third party provider. However, the digital signature itself only conveys trust in the integrity and authenticity of the content. Verification of the correctness of the ontology itself is an open challenge. Because trust negotiation ontologies encapsulate human perceptions of typical relevant attributes, their *correctness* may be impossible to certify. An approximation of ontology correctness, according to the general public or other domain experts, may be obtained using a social reputation system.

Web of trust reputation systems offer the most plausible and practical solution for auditing ontology correctness. Because trust negotiation ontologies are domain specific, their independent rating by frequent users or domain experts supplements content credibility and augments public trust.

Size and scalability analysis for trust negotiation ontologies offer another area of future research. As trust negotiation ontology content grows, how will the performance of query results be affected? How should large ontologies be presented to the user for negotiation class selection when user interaction context alignment is used? Research into machine-assisted decision making will become increasingly important as ontologies become increasingly large and complex.

Semantic web research is closely tied to the concepts of the credential relevancy framework. The semantic web consists of a multi-layered collection of technologies ranging from a foundation of *URI*s and *XML* to *proof* and *trust* in the upper layers. Trust negotiation, extended with the credential relevancy framework, could integrate into the proof and trust layers of the semantic web. For example, a trust negotiation-enabled semantic web agent would utilize the semantic web's underlying technologies such as logic proofs to satisfy policies, and ontologies to make relevancy decisions automatically.

Investigation of alternative sources of context for trust negotiation may lead to novel approaches for determining relevancy. In theory, alternative context information merged with current policies may produce *enhanced* policies capable of providing proof of their own relevance.

9.2.2 Aspects of the credential relevancy framework

Additional future work exists within the credential relevancy framework itself, including: combined digital credential-ontologies, the exchange of user preferences, advanced context alignment techniques, analysis of current irrelevancy actions, and feedback techniques.

Digital credentials used in TrustBuilder currently contain attributes that are distinct from the attributes contained in attribute ontologies (see Section 5.1). Future work may explore the idea of merging the logic, expressive power, and reasoning capabilities of ontologies with digital credentials. In theory, *smart* digital credentials may contain the context necessary for equipped trust agents to automatically derive meaning from their attributes.

Section 2.3 presents four possible sources for locating a source of relevancy context. This thesis chose to focus on drawing context source from a third party. However, a peer-to-peer negotiation context exchange system (e.g., using social reputation systems) is a very practical alternative to the third party approach. Ontologies for encapsulating negotiation context may yet be important to such an exchange system.

The context channel caches ontologies because an active connection to the provider is not always available (see Section 5.2). These cached ontologies are maintained separately or merged into a single local ontology. One stated advantage for maintaining a local ontology is to facilitate the exchange of user preferences with other trust agents. User preferences may act as an auxiliary source of trust when exchanged in a reputation system. Current reputation systems recognize the need for context-based trust metrics, and local ontologies already bind negotiation context to each user preference.

Context alignment involves inferring the client's negotiation context and choosing an appropriate negotiation class before a transaction begins (see Section 5.3). The problem of aligning a user's perception to a machine-understandable classification is an open research problem. Future work may exploit expert knowledge in the disciplines of artificial intelligence, machine learning and human-computer interaction to provide new methods for context alignment.

Further study of the actions for handling irrelevant credentials is needed (see

Section 6.2). What response to irrelevant credential requests best suit client or server, and how should trust agents respond?

Additionally, feedback is important to clients, servers, and third party ontology providers (see Section 6.2.1). Considerations for feedback responses include: format, frequency, and protocol for secure transmission.

Chapter 4 introduces two additional use cases for the credential relevancy framework: predisposed negotiation and relevant credentials first. While each case is illustrated by example, the implementations of these systems were not created nor analyzed in this thesis.

References

[1] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, Cambridge, MA, 2004.

[2] M. Y. Becker and P. Sewell. Cassandra: Distributed Access Control Policies with Tunable Expressiveness. In *POLICY '04: Proceedings of the 5th International Workshop on Policies for Distributed Systems and Networks*, pages 159–168, Yorktown Heights, NY, June 2004. IEEE Computer Society Press.

[3] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-X: A Peer-to-Peer Framework for Trust Establishment. *IEEE Transactions on Knowledge and Data Engineering*, 16(7):827–842, July 2004.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. *The KeyNote Trust-Management System, Version 2*. The Internet Society, RFC 2704, September 1999.

[5] P. A. Bonatti and P. Samarati. A Unified Framework for Regulating Service Access and Information Release on the Web. *Journal of Computer Security*, 10(3):241–271, September 2002.

[6] R. Bradshaw, J. Holt, and K. E. Seamons. Concealing Complex Policies with Hidden Credentials. In *CCS '04: Proceedings of the Eleventh ACM Conference on Computer and Communications Security*, pages 146–157, Washington, DC, October 2004.

[7] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins. What Are Ontologies, and Why Do We Need Them? *IEEE Intelligent Systems*, 14(1):20–26, January/Feburary 1999.

[8] E. Child. Trust Negotiation Using Hidden Credentials. Master's thesis, Brigham Young University, Provo, UT, July 2004.

[9] M. J. Covington, W. Long, S. Srinivasan, A. Dey, M. Ahamad, and G. Abowd. Securing Context-Aware Applications Using Environment Roles. In *SACMAT '01: Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, pages 10–20, Chantilly, VA, May 2001.

[10] D. W. Embley. Toward Semantic Understanding—An Approach Based on Information Extraction Ontologies. In *CRPIT '27: Proceedings of the Fifteenth Australasian database conference*, pages 3–12, Dunedin, New Zealand, 2004. Australian Computer Society, Inc.

[11] R. Fikes, P. J. Hayes, and I. Horrocks. OWL-QL - A Language for Deductive Query Answering on the Semantic Web. *Journal of Web Semantics*, 2(1):19–29, December 2004.

[12] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. Technical report, Stanford Medical Informatics, 2002.

[13] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[14] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 2–14, Oakland, CA, May 2000. IEEE Computer Society.

[15] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden Credentials. In *WPES '03: Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society*, pages 1–8, Washington, DC, October 2003.

[16] M. Horridge, H. Knublauch, A. Rector, R. Stevens, and C. Wroe. *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools*. The University of Manchester, Manchester, United Kingdom, 1st edition, August 2004.

[17] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. The Internet Society, RFC 3280, April 2002.

[18] R. Jarvis. Protecting Sensitive Credential Content During Trust Negotiation. Master's thesis, Brigham Young University, Provo, UT, April 2003.

[19] T. Jim. SD3: A Trust Management System with Certified Evaluation. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115, Oakland, CA, May 2001. IEEE Computer Society.

[20] R. Khare and A. Rifkin. Weaving a Web of Trust. *World Wide Web*, 2(3):77–112, Summer 1997.

[21] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*. The Internet Society, RFC 1510, September 1993.

[22] J. Lee, R. Goodwin, R. Akkiraju, A. Ranganathan, K. Verma, and S. Goh. *Towards Enterprise-Scale Ontology Management*. IBM T. J. Watson Research Center, Hawthorne, NY, 2003.

[23] T. Leithead, W. Nejdl, D. Olmedilla, K. Seamons, M. Winslett, T. Yu, and C. Zhang. How to Exploit Ontologies in Trust Negotiation. In *ISWC '04: Workshop on Trust, Security, and Reputation on the Semantic Web*, volume 127 of *CEUR Workshop Proceedings*, Hiroshima, Japan, November 2004. Technical University of Aachen (RWTH).

[24] 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*, pages 92–103, Monterey, CA, June 2002. IEEE Computer Society.

[25] S. Lu, M. Dong, and F. Fotouhi. The Semantic Web: Opportunities and Challenges for Next-Generation Web Applications. *Information Research*, 7(4), July 2002. Available at: http://informationr.net/ir/7-4/paper134.html.

[26] W. Nejdl, D. Olmedilla, and M. Winslett. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In *SDM '04: Workshop on Secure Data Management in a Connected World*, pages 118–132, Toronto, Canada, August 2004.

[27] I. Niles and A. Pease. Towards a Standard Upper Ontology. In *FOIS '01: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems*, pages 2–9, Ogunquit, ME, October 2001.

[28] N. F. Noy and D. L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford University, Stanford, CA, 2002.

[29] J. Reagle. *XML Signature Requirements.* The Internet Society, RFC 2807, July 2000.

[30] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. E. Seamons. Adaptive Trust Negotiation and Access Control. In *SACMAT '05: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*, pages 139–146, Stockholm, Sweden, June 2005.

[31] K. E. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation. In *NDSS '01: Network and Distributed System Security Symposium*, pages 109–124, San Diego, CA, February 2001.

[32] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for Policy Languages for Trust Negotiation. In *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 68–79, Monterey, CA, June 2002. IEEE Computer Society Press.

[33] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-line Trust Negotiation. In *PET '02: Proceedings of the 2nd Workshop on Privacy Enhancing Technologies*, pages 129–143, San Francisco, CA, April 2002.

[34] H. Skogsrud, B. Benatallah, and F. Casati. Model-Driven Trust Negotiation for Web Services. *IEEE Internet Computing*, 7:45–52, November/December 2003.

[35] B. Smith, K. E. Seamons, and M. D. Jones. Responding to Policies at Runtime in TrustBuilder. In *POLICY '04: Proceedings of the 5th International Workshop on Policies for Distributed Systems and Networks*, pages 149–158, Yorktown Heights, NY, June 2004.

[36] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In *ISWC '03: Proceedings of the 2nd International Semantic Web Conference*, pages 419–437, Sanibel Island, FL, October 2003.

[37] W3C. *OWL Web Ontology Language Use Cases and Requirements*, February 2004. http://www.w3.org/TR/webont-req/.

[38] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated Trust Negotiation. In *DISCEX '00: Proceedings of the DARPA Information Survivability Conference and Exposition*, volume 1, pages 88–102, Hilton Head, SC, January 2000.

[39] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing*, 6(6):30–37, November/December 2002.

[40] C. Wullems, M. Looi, and A. Clark. Towards Context-aware Security: An Authorization Architecture for Intranet Environments. In *PerCom '04: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pages 132–137, Orlando, FL, March 2004. IEEE Computer Society.

[41] B. Yu and M. P. Singh. A Social Mechanism of Reputation Management in Electronic Communities. In *CIA '00: Proceedings of Fourth International Workshop on Cooperative Information Agents*, pages 154–165, Boston, MA, 2000.

[42] T. Yu and M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 110–122, Oakland, CA, May 2003. IEEE Computer Society.

[43] T. Yu, M. Winslett, and K. E. Seamons. Interoperable Strategies in Automated Trust Negotiation. In *CCS '01: Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 146–155, Philadelphia, PA, November 2001.

Appendix A — OWL Ontology Source

This appendix contains the OWL ontology source for the prototype implementation of the credential relevancy framework. The complete ontology is broken up into sections based on the separate attribute and trust negotiation ontologies for easy reference.

A.1 Header information

```
<?xml version="1.0"?> <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://isrl.cs.byu.edu/travisthesis/ontology.owl#"
  xml:base="http://isrl.cs.byu.edu/travisthesis/ontology.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:about="#Attribute">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    Master super-class for all attributes
    </rdfs:comment>
  </owl:Class>
  <owl:TransitiveProperty rdf:about="#hasTypicalAttribute">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >describes the attributes that are typical of this class</rdfs:comment>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#Attribute"/>
    <rdfs:domain rdf:resource="#Negotiation"/>
  </owl:TransitiveProperty>
```

A.2 Supplementary attribute ontologies

A.2.1 HospitalStaff

```
  <owl:Class rdf:ID="Billing">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="HospitalStaff"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MedicalPractitioner">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HospitalStaff"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DeskEmployee">
    <rdfs:subClassOf rdf:resource="#Billing"/>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Records"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="NurseRS">
```

```
    <rdfs:subClassOf rdf:resource="#MedicalPractitioner"/>
  </owl:Class>
  <owl:Class rdf:ID="MidwifeCNM">
    <rdfs:subClassOf rdf:resource="#MedicalPractitioner"/>
  </owl:Class>
  <owl:Class rdf:ID="Janitorial">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HospitalStaff"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DoctorMD">
    <rdfs:subClassOf rdf:resource="#MedicalPractitioner"/>
  </owl:Class>
  <owl:Class rdf:about="#Records">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HospitalStaff"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#HospitalStaff">
    <rdfs:subClassOf rdf:resource="#Attribute"/>
  </owl:Class>
```

A.2.2 University

```
  <owl:Class rdf:about="#University">
    <rdfs:subClassOf rdf:resource="#Attribute"/>
  </owl:Class>
  <owl:Class rdf:ID="Faculty">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CurrentMember"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Professor">
    <rdfs:subClassOf rdf:resource="#Faculty"/>
  </owl:Class>
  <owl:Class rdf:ID="Salary">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="UniversityEmployed"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PartTime">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Hourly"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="AssistantTeachingProfessor">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="TeachingProfessor"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Graduate">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Student"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#CurrentMember">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="University"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#UniversityEmployed">
    <rdfs:subClassOf rdf:resource="#CurrentMember"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Alumni">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#University"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AssistantProfessor">
  <rdfs:subClassOf rdf:resource="#Professor"/>
</owl:Class>
<owl:Class rdf:ID="AssociateTeachingProfessor">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#TeachingProfessor"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FullTime">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Hourly"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="AssociateProfessor">
  <rdfs:subClassOf rdf:resource="#Professor"/>
</owl:Class>
<owl:Class rdf:ID="PostGraduate">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Student"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Hourly">
  <rdfs:subClassOf rdf:resource="#UniversityEmployed"/>
</owl:Class>
<owl:Class rdf:about="#TeachingProfessor">
  <rdfs:subClassOf rdf:resource="#Professor"/>
</owl:Class>
<owl:Class rdf:ID="Undergraduate">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Student"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Student">
  <rdfs:subClassOf rdf:resource="#CurrentMember"/>
</owl:Class>
```

A.2.3 Certification

```
<owl:Class rdf:ID="Certification">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Attribute"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#BusinessCertification">
  <rdfs:subClassOf rdf:resource="#Certification"/>
  <owl:disjointWith rdf:resource="#GovernmentCertification"/>
</owl:Class>
<owl:Class rdf:ID="CiscoProfessional">
  <owl:disjointWith>
    <owl:Class rdf:ID="CiscoExpert"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="CiscoCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="CiscoAssociate"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#ProjectPlus">
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
    <owl:disjointWith rdf:resource="#NetworkPlus"/>
    <owl:disjointWith rdf:resource="#ServerPlus"/>
    <owl:disjointWith rdf:resource="#APlus"/>
    <owl:disjointWith rdf:resource="#e-BizPlus"/>
    <owl:disjointWith rdf:resource="#i-NetPlus"/>
    <owl:disjointWith rdf:resource="#HTIPlus"/>
    <owl:disjointWith rdf:resource="#SecurityPlus"/>
    <owl:disjointWith rdf:resource="#CDIAPlus"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#CTTPlus"/>
  </owl:Class>
  <owl:Class rdf:ID="MCDST">
    <owl:disjointWith>
      <owl:Class rdf:ID="MCSA"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="MicrosoftCertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="MCSE"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MCAD"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MCT"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MCDBA"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MCSD"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="PrivacySealCertification">
    <owl:disjointWith>
      <owl:Class rdf:ID="ITCertification"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="BusinessCertification"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="LinuxPlus">
    <owl:disjointWith>
      <owl:Class rdf:ID="APlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="SecurityPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="HTIPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="e-BizPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="CTTPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="NetworkPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="ProjectPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="CDIAPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="ServerPlus"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="CompTIACertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="i-NetPlus"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#MCSA">
  <owl:disjointWith>
    <owl:Class rdf:about="#MCSE"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#MCDST"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#MicrosoftCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCAD"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCT"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCSD"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCDBA"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#MCSE">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#MicrosoftCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCAD"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCDBA"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#MCDST"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCT"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCSD"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#MCSA"/>
</owl:Class>
<owl:Class rdf:ID="CCNP">
  <owl:disjointWith>
    <owl:Class rdf:ID="CCIP"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="CCDP"/>
    </owl:disjointWith>
    <rdfs:subClassOf rdf:resource="#CiscoProfessional"/>
    <owl:disjointWith>
      <owl:Class rdf:ID="CCSP"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#HTIPlus">
    <owl:disjointWith>
      <owl:Class rdf:about="#CTTPlus"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#i-NetPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#NetworkPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#e-BizPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#CDIAPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#APlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#SecurityPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#ServerPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
  </owl:Class>
  <owl:Class rdf:about="#MCAD">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#MicrosoftCertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#MCDST"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#MCDBA"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#MCSA"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#MCSD"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#MCT"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#MCSE"/>
  </owl:Class>
  <owl:Class rdf:ID="USCertification">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GovernmentCertification"/>
    </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="StateCertification"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="CCNA">
  <owl:disjointWith>
    <owl:Class rdf:ID="CCDA"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CiscoAssociate"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#CCSP">
  <owl:disjointWith>
    <owl:Class rdf:about="#CCDP"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#CiscoProfessional"/>
  <owl:disjointWith rdf:resource="#CCNP"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#CCIP"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#CDIAPlus">
  <owl:disjointWith rdf:resource="#HTIPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#NetworkPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#e-BizPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#APlus"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CompTIACertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#ProjectPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ServerPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#i-NetPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#CTTPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SecurityPlus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#LinuxPlus"/>
</owl:Class>
<owl:Class rdf:about="#ServerPlus">
  <owl:disjointWith rdf:resource="#CDIAPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#e-BizPlus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#HTIPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SecurityPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
```

```
      <owl:Class rdf:about="#i-NetPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#APlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#NetworkPlus"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#CTTPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
  </owl:Class>
  <owl:Class rdf:about="#GovernmentCertification">
    <rdfs:subClassOf rdf:resource="#Certification"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#BusinessCertification"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#NetworkPlus">
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
    <owl:disjointWith rdf:resource="#CDIAPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#e-BizPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#CTTPlus"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#i-NetPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#APlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#ServerPlus"/>
    <owl:disjointWith rdf:resource="#HTIPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#SecurityPlus"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#CCDA">
    <owl:disjointWith rdf:resource="#CCNA"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CiscoAssociate"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#MCDBA">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#MicrosoftCertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#MCAD"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCT"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#MCSA"/>
  <owl:disjointWith rdf:resource="#MCDST"/>
  <owl:disjointWith rdf:resource="#MCSE"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#MCSD"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#i-NetPlus">
  <owl:disjointWith rdf:resource="#LinuxPlus"/>
  <owl:disjointWith rdf:resource="#NetworkPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#CTTPlus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#CDIAPlus"/>
  <owl:disjointWith rdf:resource="#ServerPlus"/>
  <owl:disjointWith rdf:resource="#HTIPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SecurityPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#e-BizPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#ProjectPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#APlus"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CompTIACertification"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#e-BizPlus">
  <owl:disjointWith rdf:resource="#HTIPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#ProjectPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#APlus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#ServerPlus"/>
  <owl:disjointWith rdf:resource="#i-NetPlus"/>
  <owl:disjointWith rdf:resource="#NetworkPlus"/>
  <owl:disjointWith rdf:resource="#CDIAPlus"/>
  <owl:disjointWith rdf:resource="#LinuxPlus"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#CTTPlus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#SecurityPlus"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CompTIACertification"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#StateCertification">
  <rdfs:subClassOf rdf:resource="#GovernmentCertification"/>
  <owl:disjointWith rdf:resource="#USCertification"/>
</owl:Class>
<owl:Class rdf:about="#APlus">
```

```
    <owl:disjointWith rdf:resource="#e-BizPlus"/>
    <owl:disjointWith rdf:resource="#HTIPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#NetworkPlus"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#SecurityPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#CDIAPlus"/>
    <owl:disjointWith rdf:resource="#ServerPlus"/>
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#CTTPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#i-NetPlus"/>
  </owl:Class>
  <owl:Class rdf:about="#CCIP">
    <owl:disjointWith rdf:resource="#CCNP"/>
    <rdfs:subClassOf rdf:resource="#CiscoProfessional"/>
    <owl:disjointWith rdf:resource="#CCSP"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#CCDP"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#SecurityPlus">
    <owl:disjointWith rdf:resource="#e-BizPlus"/>
    <owl:disjointWith rdf:resource="#ServerPlus"/>
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#CTTPlus"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#NetworkPlus"/>
    <owl:disjointWith rdf:resource="#i-NetPlus"/>
    <owl:disjointWith rdf:resource="#HTIPlus"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#CDIAPlus"/>
    <owl:disjointWith rdf:resource="#APlus"/>
  </owl:Class>
  <owl:Class rdf:about="#CTTPlus">
    <owl:disjointWith rdf:resource="#CDIAPlus"/>
    <owl:disjointWith rdf:resource="#ServerPlus"/>
    <owl:disjointWith rdf:resource="#e-BizPlus"/>
    <owl:disjointWith rdf:resource="#NetworkPlus"/>
    <owl:disjointWith rdf:resource="#APlus"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompTIACertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#LinuxPlus"/>
    <owl:disjointWith rdf:resource="#HTIPlus"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#ProjectPlus"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#i-NetPlus"/>
    <owl:disjointWith rdf:resource="#SecurityPlus"/>
```

```
</owl:Class>
<owl:Class rdf:about="#CiscoAssociate">
  <owl:disjointWith>
    <owl:Class rdf:about="#CiscoExpert"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CiscoCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#CiscoProfessional"/>
</owl:Class>
<owl:Class rdf:about="#CiscoExpert">
  <owl:disjointWith rdf:resource="#CiscoProfessional"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#CiscoCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#CiscoAssociate"/>
</owl:Class>
<owl:Class rdf:about="#CCDP">
  <owl:disjointWith rdf:resource="#CCNP"/>
  <rdfs:subClassOf rdf:resource="#CiscoProfessional"/>
  <owl:disjointWith rdf:resource="#CCSP"/>
  <owl:disjointWith rdf:resource="#CCIP"/>
</owl:Class>
<owl:Class rdf:ID="CCIE">
  <rdfs:subClassOf rdf:resource="#CiscoExpert"/>
</owl:Class>
<owl:Class rdf:about="#CiscoCertification">
  <owl:disjointWith>
    <owl:Class rdf:about="#CompTIACertification"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ITCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#MicrosoftCertification"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#MCT">
  <owl:disjointWith>
    <owl:Class rdf:about="#MCSD"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#MCAD"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#MicrosoftCertification"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#MCSE"/>
  <owl:disjointWith rdf:resource="#MCDBA"/>
  <owl:disjointWith rdf:resource="#MCSA"/>
  <owl:disjointWith rdf:resource="#MCDST"/>
</owl:Class>
<owl:Class rdf:about="#CompTIACertification">
  <owl:disjointWith rdf:resource="#CiscoCertification"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#MicrosoftCertification"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#ITCertification"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#MCSD">
  <owl:disjointWith rdf:resource="#MCAD"/>
  <owl:disjointWith rdf:resource="#MCSE"/>
  <owl:disjointWith rdf:resource="#MCDST"/>
```

```
    <owl:disjointWith rdf:resource="#MCT"/>
    <owl:disjointWith rdf:resource="#MCSA"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#MicrosoftCertification"/>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#MCDBA"/>
  </owl:Class>
  <owl:Class rdf:about="#ITCertification">
    <owl:disjointWith rdf:resource="#PrivacySealCertification"/>
    <rdfs:subClassOf rdf:resource="#BusinessCertification"/>
  </owl:Class>
  <owl:Class rdf:about="#MicrosoftCertification">
    <owl:disjointWith rdf:resource="#CompTIACertification"/>
    <rdfs:subClassOf rdf:resource="#ITCertification"/>
    <owl:disjointWith rdf:resource="#CiscoCertification"/>
  </owl:Class>
```

A.2.4 Provider

```
  <owl:Class rdf:about="#Provider">
    <rdfs:subClassOf rdf:resource="#Attribute"/>
  </owl:Class>
  <owl:Class rdf:ID="HealthCareProvider">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Provider"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PhysicalTherapyProvider">
    <rdfs:subClassOf rdf:resource="#Provider"/>
  </owl:Class>
```

A.2.5 USGovernmentDepartment

```
  <owl:Class rdf:about="#USGovernmentDepartment">
    <owl:disjointWith rdf:resource="#PoliticalDivisions"/>
    <owl:disjointWith rdf:resource="#PersonalInformation"/>
    <rdfs:subClassOf rdf:resource="#Attribute"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Payment"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="DepartmentOfMotorVehicles">
    <owl:disjointWith>
      <owl:Class rdf:ID="DepartmentOfState"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="DepartmentOfHeathAndHumanServices"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="USGovernmentDepartment"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#DepartmentOfHeathAndHumanServices">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#USGovernmentDepartment"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#DepartmentOfState"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#DepartmentOfMotorVehicles"/>
```

```
</owl:Class>
<owl:Class rdf:about="#DepartmentOfState">
  <owl:disjointWith rdf:resource="#DepartmentOfHeathAndHumanServices"/>
  <owl:disjointWith rdf:resource="#DepartmentOfMotorVehicles"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#USGovernmentDepartment"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#StateMedicalBoard">
  <rdfs:subClassOf rdf:resource="#DepartmentOfHeathAndHumanServices"/>
</owl:Class>
```

A.2.6 PoliticalDivisions

```
<owl:Class rdf:about="#PoliticalDivisions">
  <owl:disjointWith rdf:resource="#PersonalInformation"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Attribute"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#USGovernmentDepartment"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Payment"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="State">
  <owl:disjointWith>
    <owl:Class rdf:ID="Country"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PoliticalDivisions"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="UnitedStatesOfAmerica">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Country"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Country">
  <owl:disjointWith rdf:resource="#State"/>
  <rdfs:subClassOf rdf:resource="#PoliticalDivisions"/>
</owl:Class>
```

A.2.7 Payment

```
<owl:Class rdf:about="#Payment">
  <rdfs:subClassOf rdf:resource="#Attribute"/>
  <owl:disjointWith rdf:resource="#PoliticalDivisions"/>
  <owl:disjointWith rdf:resource="#PersonalInformation"/>
  <owl:disjointWith rdf:resource="#USGovernmentDepartment"/>
</owl:Class>
<owl:Class rdf:about="#CompanyCheck">
  <rdfs:subClassOf rdf:resource="#Check"/>
  <owl:disjointWith rdf:resource="#PayrollCheck"/>
  <owl:disjointWith rdf:resource="#PersonalCheck"/>
</owl:Class>
<owl:Class rdf:ID="PersonalCheck">
  <owl:disjointWith>
    <owl:Class rdf:ID="CompanyCheck"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Check"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="PayrollCheck"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#PayrollCheck">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Check"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:about="#CompanyCheck"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#PersonalCheck"/>
  </owl:Class>
  <owl:Class rdf:ID="VISACard">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CreditCard"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="AmericanExpressCard"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="DiscoverCard"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="MasterCardCard"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="PaymentVerification">
    <owl:disjointWith>
      <owl:Class rdf:about="#Check"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="OnlinePayment"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#CreditCard"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Payment"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="MoneyOrder"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#DiscoverCard">
    <owl:disjointWith>
      <owl:Class rdf:about="#AmericanExpressCard"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#MasterCardCard"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#VISACard"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CreditCard"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#MasterCardCard">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CreditCard"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#DiscoverCard"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#AmericanExpressCard"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#VISACard"/>
</owl:Class>
<owl:Class rdf:about="#CreditCard">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Payment"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#OnlinePayment"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MoneyOrder"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Check"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PaymentVerification"/>
</owl:Class>
<owl:Class rdf:about="#AmericanExpressCard">
  <owl:disjointWith rdf:resource="#VISACard"/>
  <owl:disjointWith rdf:resource="#MasterCardCard"/>
  <owl:disjointWith rdf:resource="#DiscoverCard"/>
  <rdfs:subClassOf rdf:resource="#CreditCard"/>
</owl:Class>
<owl:Class rdf:about="#MoneyOrder">
  <owl:disjointWith rdf:resource="#CreditCard"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#OnlinePayment"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Payment"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#Check"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#PaymentVerification"/>
</owl:Class>
<owl:Class rdf:about="#Check">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Payment"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#MoneyOrder"/>
  <owl:disjointWith rdf:resource="#PaymentVerification"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#OnlinePayment"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#CreditCard"/>
</owl:Class>
<owl:Class rdf:about="#OnlinePayment">
  <owl:disjointWith rdf:resource="#MoneyOrder"/>
  <owl:disjointWith rdf:resource="#PaymentVerification"/>
  <owl:disjointWith rdf:resource="#Check"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Payment"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#CreditCard"/>
</owl:Class>
<owl:Class rdf:ID="SalesReceipt">
  <rdfs:subClassOf rdf:resource="#PaymentVerification"/>
```

```
</owl:Class>
```

A.2.8 PersonalInformation

```
<owl:Class rdf:ID="LastName">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Name"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="DriversLicense">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ID"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#VoterRegCard"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Passport"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#IDCard"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="Address">
  <owl:disjointWith>
    <owl:Class rdf:about="#ID"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="DisabledStatus"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Name"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="PersonalInformation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#PersonalInformation">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Attribute"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:about="#USGovernmentDepartment"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#PoliticalDivisions"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Payment"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="MarriedName">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Name"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FirstName">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Name"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Name">
```

```
  <owl:disjointWith>
    <owl:Class rdf:about="#ID"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#DisabledStatus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Address"/>
  <rdfs:subClassOf rdf:resource="#PersonalInformation"/>
</owl:Class>
<owl:Class rdf:about="#ID">
  <owl:disjointWith>
    <owl:Class rdf:about="#DisabledStatus"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Address"/>
  <owl:disjointWith rdf:resource="#Name"/>
  <rdfs:subClassOf rdf:resource="#PersonalInformation"/>
</owl:Class>
<owl:Class rdf:ID="PreferredName">
  <rdfs:subClassOf rdf:resource="#Name"/>
</owl:Class>
<owl:Class rdf:about="#VoterRegCard">
  <rdfs:subClassOf rdf:resource="#ID"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#IDCard"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#DriversLicense"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Passport"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="StudentID">
  <owl:disjointWith>
    <owl:Class rdf:about="#SSCard"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#MilitaryID"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#IDCard"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Passport">
  <owl:disjointWith>
    <owl:Class rdf:about="#IDCard"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#VoterRegCard"/>
  <owl:disjointWith rdf:resource="#DriversLicense"/>
  <rdfs:subClassOf rdf:resource="#ID"/>
</owl:Class>
<owl:Class rdf:about="#IDCard">
  <owl:disjointWith rdf:resource="#Passport"/>
  <owl:disjointWith rdf:resource="#DriversLicense"/>
  <rdfs:subClassOf rdf:resource="#ID"/>
  <owl:disjointWith rdf:resource="#VoterRegCard"/>
</owl:Class>
<owl:Class rdf:ID="CurrentAddress">
  <rdfs:subClassOf rdf:resource="#Address"/>
  <owl:disjointWith>
    <owl:Class rdf:ID="PreviousAddress"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#MilitaryID">
  <owl:disjointWith>
```

```
      <owl:Class rdf:about="#SSCard"/>
    </owl:disjointWith>
    <rdfs:subClassOf rdf:resource="#IDCard"/>
    <owl:disjointWith rdf:resource="#StudentID"/>
  </owl:Class>
  <owl:Class rdf:about="#DisabledStatus">
    <owl:disjointWith rdf:resource="#Name"/>
    <owl:disjointWith rdf:resource="#ID"/>
    <rdfs:subClassOf rdf:resource="#PersonalInformation"/>
    <owl:disjointWith rdf:resource="#Address"/>
  </owl:Class>
  <owl:Class rdf:about="#SSCard">
    <owl:disjointWith rdf:resource="#StudentID"/>
    <owl:disjointWith rdf:resource="#MilitaryID"/>
    <rdfs:subClassOf rdf:resource="#IDCard"/>
  </owl:Class>
  <owl:Class rdf:about="#PreviousAddress">
    <owl:disjointWith rdf:resource="#CurrentAddress"/>
    <rdfs:subClassOf rdf:resource="#Address"/>
  </owl:Class>
  <owl:Class rdf:ID="MiddleName">
    <rdfs:subClassOf rdf:resource="#Name"/>
  </owl:Class>
```

A.2.9 Store

```
  <owl:Class rdf:ID="ElectronicsStore">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Store"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SuperMarket">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Store"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Store">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Attribute"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DepartmentStore">
    <rdfs:subClassOf rdf:resource="#Store"/>
  </owl:Class>
  <owl:Class rdf:ID="ThriftStore">
    <rdfs:subClassOf rdf:resource="#Store"/>
  </owl:Class>
  <owl:Class rdf:ID="Bookstore">
    <rdfs:subClassOf rdf:resource="#Store"/>
  </owl:Class>
  <owl:Class rdf:ID="RetailStore">
    <rdfs:subClassOf rdf:resource="#Store"/>
  </owl:Class>
```

A.3 Trust negotiation ontologies

A.3.1 BusinessDiscount

```
  <owl:Class rdf:about="#BusinessDiscount">
  <rdfs:subClassOf rdf:resource="#Negotiation"/>
</owl:Class>
<owl:Class rdf:ID="ClubMembershipDiscount">
  <owl:disjointWith>
    <owl:Class rdf:ID="AcademicDiscount"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="SeniorDiscount"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="EmployeeDiscount"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="BusinessDiscount"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="EmeritusDiscout">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="UniversityDiscount"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#AcademicDiscount">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BusinessDiscount"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#Store"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#ClubMembershipDiscount"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#SeniorDiscount"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#EmployeeDiscount"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#EmployeeDiscount">
  <owl:disjointWith>
    <owl:Class rdf:about="#SeniorDiscount"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#AcademicDiscount"/>
  <owl:disjointWith rdf:resource="#ClubMembershipDiscount"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BusinessDiscount"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#SeniorDiscount">
  <owl:disjointWith rdf:resource="#ClubMembershipDiscount"/>
  <owl:disjointWith rdf:resource="#AcademicDiscount"/>
  <owl:disjointWith rdf:resource="#EmployeeDiscount"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#BusinessDiscount"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="K12Discout">
  <rdfs:subClassOf rdf:resource="#AcademicDiscount"/>
</owl:Class>
<owl:Class rdf:ID="TeacherDiscount">
```

```
    <rdfs:subClassOf rdf:resource="#K12Discout"/>
  </owl:Class>
  <owl:Class rdf:ID="StudentDiscount">
    <rdfs:subClassOf rdf:resource="#K12Discout"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Student"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#UniversityDiscount"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="FacultyDiscount">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#UniversityDiscount"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#UniversityDiscount">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#University"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#AcademicDiscount"/>
  </owl:Class>
```

A.3.2 MedicalInformationExchange

```
  <owl:Class rdf:ID="MedicalInformationExchange">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Negotiation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ClaimStatusInquiryOrResponse">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="HIPAANegotiation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ReferralAuthorizationInquiryResponse">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#HIPAANegotiation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#HIPAANegotiation">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:about="#Provider"/>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#MedicalInformationExchange"/>
</owl:Class>
<owl:Class rdf:ID="PaymentOrRemittanceAdvice">
  <rdfs:subClassOf rdf:resource="#HIPAANegotiation"/>
</owl:Class>
<owl:Class rdf:ID="EligibilityInquiryOrResponse">
  <rdfs:subClassOf rdf:resource="#HIPAANegotiation"/>
</owl:Class>
  <owl:Class rdf:ID="MedicalRecordsRequest">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#MedicalPractitioner"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom>
        <owl:Class rdf:ID="StateMedicalBoard"/>
      </owl:someValuesFrom>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#HIPAANegotiation"/>
</owl:Class>
<owl:Class rdf:ID="ClaimsOrEncounterInformation">
  <rdfs:subClassOf rdf:resource="#HIPAANegotiation"/>
</owl:Class>
```

A.3.3 DMV

```
<owl:Class rdf:about="#DMV">
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Department of motor vehicles</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="#Certification"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:someValuesFrom rdf:resource="#State"/>
      <owl:onProperty>
        <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#Negotiation"/>
</owl:Class>
<owl:Class rdf:ID="RegistrationRenewal">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Registration"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Licensing">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="VoterRegCard"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="hasTypicalAttribute"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="Passport"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="MilitaryID"/>
        </owl:someValuesFrom>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="IDCard"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom>
          <owl:Class rdf:ID="SSCard"/>
        </owl:someValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DMV"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DriversLicenseFirstTime">
    <rdfs:subClassOf rdf:resource="#Licensing"/>
  </owl:Class>
  <owl:Class rdf:ID="Accident">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Reports"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="DrivingRecord"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="CarHistory"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="CarInsuranceQuotes"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:about="#DrivingRecord">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Reports"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Accident"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#CarHistory"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#CarInsuranceQuotes"/>
  </owl:disjointWith>
</owl:Class>
<owl:Class rdf:ID="DisabledPlates">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="LicensePlateNeg"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChangeOfAddress">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DMV"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Financial">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DMV"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Reports">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DMV"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Registration">
  <owl:disjointWith>
    <owl:Class rdf:ID="TransferOfTitle"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#LicensePlateNeg"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="TitleRegistration"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#TransferOfTitle">
  <owl:disjointWith rdf:resource="#Registration"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#LicensePlateNeg"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#TitleRegistration"/>
  </rdfs:subClassOf>
</owl:Class>
  <owl:Class rdf:about="#LicensePlateNeg">
    <owl:disjointWith rdf:resource="#Registration"/>
    <owl:disjointWith rdf:resource="#TransferOfTitle"/>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#TitleRegistration"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DriversLicenseRenewal">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:someValuesFrom rdf:resource="#CreditCard"/>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:about="#hasTypicalAttribute"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource="#DriversLicense"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#Licensing"/>
  </owl:Class>
  <owl:Class rdf:about="#CarInsuranceQuotes">
    <owl:disjointWith rdf:resource="#DrivingRecord"/>
    <owl:disjointWith rdf:resource="#Accident"/>
    <rdfs:subClassOf rdf:resource="#Reports"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#CarHistory"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#TitleRegistration">
    <rdfs:subClassOf rdf:resource="#DMV"/>
  </owl:Class>
  <owl:Class rdf:about="#CarHistory">
    <owl:disjointWith rdf:resource="#CarInsuranceQuotes"/>
    <owl:disjointWith rdf:resource="#Accident"/>
    <owl:disjointWith rdf:resource="#DrivingRecord"/>
    <rdfs:subClassOf rdf:resource="#Reports"/>
  </owl:Class>

</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 2.0 beta, Build 242)
http://protege.stanford.edu -->
```

Appendix B — TPL Policies

This appendix contains a description of the modifications made to each of the three scenarios used in the analysis of the credential relevancy framework prototype, together with their TPL policies. For additional explanation of TPL policy syntax and semantics see Herzberg et al. [14].

B.1 DMV negotiation server policy

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- DRIVERS LICENSE Scenario: Department of Public Safety Policy (Server) -->

<POLICY>
  <GROUP NAME="self" />

  <GROUP NAME="USCountry">
    <RULE>
      <INCLUSION ID="us_cert" TYPE="Country" FROM="self">
        <FUNCTION>
          <EQ><FIELD ID="us_cert" NAME="CountryCode" /><CONST>US</CONST></EQ>
        </FUNCTION>
      </INCLUSION>
    </RULE>
  </GROUP>

  <GROUP NAME="ForeignCountry">
    <RULE>
      <INCLUSION ID="foreign_cert" TYPE="Country" FROM="self">
        <FUNCTION>
          <NE><FIELD ID="foreign_cert" NAME="CountryCode" /><CONST>US</CONST></NE>
        </FUNCTION>
      </INCLUSION>
    </RULE>
  </GROUP>

  <GROUP NAME="InState">
    <RULE>
      <INCLUSION ID="instate_cert" TYPE="State" FROM="USCountry">
        <FUNCTION>
          <EQ><FIELD ID="instate_cert" NAME="StateCode" /><CONST>IL</CONST></EQ>
        </FUNCTION>
      </INCLUSION>
    </RULE>
  </GROUP>

  <GROUP NAME="OutofState">
    <RULE>
      <INCLUSION ID="outstate_cert" TYPE="State" FROM="USCountry">
        <FUNCTION>
          <NE><FIELD ID="outstate_cert" NAME="StateCode" /><CONST>IL</CONST></NE>
        </FUNCTION>
      </INCLUSION>
    </RULE>
```

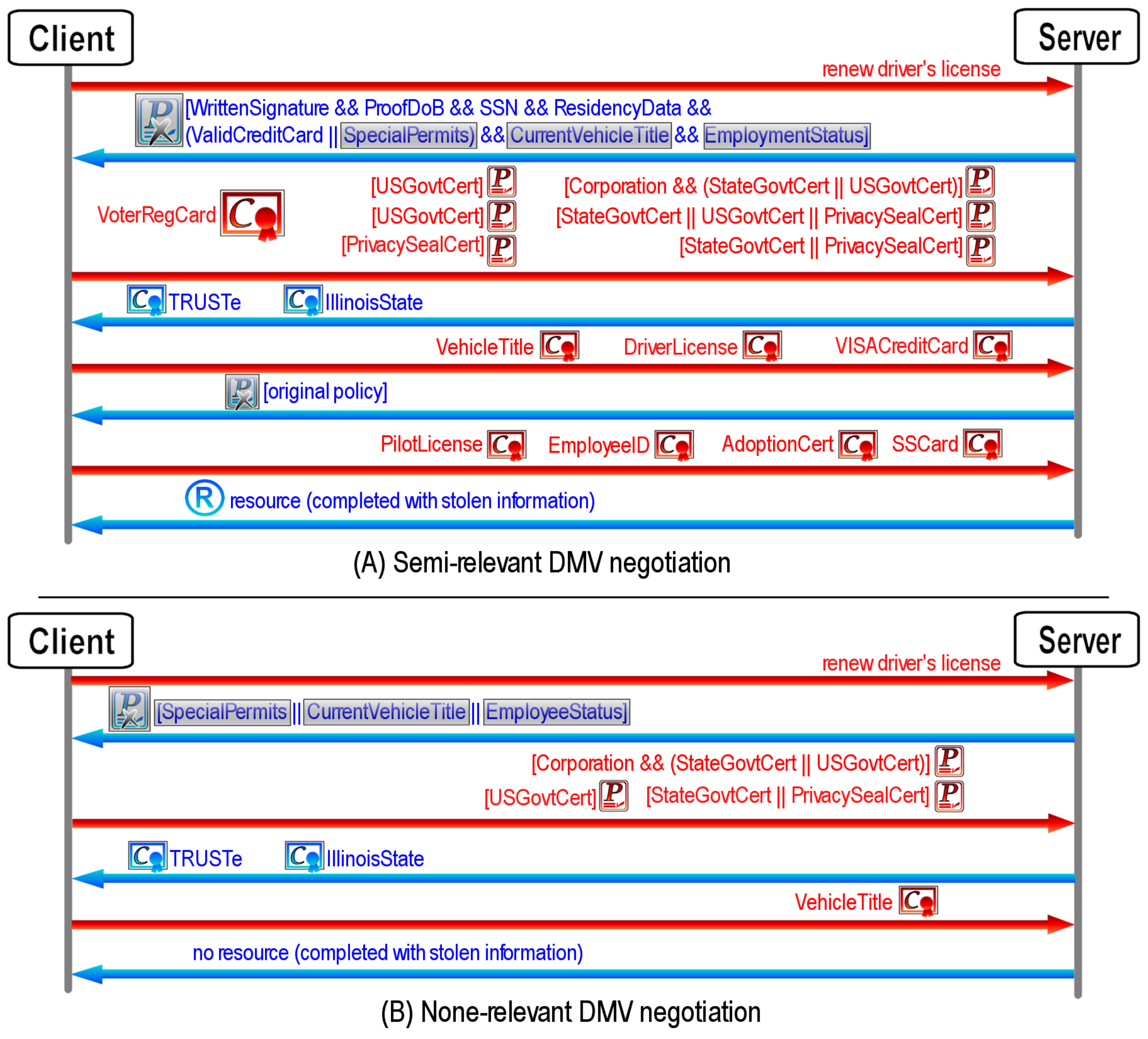

Client
Server
renew driver's license
[WrittenSignature && ProofDoB && SSN && ResidencyData &&
(ValidCreditCard || SpecialPermits) && CurrentVehicleTitle && EmploymentStatus]
VoterRegCard
[USGovtCert]
[USGovtCert]
[PrivacySealCert]
[Corporation && (StateGovtCert || USGovtCert)]
[StateGovtCert || USGovtCert || PrivacySealCert]
[StateGovtCert || PrivacySealCert]
TRUSTe
IllinoisState
VehicleTitle
DriverLicense
VISACreditCard
[original policy]
PilotLicense
EmployeeID
AdoptionCert
SSCard
resource (completed with stolen information)
(A) Semi-relevant DMV negotiation
Client
Server
renew driver's license
[SpecialPermits || CurrentVehicleTitle || EmployeeStatus]
[Corporation && (StateGovtCert || USGovtCert)]
[USGovtCert]
[StateGovtCert || PrivacySealCert]
TRUSTe
IllinoisState
VehicleTitle
no resource (completed with stolen information)
(B) None-relevant DMV negotiation

Figure B.1: The semi-relevant DMV negotiation extends the baseline DMV negotiation by adding three new policy groups *SpecialPermits*, *CurrentVehicleTitle*, and *Employment Status*. The existing groups *ResidencyPersonalData* and *ProofDateOf-Birth* were also extended. The semi-relevant negotiation employs *backtracking* [35] after the third round to recompute a satisfying set of credentials. The server in this scenario is unable to satisfy the client's *USGovtCertification* and *Corporation* policy groups (i.e., the server lacks the appropriate credentials), yet it can still usurp the client's irrelevant credentials. Both illustrated negotiations omit redundant policy requests during each round.

```
</GROUP>

<GROUP NAME="CCCompany">
  <RULE><INCLUSION ID="cccompany_cert" TYPE="CCCompany" FROM="self" /></RULE>
</GROUP>

<GROUP NAME="WrittenSignature">
  <RULE><INCLUSION ID="cc_cert" TYPE="CreditCard" FROM="CCCompany"/></RULE>
  <RULE><INCLUSION ID="dl_cert" TYPE="DriversLicense" FROM="InState"/></RULE>
  <RULE><INCLUSION ID="milID_cert" TYPE="MilitaryID" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="OoSdl_cert" TYPE="DriversLicense" FROM="OutofState"/></RULE>
  <RULE><INCLUSION ID="OoSid_cert" TYPE="IDCard" FROM="OutofState"/></RULE>
  <RULE><INCLUSION ID="passport_cert" TYPE="Passport" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="sscard_cert" TYPE="SSCard" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="ProofDateOfBirth">
  <RULE><INCLUSION ID="dl_cert2" TYPE="DriversLicense" FROM="InState"/></RULE>
  <RULE><INCLUSION ID="id_cert2" TYPE="IDCard" FROM="InState"/></RULE>
  <RULE><INCLUSION ID="milID_cert2" TYPE="MilitaryID" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="passport_cert2" TYPE="Passport" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="birth_cert" TYPE="BirthCertificate" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="adoption_cert" TYPE="AdoptionCertificate" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="SocialSecurityNumber">
  <RULE><INCLUSION ID="milID_cert3" TYPE="MilitaryID" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="sscard_cert2" TYPE="SSCard" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="ResidencyPersonalData">
  <RULE><INCLUSION ID="voterreg_cert" TYPE="VoterRegCard" FROM="USCountry"/></RULE>
  <RULE><INCLUSION ID="dl_cert3" TYPE="DriversLicense" FROM="InState"/></RULE>
  <RULE><INCLUSION ID="id_cert3" TYPE="IDCard" FROM="InState"/></RULE>
  <RULE><INCLUSION ID="empid" TYPE="EmployeeID" FROM="IBM"/></RULE>
</GROUP>

<GROUP NAME="ValidCreditCardHolder">
  <RULE><INCLUSION ID="cc_cert2" TYPE="CreditCard" FROM="CCCompany"/></RULE>
</GROUP>

<GROUP NAME="SpecialPermits">
  <RULE><INCLUSION ID="spt2" TYPE="WeaponPermit" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="LocalDMV">
  <RULE><INCLUSION ID="a_local_dmv" TYPE="StateDMV" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="TrustedFlightSchool">
  <RULE><INCLUSION ID="flight_s" TYPE="TrustedFlightSchool" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="CurrentVehicleTitle">
  <RULE><INCLUSION ID="cvtx" TYPE="VehicleTitle" FROM="LocalDMV"/></RULE>
  <RULE><INCLUSION ID="pilotl1" TYPE="PilotLicense" FROM="TrustedFlightSchool"/></RULE>
</GROUP>

<GROUP NAME="IBM">
  <RULE><INCLUSION ID="bus_id_1" TYPE="RegisteredBusiness" FROM="USCountry"/></RULE>
</GROUP>

<GROUP NAME="EmploymentStatus">
```

```
    <RULE><INCLUSION ID="empid1" TYPE="Disabled" FROM="USCountry"/></RULE>
    <RULE><INCLUSION ID="empid2" TYPE="EmployeeID" FROM="IBM"/></RULE>
  </GROUP>
</POLICY>
```

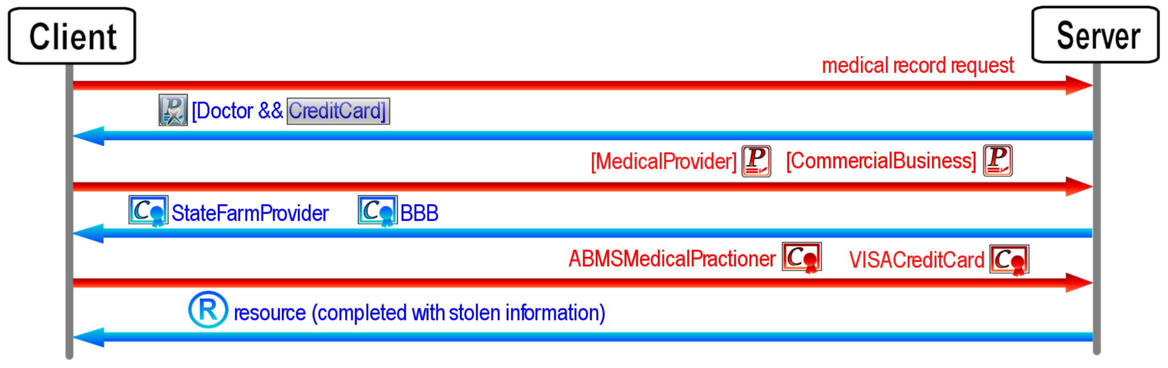
B.2 DMV negotiation client policy

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- DRIVERS LICENSE Scenario: Andy's Policy (Client) -->

<POLICY>
  <GROUP NAME="self" />

  <GROUP NAME="USCountry">
    <RULE>
      <INCLUSION ID="us_cert" TYPE="Country" FROM="self">
        <FUNCTION>
          <EQ><FIELD ID="us_cert" NAME="CountryCode" /><CONST>US</CONST></EQ>
        </FUNCTION>
      </INCLUSION>
    </RULE>
  </GROUP>

  <GROUP NAME="ForeignCountry">
    <RULE>
      <INCLUSION ID="foreign_cert" TYPE="Country" FROM="self">
        <FUNCTION>
          <NE><FIELD ID="foreign_cert" NAME="CountryCode" /><CONST>US</CONST></NE>
        </FUNCTION>
      </INCLUSION>
    </RULE>
  </GROUP>

  <GROUP NAME="State">
    <RULE><INCLUSION ID="instate_cert" TYPE="State" FROM="USCountry" /></RULE>
  </GROUP>

  <GROUP NAME="PrivacySealProgram">
    <RULE><INCLUSION ID="prog_cert" TYPE="Program" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="PrivacySealCertified">
    <RULE><INCLUSION ID="privacy_cert" TYPE="Certification" FROM="PrivacySealProgram" /></RULE>
  </GROUP>

  <GROUP NAME="USGovtCertified">
    <RULE><INCLUSION ID="usgovt_cert" TYPE="Certification" FROM="USCountry" /></RULE>
  </GROUP>

  <GROUP NAME="StateGovtCertified">
    <RULE><INCLUSION ID="stategovt_cert" TYPE="Certification" FROM="State" /></RULE>
  </GROUP>

  <GROUP NAME="ForeignGovtCertified">
    <RULE><INCLUSION ID="foreigngovt_cert" TYPE="Certification" FROM="ForeignCountry" /></RULE>
  </GROUP>

  <GROUP NAME="BBB">
    <RULE><INCLUSION ID="bbb111" TYPE="CertificationAuthority" FROM="self" /></RULE>
  </GROUP>

  <!-- This group cannot be satisfied on the Server :) -->
  <GROUP NAME="Corporation">
```

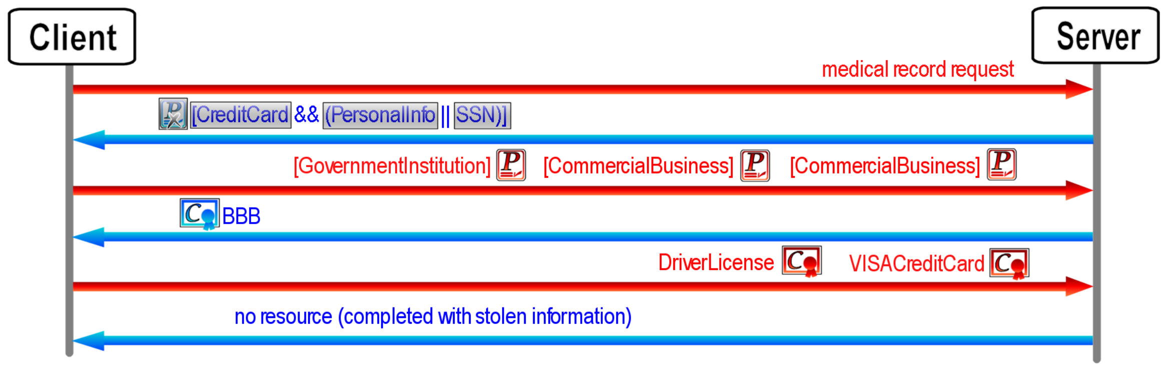
```
    <RULE><INCLUSION ID="corporation1" TYPE="Incorporated" FROM="BBB" /></RULE>
  </GROUP>
</POLICY>
```

B.3 Medical records negotiation server policy

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- MEDICAL RECORDS Scenario: State Farm Policy (Server) -->

<POLICY>
  <GROUP NAME="self" />

  <GROUP NAME="State">
    <RULE><INCLUSION ID="Board_cert0" TYPE="State" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="StateMedicalBoard">
    <RULE><INCLUSION ID="Board_cert1" TYPE="StateMedicalBoard" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="CertificationBoard">
    <RULE><INCLUSION ID="Board_cert2" TYPE="CertificationBoard" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="Doctor">
    <RULE><INCLUSION ID="Doctor_cert1" TYPE="MedicalPractitioner" FROM="StateMedicalBoard" /></RULE>
    <RULE><INCLUSION ID="Doctor_cert2" TYPE="MedicalPractitioner" FROM="CertificationBoard" /></RULE>
  </GROUP>

  <GROUP NAME="CCCompany">
   <RULE><INCLUSION ID="ccc" TYPE="TrustedCreditCardCompany" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="CreditCard">
    <RULE><INCLUSION ID="cc" TYPE="MasterCard" FROM="CCCompany" /></RULE>
    <!-- OR -->
    <RULE><INCLUSION ID="cc2" TYPE="Discover" FROM="CCCompany" /></RULE>
    <RULE><INCLUSION ID="cc3" TYPE="AmericanExpress" FROM="CCCompany" /></RULE>
    <RULE><INCLUSION ID="cc4" TYPE="VISA" FROM="CCCompany" /></RULE>
  </GROUP>

  <GROUP NAME="USGov">
    <RULE><INCLUSION ID="usgov" TYPE="TrustedGovernment" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="LocalDMV">
    <RULE><INCLUSION ID="ldmv" TYPE="GovernmentBranch" FROM="USGov" /></RULE>
  </GROUP>

  <GROUP NAME="PersonalInfo">
    <RULE><INCLUSION ID="dl" TYPE="DriverLicense" FROM="LocalDMV" /></RULE>
    <!-- OR -->
    <RULE><INCLUSION ID="pass" TYPE="Passport" FROM="USGov" /></RULE>
  </GROUP>

  <GROUP NAME="SocialSecurityAdministration">
    <RULE><INCLUSION ID="ssa" TYPE="GovernmentBranch" FROM="USGov" /></RULE>
  </GROUP>

  <GROUP NAME="SocialSecurityNumber">
    <RULE><INCLUSION ID="ssn1" TYPE="SSCard" FROM="SocialSecurityAdministration" /></RULE>
  </GROUP>
</POLICY>
```


Client
Server
medical record request
[Doctor && CreditCard]
[MedicalProvider] P [CommercialBusiness] P
C StateFarmProvider C BBB
ABMSMedicalPractioner C VISACreditCard C
R resource (completed with stolen information)

(A) Semi-relevant medical records negotiation


Client
Server
medical record request
[CreditCard && (PersonalInfo || SSN)]
[GovernmentInstitution] P [CommercialBusiness] P [CommercialBusiness] P
C BBB
DriverLicense C VISACreditCard C
no resource (completed with stolen information)

(B) None-relevant medical records negotiation

Figure B.2: The semi-relevant medical records negotiation (A) extends the baseline negotiation by adding a *CreditCard* policy group. The none-relevant negotiation (B) adds the policy groups *PersonalInfo* and *SocialSecurityNumber*. The semi-relevant negotiation illustrates how the server is able to retrieve the client's credit card during the negotiation (because the server also possesses a BBB credential). The client is unaware that medical records are not the subject of the server's policy in the none-relevant negotiation. Because the server can partially satisfy the client's policies with a BBB credential (the server does not have a credential that can satisfy the *GovernmentInstitution* policy group), the DriverLicense and VISA card are released.

B.4 Medical records negotiation client policy

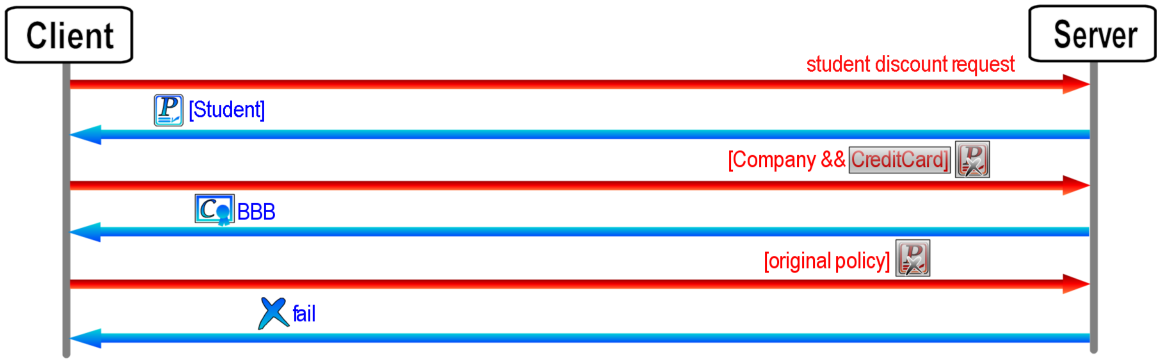
```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- MEDICAL RECORDS Scenario: Bob's Policy (Client) -->

<POLICY>
  <GROUP NAME="self" />

  <GROUP NAME="State">
    <RULE><INCLUSION ID="State_cert1" TYPE="State" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="Provider">
    <RULE><INCLUSION ID="Provider_cert1" TYPE="Provider" FROM="State" /></RULE>
  </GROUP>

  <GROUP NAME="ValidBizLicensers">
    <RULE><INCLUSION ID="vbl1" TYPE="TrustedBizCertifier" FROM="self" /></RULE>
  </GROUP>


  <GROUP NAME="CommercialBusiness">
    <RULE><INCLUSION ID="combus" TYPE="BBB" FROM="ValidBizLicensers" /></RULE>
  </GROUP>

  <GROUP NAME="GovernmentSelfSigned">
    <RULE><INCLUSION ID="gov_self_signed" TYPE="US" FROM="self" /></RULE>
  </GROUP>

  <GROUP NAME="GovernmentInstitution">
    <RULE><INCLUSION ID="gov_ent" TYPE="USGovernment" FROM="GovernmentSelfSigned" /></RULE>
  </GROUP>
</POLICY>
```

B.5 Bookstore negotiation server TPL policy

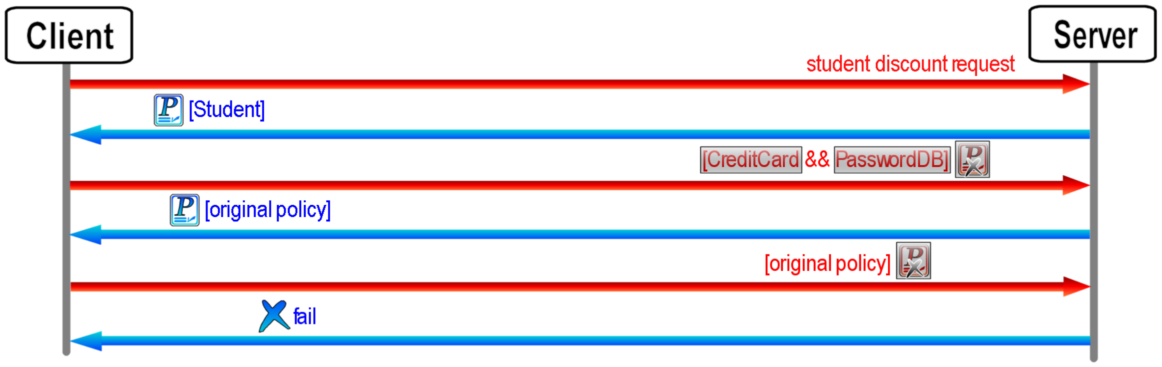
```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- BOOKSTORE Scenario: Store Policy (Server) -->

<POLICY>
  <GROUP NAME="self" />

  <GROUP NAME="AccreditingBody">
    <RULE><INCLUSION FROM="self" ID="accreditingbodycert" TYPE="AccreditingBody" /></RULE>
  </GROUP>

  <GROUP NAME="University">
    <RULE><INCLUSION FROM="AccreditingBody" ID="universitycert" TYPE="University" /></RULE>
  </GROUP>

  <GROUP NAME="Student">
    <RULE><INCLUSION FROM="University" ID="studentcert" TYPE="Student" /></RULE>
  </GROUP>
</POLICY>
```

B.6 Bookstore negotiation client TPL policy

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE POLICY SYSTEM "Policy.dtd">
<!-- BOOKSTORE Scenario: Student Policy (Client) -->

<POLICY>
  <GROUP NAME="self"></GROUP>
```



(A) Semi-relevant bookstore negotiation



(B) None-relevant bookstore negotiation

Figure B.3: The bookstore semi-relevant negotiation differs from the previous two because the client sends the malicious policy. The malicious policy in the semi-relevant negotiation includes the policy group *CreditCard* and is further extended in the none-relevant negotiation with the group *PasswordDB*. In both negotiations, the server does not have the credentials necessary to satisfy either the *CreditCard* or the *PasswordDB* policy groups (wishful thinking on behalf of the client). Both negotiations terminate with failure.

```
  <GROUP NAME="PasswordDB">
    <RULE><INCLUSION FROM="self" ID="password_database" TYPE="DBFile" /></RULE>
  </GROUP>

  <GROUP NAME="CCCompany">
    <RULE><INCLUSION FROM="self" ID="credit_company" TYPE="CCCompany" /></RULE>
  </GROUP>

  <GROUP NAME="CreditCards">
    <RULE><INCLUSION FROM="CCCompany" ID="credit" TYPE="CreditCard" /></RULE>
  </GROUP>
</POLICY>
```