Università degli Studi dell'Insubria
Dipartimento di Informatica e Comunicazione

Dottorato di Ricerca in Informatica
XXIII Ciclo – 2007-2010
Ph.D. Thesis

Advanced Languages and Techniques for Trust Negotiation

Stefano Braghin

Advisor:
Dr. Alberto Trombetta

External Advisor:
Prof. Anna Cinzia Squicciarini

Supervisor of the Doctoral program:
Prof. Gaetano Aurelio Lanzarone

In memory of my grandparents.

Abstract

The Web is quickly shifting from a document browsing and delivery system to a hugely complex ecosystem of interconnected online applications. A relevant portion of these applications dramatically increase the number of users required to dynamically authenticate themselves and to, on the other hand, to identify the service they want to use. In order to manage interactions among such users/services is required a flexible but powerful mechanism. Trust management, and in particular trust negotiation techniques, is a reasonable solution.

In this work we present the formalization of the well known trust negotiation framework Trust-$\mathcal{X}$, of a rule-based policy definition language, called $\mathcal{X}$-RNL.

Moreover, we present the extension of both the framework and of the language to provide advanced trust negotiation architectures, namely negotiations among groups.

We also provide protocols to adapt trust negotiations to mobile environments, specifically, we present protocols allowing a trust negotiation to be executed among several, distinct, sessions while still preserving its security properties. Such protocols have also been extended to provides the capability to migrate a ongoing trust negotiation among a set of known, reliable, subjects.

Finally, we present the application of the previously introduced trust negotiation techniques into real world scenarios: online social networks, critical infrastructures and cognitive radio networks.

Acknowledgements

First and foremost, I would like to sincerely thank my advisor Alberto Trombetta for constant support and collaboration. During the last years, as thesis advisor first and as Ph.D. advisor later, he helped me with precious recommendations and suggestions.

I am also really grateful to prof. Anna Cinzia Squicciarini and dr. Igor Nai Fovino for their help in teaching me by examples the meaning of "hard work".

I wish to also thank prof. Gaetano Aurelio Lanzarone, supervisor of the doctoral program, for the support he gave to all of us as Ph.D. students.

Moreover, I wish to thank all my Lab colleagues but in particular Michele Chinosi, Loris Bozzato, Paolo Brivio, Lorenzo Bossi, Mauro Santabarbara, Mauro Ferrari and Pietro Colombo, essential for all the advices and the coffee breaks we shared.

Finally, I thank my family, my girlfriend Cristina and my friends for encouraging, supporting and bearing me all the time.

Thank you all,
Stefano

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The Web is quickly shifting from a document browsing and delivery system to a hugely complex ecosystem of interconnected online applications. A relevant portion of these applications dramatically increase the number of users required to dynamically authenticate themselves and, on the other hand, to identify the service they want to use.

Traditional solutions to the authorization problem for distributed systems have made implicit use of a *closed world* model, in which users and resources have a priori knowledge of one another. Therefore, in such model, if a user Alice wants to access some resource, she must first prove her identity to the resource owner Bob, by using a username/password pair or more sophisticate authentication techniques such as X.509 identity certificate [51] or Kerberos ticket [74]. If Alice will be able to authenticate herself as entitled to the right to access the resource, then she will be granted by Bob (see Figure 1.1).

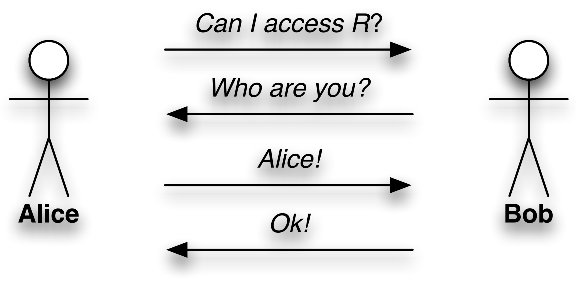


Figure 1.1: An example of identity-based authentication protocol

The evolution of the distributed environment to open environment does not allow the use of the closed world model. Traditional identity-based access control systems are not able to establish in a proper way a certain level of trust among service users and service providers involving unknown users. Therefore, in order to provide a proper authorization infrastructure to open environments, like the Web or some large application domains, researchers have begun to investigate novel attribute-based access control (ABAC for short) systems based on techniques such as *trust management*.

In the recent years, *trust management* has become an active and relevant research area [87, 126, 60, 114, 15, 17], developing – among other things – efficient techniques aiming at the automated negotiation of a proper level of trust among (initially) mutually untrusting parties. Usually, this is needed since a party may request access to a resource owned by another party and there is no shared policy regulating resource access. Nevertheless, parties are assumed to possibly posses local access policies.

Aim of trust management – and of *trust negotiation* [16, 127, 92, 48] sub-field in particular – is to build such shared policy starting from parties' local policies. As a result, a trust negotiation yields a yes/no answer to an initial request of resource's access, given the parties' policies.

More precisely, trust negotiation [116] is an authorization approach in which resources are protected by attribute-based access policies, rather than explicit access control lists. Furthermore, each user maintains some collection of digital credentials, like X.509 certificates [51], attesting to various attributes of the user or the context of his or her surrounding environment. These credentials are generally issued by a trusted third-party, such as professional organizations or employers, like public key infrastructure (PKI) [56, 2] and *webs of trust* [41].

A trust negotiation process starts up when a peer is requesting access to a resource owned by another peer. As said before, such resource is protected by policies that will check values of attributes (or credentials) of other resources . The resources that satisfy the first asked resource's access control policy may be protected by access control policies themselves. Thus, rounds in which peer mutually ask for resources ensues, yielding the so-called trust negotiation.

This is different from the behavior of most of the other distributed proof systems, such as distributed proof construction. *Distributed proof construction* [59] is another approach to trust management parallel to trust negotiation system. Distributed proof construction is, therefore, another ABAC system in which each principal maintains a local *knowledge base* that encapsulates its own view of the system. One portion of a principal's knowledge base is the set of *facts* that are currently known by the principal, which may include both *local* facts as well as digitally-signed *quoted* facts asserted by other principals. This set of facts is known as the principal's *extensional* knowledge base. A principal's knowledge base also contains a set of *derivation rules* that can be used to derive new facts from existing local facts and facts stored in the knowledge bases of other principals. The set of facts that can be derived using these derivation rules is known as the principal's *intensional* knowledge base. The decentralized nature of the distributed proof process makes it particularly well-suited for use in open systems, as the incomplete (and often complementary) views of many principals can be used to make decisions involving knowledge spread across multiple administrative domains.

Distributed proof construction differs from trust negotiation where, on the other hand, little or no protection is offered to the informations used in the authentication process. This flexibility to define attribute-based release policies for individual credentials gives users fine-grained, yet flexible, control over the disclosure of their potentially-sensitive attribute data.

In this thesis we will present some relevant extensions to the well-known trust negotiation framework Trust-$\mathcal{X}$ (described at length in Chapter 2 and Chapter 3) in order to improve its flexibility and its expressiveness and to increase the provided features in general. We mainly focused our work on two issues:

1. to extend trust negotiations between groups of peers instead of only between two peers

2. to make trust negotiations more suitable for mobile environments.

As a side effect we have implemented a full refactoring of the Trust-$\mathcal{X}$ prototype.

We designed our protocols keeping in mind the peer-to-peer (P2P) environments. A relevant feature of P2P systems is the possibility to dynamically create groups. Such groups are typically created to share resources and/or to carry on joint tasks [125, 80].

For a peer group to successfully pursue its goals, it is crucial that only cooperative and contributing peers be allowed to join the group. Moreover, if the shared resources and/or common tasks are sensitive, the access of peers joining the group must be controlled. Also, a peer before joining a group may want to verify conditions concerning the peers of this group, in order to avoid joining a malicious peer group. Therefore, the *group joining process* is a crucial phase in the lifetime of peer groups, because during such phase:

- peer group members can verify that the joining peers own resources of interest for the group and

- joining peers can evaluate the resources and services offered by the members of the group.

Current P2P systems adopt peer authentication techniques based on the traditional identity-based mechanisms. Such mechanisms are inadequate for the peer joining process because the joining criteria are not related with the actual features of the peers, such as the domain they belong to, or the resources and services they make available. Therefore, by authenticating the peer there is no actual guarantee regarding the peer's trustworthiness and its ability to contribute to the peer group. Rather, authentication mechanisms that are based on the verification of properties related to the peers are desirable.

The adoption of trust negotiations into the peer group joining process addresses the requirement of a simple yet effective authentication mechanism that is property-based. However, the application of trust negotiation techniques into the P2P joining process is challenging. First, the policy negotiation language should allow a peer group to state the conditions under which a new peer can join the group and the conditions that a joining peer wants to be satisfied to join the group. Typical trust negotiation languages do not support these features [92]; in the context of peer groups, conditions should be specified against the resources that the joining peer and the peer group members make available.

Second, the verification of such policies during the joining process might involve a subset of peers in the peer group, while current negotiation processes do not consider group policy verification. Therefore, the negotiation protocol should be extended so that the negotiation is carried out according to the disclosure policies of all the peers in the peer group and the policies verification is a collaborative process between the peer group members.

Moreover, the policies based on which a peer has joined a group must be satisfied not only when the membership has been granted to such peer but for all the duration of the membership. The policies satisfaction might not be guaranteed for different reasons:

a) a peer is malicious and does not want to make available the resources it has promised during the joining process;

b) the peer groups are dynamic and thus the composition of the peer group may change.

To address such requirements we created a join process for P2P systems based on trust negotiations. The trust negotiation is driven by policies stating conditions on peers' *verifiable* properties. Such properties can be the peers' inherent attributes or properties of the resources offered by the peers.

Beside the dynamic creation of groups, another characteristic of P2P environment is that the services and resources provided by P2P network are ubiquitously available. Such availability further improves the usage of mobile devices to access the available resources. Therefore, such widespread use of mobile devices requires real applications to provide flexible approaches to trust negotiations. Consider, for example, mobile clients negotiating accesses to services hosted on servers' clusters: negotiations may interrupt due to communication channel fault or may be voluntarily suspended, to be resumed under more favorable conditions.

Mobile devices need to be able to seamlessly migrate from different physical servers belonging to the same service provider. Also, negotiations may last a considerable time span and the involved parties may not be able to support long negotiations. However, none of the existing trust negotiation systems currently support any form of suspension or interruption, nor do they allow the negotiators to be replaced (or delegated) while the negotiation is ongoing.

Interruptions in ongoing trust negotiations can be the result of external, unforeseeable events (e.g. peers' crashes, faulty transmission channels) or decisions by the involved parties. A peer may not be able to advance the negotiation for temporary lack of resources. Or the peer may not have readily available the credentials required by the counterpart, although eligible to them. For example, users may not have the capabilities or rights of storing certificates such as birth certificates, marriage certificates and soon and so forth, although entitled to them.

Peers may also employ one-time credentials to conduct negotiations. Temporary and one-time credentials allow a peer to disclose sensitive information while at the same limiting the possibility for an attacker to steal identity related information. However, once such a credential is disclosed, it cannot be re-used. Hence, completing a negotiation in which such type of credential is used becomes crucial.

Interrupted negotiations however represent not only undesired events, but also vulnerabilities that could facilitate attackers' eavesdropping and other malicious behavior. Additionally, abrupt interruptions of trust negotiations may imply the disclosure of possible sensitive data which did not lead to any successful trust establishment despite the potential of being so. Starting a new negotiation process may therefore not always be feasible nor desirable. Unfortunately, there are no approaches addressing such an issues.

Hence, it is crucial to extend trust negotiation protocols along several directions. First , the protocols must be able to adapt to context changes and be dependable.

A *long lasting* trust negotiation [106] should successfully withstand suspensions and interruptions. Also, given the ubiquitous nature of on-line peer-to-peer systems, and the increasing number of moving objects involved in online transactions, negotiators must be allowed to switch roles while the negotiation is ongoing, so to guarantee dependability, when contextual conditions, such as availability of resources and peers, change.

Resuming the negotiations is often desirable when the involved entities are peers temporary unable to complete the negotiations, but not willing or not interested in restarting from the beginning, for example due to the usage of one-time credentials or credentials which the peer is only willing to disclose once, as previously introduced. Also, some physical constraints may apply, related to network bandwidth or availability of data. Additionally, long lasting negotiations may represent a need in case of failure, due to peers' temporary crash.

The rest of the thesis is organized as follows. We begin discussing related work in Chapter 2. Subsequently, in Chapter 3, we formalize our policy definition language and the base trust negotiation protocol provides by Trust-$\mathcal{X}$. After that, in Chapter 4 we present our extension to both the language and the protocols provided by Trust-$\mathcal{X}$ in order to support trust negotiation among groups. Chapter 5 presents at length the extension regarding the mobile multi-session trust negotiation. Chapter 6, Chapter 7 and Chapter 8 present applications of the techniques presented in the previous chapter to real-world scenarios. Finally, we present our conclusions and discuss promising directions for future work in Chapter 9.

Chapter 2

State of the art

Trust negotiation, especially for web-based application has been recognized as an interesting and challenging area and it has been an active field of research in the last years. In the following will be presented an overview of the main work developed.

Trust negotiation was originally introduced by Winsborough et al. [115], who presented two negotiation strategies for conducting on-line transactions between strangers.

Generally, the goal of trust negotiation protocols is to enable two stranger entities to establish mutual on-line trust. Such protocols however have several limitations as compared to our approach, in that they do not support negotiations between a party and a group of parties and do not deal with verification of resource availability.

Several implementations of different trust negotiation protocols exist.

Up to now, the best-known trust management system is KeyNote [16]. Keynote was designed to work for a variety of large and small scale Internet-based applications. It provides a single, language for both local policies and credentials. KeyNote credentials, called 'assertions', contain predicates describing delegations in terms of actions that are relevant to a given application. As a result, KeyNote policies do not handle credentials as a mean to establish trust. Therefore, it has several shortcomings with respect to trust negotiations.

A trust negotiation approach specifically developed for P2P systems is PeerTrust [124]. PeerTrust is a reputation-based framework, which includes a decentralized implementation over a structured P2P network. PeerTrust has been extended [124, 73] to support trust negotiation for peers on the Semantic Web.

PeerTrust does not allow one to express policies against group properties.
PeerTrust has been further extended by Nejdl et al. [73] to support trust negotiation for peers on the Semantic Web. The goal of Nejdl's approach is to automatically create a certified proof that a party is entitled to access a particular resource on the Semantic Web. Their approach therefore does not deal with allowing a party to join a peer group, which is the goal of ours. The parties' credentials in the approach by Nejdl et al. are personal to the users managing the peers and are not related to the

actual peers. Also, they consider negotiations as a two party process that does not relate nor involve peer groups.

The prototype trust negotiation system for the TrustBuilder Project is being designed and developed at the Internet Security Research Lab at Brigham Young University, under Prof. Kent E. Seamons [9]. The implementations utilize the IBM Trust Establishment system to create X.509v3 certificates. The Trust Establishment system supports XML role-based access control policies that TrustBuilder uses to govern access to sensitive credentials, policies, and services. The Trust Establishment runtime system includes a compliance checker that TrustBuilder uses to verify whether a set of certificates satisfies an access control policy and to determine which credentials satisfy a policy.

The Trust Establishment Project at Haifa Research Lab has developed a tool (TE) for enabling trust relationships between strangers based on public key certificates. The TE system includes an intelligent certificate collector that automatically collects missing certificates from certificate repositories, allowing the use of standard browsers that can only pass one certificate to the server. The TE system does not have the notion of sensitive policies nor it supports sensitive credentials.

The TrustBuilder prototype has been extended into TrustBuilder2 [62]. TrustBuilder2 leverages a plug-in based architecture, extensible data type hierarchy, and flexible communication protocol to provide a framework within which numerous trust negotiation protocols and system configurations can be quantitatively analyzed.

Another interesting actor in the trust negotiation area is Traust [61, 62], which has been released by the same group of researcher that released Trust Builder. Traust is a third-party authorization service that leverages the strengths of existing prototype trust negotiation systems. Traust acts as an authorization broker that issues access tokens for resources in an open system after entities use trust negotiation to satisfy the appropriate resource access policies. The Traust architecture is designed to allow Traust to be integrated either directly with newer trust-aware applications or indirectly with existing legacy applications.

On other hand, Hess et al.[49] proposed a trust negotiation in TLS (TNT) handshake protocol by adding trust negotiation features.

Further, another prototype for trust negotiation designed for the World Wide Web is Protune [32, 21, 128, 20, 19], a rule-based trust negotiation system. By describing Protune, we will illustrate the advantages that arise from an advanced rule-based approach in terms of deployment efforts, user friendliness, communication efficiency, and interoperability. Protune relies on logic programming for representing policies and for reasoning with and about them

Several privacy-enabled identity management systems have been proposed, mainly based on the notion of anonymous credential [27, 30]. In anonymous credential systems, organizations know the users only by pseudonyms. Different pseudonyms of the same user cannot be linked.

Brands presents a form of digital credential in which a user has a single public credential, but that credential is pseudo-anonymous, even to the issuer [26]. The credential holds attributes that the user can selectively prove to a service provider. Repeated showings of the same credential are linkable, both if shown to the same or different service providers.

Camenisch et al. have proposed and implemented yet another framework for managing digital credential, referred to as Idemix [27]. While they describe a system for implementing a Chaum-like pseudonym system, their system is much more flexible, and can be used without pseudonyms. Idemix is the first system implementing anonymous credentials in a federated identity management system. Idemix provides mechanisms for efficient multi-show credentials and a flexible scheme for issuing and revoking anonymous credentials. It also provides a mechanism for *all or nothing* sharing and PKI-based non-transferability. Anonymous credentials however may not be adequate for real world e-commerce applications and web services that require disclosure of various attributes. In our approach, we do not require the user identity to be hidden, even if we protect his/her attributes. Additionally, none of the approaches discussed so far, take into account the possibility of suspending or accidentally interrupting a negotiation, nor they allow to delegate the negotiation process to any entity.

Again, Backes et al. also presented an attribute-based authentication mechanism using anonymous credentials [7]. Their work takes advantage of cryptographic protocols such as zero-knowledge protocols to anonymously authenticate the requester against attribute-based access control policies. Nevertheless, the presented approach does not allow the requester to specify access control policies on the attributes used in the verification process.

In order to orchestrate the negotiation process among multiple parties without a centralized moderator, they introduce an interesting diffusion negotiation protocol, that is, a set of message-passing conventions that allows parties to carry out a negotiation in a distributed fashion. This work, although related to ours, differs in its overall goal; our goal is not to support multi-party negotiators, but to ensure robustness and stability of the negotiations, regardless of the actual availability of the peers who originally started the process.

Our approach has some similarities with SecPal [10], a decentralized authorization language. SecPal is a declarative authorization language that supports domain-specific constraints and negation. SecPal represents a potential candidate for supporting the Trust-$\mathcal{X}$ protocol, although it would require several extensions. For example, SecPal does not provide a direct way to address credential-based policies, nor it provides an approach for specifying conditions against credentials. A desirable feature supported by SecPal is delegation, which may be very useful in case mobile negotiators decide to delegate not only the process but also the policies and credentials required to complete the negotiation on behalf of another party. However, SecPal does not include the infrastructure and the framework required to support trust negotiations. Research has also been conducted on logic-based access control languages for single administrative domains that do not require centralized delegation of authority. Many of them are based on DataLog [29] or DataLog with Constraints [65].

Our approach to the suspend and resume procedures described in the paper is similar to saving and restoring a partial proof graph in a Datalog-based trust management system. When the negotiation is suspended, one of the two negotiating parties save their copies of the partial proof graph (to-be-proven subgoals are similar to "open" subnodes) along with any variable mappings in effect. Claims that might expire can be denoted along side the proof graph. It would however be necessary to rescan all the provided claims on restart to determine in any of them had been revoked during the suspension period. Although DataLog provides a good foundation for access control in distributed systems, it lacks concrete applicability and does not efficiently model the realistic requirements of peer to peer interactions. Additionally, Datalog does not meet the practical need for policies about common structured resources, and needs to be augmented with constraints, which imply an increased complexity and even longer processing time.

Bowers et al. [23] developed an access control framework which uses a particular extension of linear logic – further extended by Garg et al. [44] by means of the introduction of the knowledge of the principals within the model – to deal with consumable credentials. In particular, the atomicity property of their framework is an alternative solution to the issue relative to the consumption of one-time credentials. By means of contract signing protocols their approach avoids the consumption of credentials performing a rollback of the transaction when the proof is not successful. However, our approach provides a more flexible solution consenting the parties to perform the verification of the properties over a longer period. Moreover, their approach requires the presence of online "ratifier" to guarantee the correctness of the proof while our approach does not involves external entities.

Trust negotiation systems have also been investigated with respect to privacy. Work along this direction has focused on the protection of sensitive policies and credentials. Winslett and Yu [127] have developed a unified scheme, known as *Unipro*, to model resource protection, which applies to both the actual resources to be protected and to the policies. Those approaches, however, are based on a notion of protection closer to the notion of access control. A formal framework for trust negotiations has been proposed by Winsborough and Li [117]. Their approach to safe enforcement of policies focuses on a privacy-preserving credential exchange. A formal notion of safety in automated trust negotiation is given, stating when a negotiation is secure against inferences that a party may make against the profile of the other party.

Another related work has been done by Ramakrishna et al. [86]. The authors propose a negotiation framework for ubiquitous systems. As in our work, the negotiation process between two peers is driven by local policies. The policies encode not only rules to carry on the negotiation but facts about the state of the system and of the negotiating peers. In the approach by Ramakrishna et al. the management of policies is centralized. Policies are stored in a single database and the access to these policies is managed by a policy engine. Additionally, a verification protocol to check the availability of the negotiated resources is not provided, neither during the negotiation nor after the negotiation is over. Also, the issue of detecting malicious peers is not addressed.

With respect to extensions to JXTA authentication resources, an approach similar to the one presented in the following of this thesis has been proposed by Kawulok et al. [57]. Their goal is to develop a Group Membership Service for JXTA extended with single or bi-directional authentication. The proposed solution supports bi-lateral authentication based on PKI but does not offer any approach for the peers to negotiate during the joining process, and it only focuses on authentication rather than trust establishment based on qualifying resources offered by peers and peer groups.

Chen et al. [31] have proposed Poblano, a decentralized trust model for JXTA. Such model represents trust relationships between peers, computed on the basis of quantitative and qualitative values, such as risks and confidence. Trust is not negotiated by the peers, but distributed across peers according to the estimated risks and confidence. This approach, although interesting, does not allow a single peer to evaluate the trustworthiness of the group during the joining process, rather it is suitable only once the membership is granted so that peers' reputation can be evaluated.

Finally, in relation to the issue on how to replace a failed group manager that is discussed in Chapter 4.3.2, finding suitable super-peers in P2P groups, according to given performance metrics, is an active research area. In particular, in [47, 88] it is pointed out that assessing metrics for identifying suitable candidates for super-peer role is not a trivial task and several techniques for finding the most suitable candidates are proposed. Two notable proposals for super-peers election are described in [111, 55]. However, such proposals do not fit within our system: in particular, the techniques deployed in [111] assume the adoption of routing mechanisms among peers of a structured, Chord-like P2P network, which is not supported by the JXTA-based systems. In [55] an adaptive, neural networks-based technique for identifying a super-peer is proposed. Such approach relies on an ontology (which is adaptively modified by a neural network), which classifies group peers upon their features. In our context, the deployment of such advanced classification techniques impose a too heavy computational burden. Hence, we have proposed a simple yet effective protocol to manage GMs' election, which takes advantage of the information about the group available in the GM's database.

Finally, we cite Trust-$\mathcal{X}$. Trust-$\mathcal{X}$ is a comprehensive framework for trust negotiation, providing both an XML-based language – referred to as $\mathcal{X}$-RNL, which is an extension of the previously provided $\mathcal{X}$-TNL [42] language – to encode policies and certificates, and a system architecture. Such framework was originally developed by Bertino, Ferrari and Squicciarini, which we extended in several directions [101, 105, 24, 106], including in this thesis, and it has been interested by other researcher [81, 103]

Chapter 3

The Trust-$\mathcal{X}$ policy definition language and trust negotiation architecture

3.1 Introduction

As mentioned in Chapter 1, the trust negotiation framework here presented uses a language called $\mathcal{X}$-RNL. In the current chapter will be presented the formalization of our trust negotiation system, the formal definition of the rule-based policy definition language supported by Trust-$\mathcal{X}$ and the formal definition of the resources supported.

The $\mathcal{X}$-RNL trust negotiation language is the natural evolution of the $\mathcal{X}$-TNL language[42]. Such new language allows the definition of policies using uncertified resources and the definition of requirements about properties of groups.

The application described in Appendix B implements a XML[118] serialization of the language here formalized.

3.2 Resources

We start by precisely stating what are the objects involved in the trust negotiation.

Assuming the existence of a set $\mathcal{RN}$ of resource names, a set $\mathcal{AN}$ of attribute names and – for every attribute name $a \in \mathcal{AN}$ – a corresponding set $\mathcal{V}_a$ of attribute values, we define the following building blocks which are required in order to be able to formalize what we mean with the term "resource".

First of all we need to define what an attribute is.

Definition 3.2.1 (Attribute)**.** *An* attribute *is a tuple (a, v), where $a \in \mathcal{AN}$ and $v \in \mathcal{V}_a$*

Attributes characterize resources as follows.

Definition 3.2.2 (Resource)**.** *A* resource *is a tuple* $(rn, AttrList)$ *where* $rn \in \mathcal{RN}$ *and* $AttrList$*, the so called* attribute list*, is a set of attributes.*

A particular kind of resource is represented by *credentials*. Informally, a credential is a resource whose content has been digitally signed [1] by a trusted authority.

Definition 3.2.3 (Credential)**.** *A* credential *is a resource signed by an external entity, which means that the attribute list associated with a resource C, of type credential, contains an attribute with name signature and an attribute with name* $signature_i ssuer$*. Each attribute has an appropriate value associated.*

3.3 Peers and Groups

The resources described in Section 3.2 are objects owned by some subjects in a distributed environment.

Considering that the Trust-$\mathcal{X}$ framework has been designed for a P2P environment, we will refer to the subjects as *peers*.

Definition 3.3.1 (Peer)**.** *A peer p is a tuple* $(ID, \mathcal{R}_p)$ *where* ID *is the peer identifies and* $\mathcal{R}_p$ *is the set of resources owned by p.*

Moreover, in P2P environment peers tend to cluster themselves in groups.

Definition 3.3.2 (Peer group)**.** *A peer group G is a tuple of the form* $(GID, \mathcal{P}eer_{GID})$ *where* GID *is the peer group identifier and* $\mathcal{P}eer_{GID}$ *is the set of peers operating in G, with* $\mathcal{P}eer_{GID} \neq \varnothing$

Such groups exist for different reasons, for example for sharing information or to collaboratively achieve common tasks.

The identification of why such groups exists and how they evolve is beyond the objectives of this thesis.

3.4 Policies

Having defined of the resource concept, we now pass to the problem of how to protect the resources owned by a subject. In our system, the access to the resources is controlled by means of rules called *disclosure policies*. Again, before being able to formally define the concept of disclosure policy we define their building blocks.

Definition 3.4.1 (Attribute condition)**.** *An* attribute condition $\mathcal{AC}$ *is a tuple* $(a, pred, v)$ *where* $a \in \mathcal{AN}$, $v \in \mathcal{V}_a$ *and pred is one of the binary predicates in* $\{<, \leq, =, \geq, >\}$.

[1] A digital signature or digital signature scheme is a cryptographic scheme for demonstrating the authenticity of a digital message or document.

Definition 3.4.2 (Resource condition). *A resource condition* `RC` *is an expression of the form*

$$rn(AL)$$

where $rn \in RN \cup \{``X"\}$ is a resource name and AL is a list, eventually empty, of attribute conditions.

Example 3.4.1. *Examples of resource condition are the following:*

- ID(name="John", surname="Doe")
- NonDisclosureAgreement(Company="FooBar", IssueData¡"2010-01-01", ExpireDate="2012-12-21")
- BankAccount(BankName="SomeBank", AccountID='X123456', ExpenceLimit $>=$ 400$)
- X(YearOfBirth$\leq$1981)

We can now define when a resource satisfies a resource condition.

Definition 3.4.3 (Resource condition satisfaction). *A resource condition rc is satisfied by a resource r if and only if, for each access condition $(a, pred, v) \in AL(rc)$, there exists $(a', v') \in AttrList(r)$ where $a' = a$ and $pred(a', a)$ state true and, if $rn(rc) \neq ``X"$, then $rn(rc) = rn(r)$.*

Definition 3.4.4 (Condition set). *A* condition set *$\mathcal{CS}$ is a not empty set of resource conditions.*

Intuitively, a condition set represents the conditions which have to be satisfied simultaneously. In other words all the resource condition rc_i in a condition set $\mathcal{CS}$, with $i \in \{1, \ldots, |\mathcal{CS}|\}$ are representable as a conjunctive formula of the form

$$rc_1 \wedge \ldots \wedge rc_{|\mathcal{CS}|}$$

Therefore, a condition set is satisfied if and only if all the conditions in it are satisfied.

Definition 3.4.5 (Condition set satisfaction). *A condition set $\mathcal{CS}$ is satisfied by a set of resources $\mathcal{R}$ if and only if, for each resource condition $rc \in \mathcal{CS}$, there exists a resource $r \in \mathcal{R}$ which satisfies rc.*

Definition 3.4.6 (Condition formula). *A condition formula CF is an expression of the form*

$$\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

where each $\mathcal{CS}_i$ is a condition set and $v \geq 1$.

Condition formulæ allow the composition of different condition sets as if they were connected with the disjunctive connector. Hence, a condition formula is satisfied if at least one of its condition sets is satisfied.

Definition 3.4.7 (Condition formula satisfaction). *A condition formula CF, defined over a set of condition sets $\{\mathcal{CS}_1, \ldots, \mathcal{CS}_v\}$, is satisfied by a set of resources $\mathcal{R}$ if and only if there exists a condition set $\mathcal{CS} \in CF$ which is satisfied by $\mathcal{R}$.*

3.4.1 Quantifications

The policy definition language $\mathcal{X}$-RNL has been designed to work in group environments, as it will be described in more details in Chapter 4. The concept of group will be discussed at length in Chapter 4, just consider a group as a collection of subjects sharing common interests and willing to cooperate.

The condition formulæ specify requirements which have to be satisfied by a single subject. On the other hand, when dealing with a group of subjects, such as with a group of peers, one may specify how many subjects have to satisfy such requirements.

$\mathcal{X}$-RNL does express this by means of *quantified formulæ*, where the quantification is over the subjects.

Both existential and universal quantification are possible as described in the follows:

Definition 3.4.8 (Universally quantified formula). *A* universally quantified formula *is an expression of the form*

$$\forall\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

where $\phi((\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ *is a condition formula.*

Definition 3.4.9 (Universally quantified formula satisfaction). *A universally quantified formula*

$$\phi' = \forall\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

is satisfied by a group of peers $G = \{p_1, \ldots, p_{|G|}\}$ *if and only if* $\forall p_i \in G$, $\exists \mathcal{R}_i$, *a set of resources owned by* p_i *which satisfies* $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$.

Definition 3.4.10 (Existentially quantified formula (with cardinality restriction)). *A* existentially quantified formula (with cardinality restriction) *is an expression of the form*

$$\exists^{\geq n}\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

where $\phi((\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ *is a condition formula.*

Definition 3.4.11 (Existentially quantified formula (with cardinality restriction) satisfaction). *An existentially quantified formula (with cardinality restriction)*

$$\phi' = \exists^{\geq n}\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

is satisfied by a group of peers $G = \{p_1, \ldots, p_{|G|}\}$ *if and only if* $\exists G' \subseteq G$, *with* $u \geq n$, *and* $\forall p'_i \in G'$, $\exists \mathcal{R}_i$, *a set of resources owned by* p'_i *which satisfies* $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$.

The possibility to state conditions about groups are counter-balanced by the possibility to define condition about specific members of the group. Such possibility is provided by means of the so-called *named formulae*

Definition 3.4.12 (Named formula). *A* named formula *is an expression of the form*

$$@p\, \phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

where p is a peer identifier and $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ is a condition formula

Definition 3.4.13 (Named formula satisfaction). *A named formula*

$$\phi' = @p\, \phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$$

is satisfied by a group of peers $G = \{p_1, \ldots, p_{|G|}\}$ if and only if there exists a peer $p' \in G$ such that $ID(p') = p$ and $\mathcal{R}_{p'}$, the set of resources owned by p' satisfies $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$.

3.4.2 Verification techniques

Beside expressing conditions about the number or the identity of peers which must satisfy a given formula, the $\mathcal{X}$-RNL language allows the specification of different kinds of verification techniques, differing in the time span in which the verification takes place.

At first, simple case of formula verification occurs when the truth value of the formula is verified only once (and then the time span collapse to a single instant). We call such kind of verification techniques *weak*.

Definition 3.4.14 (Weak assured formula). *A* weak assured formula *is an expression of the form*

$$\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)^w$$

where $\phi((\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ is a formula.

Definition 3.4.15 (Weak assured formula satisfaction). *A weak assured formula $\phi' = \phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)^w$ is satisfied if and only if at the verification time t the formula $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ is satisfied.*

A users may be interested in stronger requirements. Namely, one may be interested in a formula that is verified as true at the verification time and in the following. Such interest is satisfied by the $\mathcal{X}$-RNL policy definition language by means of the so-called *strong assured formulæ*

Definition 3.4.16 (Strong assured formula). *A* strong assured formula *is an expression of the form*

$$\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)^s$$

where $\phi((\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ is a formula.

Definition 3.4.17 (Strong assured formula satisfaction). *A strong assured formula $\phi' = \phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)^s$ is satisfied if and only if, given a verification time t, for each $t' \geq t$ the formula $\phi(\mathcal{CS}_1, \ldots, \mathcal{CS}_v)$ is satisfied at instance t'.*

The difference between strong and weak assured formulæ is relevant. Most of the times a user may willing to express requirements stronger than the ones expressible with a weak formula but less then what is expressible with a strong formula. $\mathcal{X}$-RNL satisfies such need with the *limited strong assured formulæ*, or *limited* for short. Informally, a limited formula is a strong formula that is verified true for a time interval starting from the verification time.

Definition 3.4.18 (Limited strong assured formula). *A* limited strong assured formula *is an expression of the form*

$$\phi(\mathcal{CS}_1,\ldots,\mathcal{CS}_v)^{t(n)}$$

where $\phi((\mathcal{CS}_1,\ldots,\mathcal{CS}_v)$ *is a formula and* $n \in \mathbb{N}$*.*

Definition 3.4.19 (Limited strong assured formula satisfaction). *A limited strong assured formula* $\phi' = \phi(\mathcal{CS}_1,\ldots,\mathcal{CS}_v)^{t(n)}$ *is satisfied if and only if, given a verification time* t*, for each* $t \leq t' \leq (t+n)$ *the formula* $\phi(\mathcal{CS}_1,\ldots,\mathcal{CS}_v)$ *is satisfied at instance* t'*.*

The different verification techniques are sketched in Figure 3.1. Moreover, weak and strong formula will be further presented in Chapter 4.

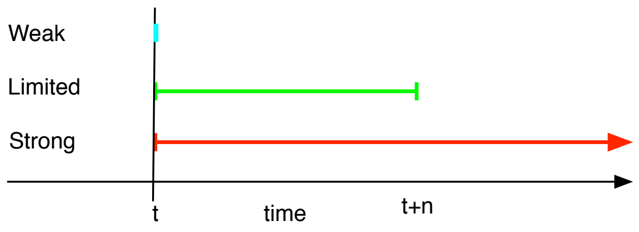


Figure 3.1: Differences between weak, strong and strong limited verification techniques

We have defined the toolkit needed for formally defining disclosure policies that, informally, is a rule expressing under what conditions a given resources may be accessed.

Definition 3.4.20 (Disclosure policy). *A* disclosure policy *is an expression of the form*

$$r \leftarrow \phi$$

where ϕ *can be, alternatively:*

- *a condition formula*
- *a universally quantified formula*
- *an existentially quantified formula*
- *a named formula*

- *a weak assured formula*
- *a strong assured formula*
- *a limited strong assured formula*

and r is a resource identifier.

Note that the symbol $\leftarrow$ is not the implication of the classical logic because, in our interpretation, the head (or left part) of the rule states true if and only if the body (or right part) of the rule is true. On the other hand, the implication in classical logic would state true even if the head is true and the body is false.

Example 3.4.2. *Examples of disclosure policies are:*

- $CreditCard \leftarrow BestBuyerID$
- $SensitiveDocument \leftarrow (PoliceID \wedge ID) \vee (MilitaryID \wedge ID)$

Definition 3.4.21 (Disclosure policy satisfaction)**.** *Given a resource r and a disclosure policy $DS = r \leftarrow \phi$ where ϕ is a formula, the disclosure policy DS is satisfied, which means the resource r can be disclosed, if and only if the formula ϕ is satisfied.*

After the satisfaction of the disclosure policy protecting a resource r, such resource can be disclosed to the group of peers satisfying the policy.

3.5 The subjects of the trust negotiation model

After defining the resources exchanged in a trust negotiation and how it is possible to specify rules to protect them, we formally define the subjects whose interactions create the trust negotiation.

Recalling the definitions of Section 3.3 and also considering what have been defined in Section 3.4, we can introduce the concepts *TNSubject* and $\mathcal{X}$-Profile.

Definition 3.5.1 (TNSubject)**.** *A TNSubject is one of:*

- *a peer p*
- *a group of peers* $G = \{p_1, \ldots, p_u\}$

Definition 3.5.2 ($\mathcal{X}$-Profile)**.** *Given a TNSubject S, the $\mathcal{X}$-Profile associated to S, denoted as $\mathcal{X}$-Profile$_S$, is a tuple of the form $\langle \mathcal{R}_S, \mathcal{P}_S \rangle$ where $\mathcal{R}_S$ is the set of resources owned by S and $\mathcal{P}_S$ is the set of disclosure policies defined by S to protect the resources in $\mathcal{R}_S$.*

So far we assumed that a resource is protected by at least a disclosure policy. This is not always the case, in fact, it is possible for a user to define no disclosure policy associated with a certain resource. Such possibility creates a particular set of resources, namely the freely available resources. These resources are called *deliverable resources*.

Definition 3.5.3 (Deliverable resource). *Given a subject S and the corresponding $\mathcal{X}$-Profile, denoted as $\mathcal{X}$-Profile$_S$, a resource $r \in \mathcal{R}(\mathcal{X}\text{-Profile}_S)$ is defined* deliverable, *or DELIV, if and only if $\nexists P \in \mathcal{P}(\mathcal{X}\text{-Profile}_X)$.*

Definition 3.5.4 (Negotiating peer). *A* negotiating peer *is a tuple of the form*

$$< p, \mathcal{R}_p, \mathcal{P}_p >$$

where p is the identifier of the peer, $\mathcal{R}_p$ is a set of resources owned by p and $\mathcal{P}_p$ is a set of resource policies defined by p to protect the resources in $\mathcal{R}_p$.

3.6 Trust Negotiation architecture

A trust negotiation is a protocol involving two negotiating subjects:

- a *controller* **Con** and
- a *requester* **Req**.

Each subject is associated to a $\mathcal{X}$-Profile, namely, $\mathcal{X}$-Profile$_{\mathbf{Con}}$ is associated to the controller while $\mathcal{X}$-Profile$_{\mathbf{Req}}$ is associated to the requester.

The trust negotiation begins when **Req** requests a certain resource r to **Con** and, by means of the resources and the disclosure polices contained in their respective $\mathcal{X}$-Profile, the negotiating subjects determine if there exists a certain trust level disclosing the resource r. As shown in Figure 3.2 and extensively described in [100, 13, 101], the trust negotiation protocol provided by the Trust-$\mathcal{X}$ framework is subdivided into three different phases:

- the *introductory phase*,
- the *policy evaluation phase* and
- the *credential exchange* phase.

Each phase must successfully terminate in order to successfully terminate the whole Trust Negotiation.

The improvements introduced into the Trust-$\mathcal{X}$ framework, in particular the ones presented in Chapter 5 and in [106, 105] allow a Trust Negotiation to be performed among different negotiation session. Nevertheless, in order to successfully terminate, the negotiation still requires to be carried out following the sequence of phases here described.

3.6.1 Introductory phase

The first phase executed is called *Introductory Phase*. During such phase the interacting peers identify the terms of the negotiation, which means that they agree on the language used in the negotiation ($\mathcal{X}$-TNL or alternatively $\mathcal{X}$-RNL) and they

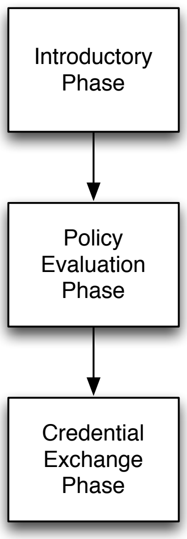


Figure 3.2: The negotiation phases

identify the resource which the requester wants to access, both stating the request or using more advanced techniques, such as the ones described in Chapter 6.

Moreover, during the introductory phase it is possible to perform all the accessory protocols, not directly expressible in the $\mathcal{X}$-RNL language, which are provided by the Trust-$\mathcal{X}$ framework. Such protocols will be presented in Chapter 5.

Eventually, before the end of the introductory phase, the system initializes the data structure which represents the progress of the trust negotiation. Such data structure is called *Negotiation Tree*. Informally, a negotiation tree is a labeled, multi-edge tree rooted in the initially requested resources where each node is a resource condition and the edges represent the exchanged disclosure policies.

Definition 3.6.1 (Negotiation Tree). *Given a resource controller* **Con** *and a resource requester* **Req** *and the corresponding* $\mathcal{X}$*-Profile*$_{\mathbf{Con}} = \langle \mathbf{Con}, \mathcal{R}_{\mathbf{Con}}, \mathcal{P}_{\mathbf{Con}} \rangle$ *and* $\mathcal{X}$*-Profile*$_{\mathbf{Req}}$ $= \langle \mathbf{Req}, \mathcal{R}_{\mathbf{Req}}, \mathcal{P}_{\mathbf{Req}} \rangle$, *a* Negotiation Tree *is a tuple of the form*

$$\mathcal{NT} = \langle \mathcal{N}, \mathcal{E}, r, \psi \rangle$$

where:

- $\mathcal{N}$ *is the set of nodes, each node* $n \in \mathcal{N}$ *corresponds to a resource condition.*
- $\mathcal{E}$ *is the set of edges, each edge* $e \in \mathcal{E}$ *represents the disclosure policy protecting the resource which is the parent node of the edge. More precisely, each condition set contained in a the represented disclosure policy brings to the creation of a set of edges. Such edges can be of two kinds: simple edges and multi-edges. A* simple edge *is used to model condition sets having only one resource condition while a* multi-edge

links several simple edges in order to represent condition sets having more than one resource condition. Nodes belonging to a multi edge are thus considered as a whole during the negotiation.

- *$r \in \mathcal{N}$ is the root of the tree with also $r(r) \in \mathcal{R}_{\mathbf{Con}}$.*
- *ψ is the labeling function defined as:*

$$\psi : N \rightarrow Label$$

where $Label = \{DELIV, UNDELIV, OPEN\}$ is a set of states in which each node can be.

The use of a tree allows one to detect cycles in the policy exchange. Repeated or recursive policies can be easily detected in the tree, as a they appear twice in the same sequence. Furthermore, a tree allows one to store all alternative sequences of policies in a unique structure, rather than having a list for each alternative. A more detailed description of the negotiation tree can be found in[100].

An example of negotiation tree is shown in Figure 3.3. Namely, Figure 3.3(a) shows the negotiation tree returned by the initialization phase.

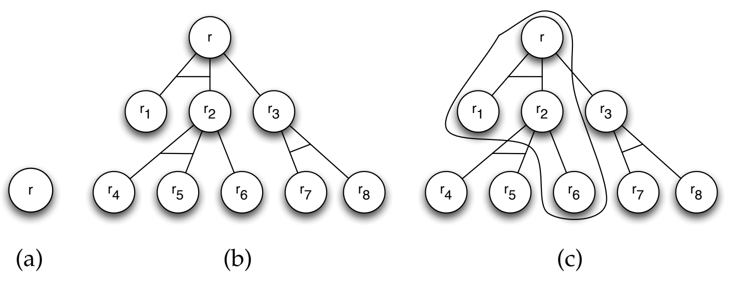


Figure 3.3: An example of negotiation tree growth: (a) after the introductory phase, (b) after the policy evaluation phase and (c) with a valid view highlighted

3.6.2 Policy evaluation phase

The *Policy Evaluation Phase* is the most relevant one and it is the most expensive phase of a trust negotiation. In such phase each peer iteratively exchanges the disclosure policies in charge of protecting the resources requested by the disclosure policies by the other one.

The exchange of the disclosure policies expands the negotiation tree. Namely, as introduced in Definition 3.6.1, the exchanged disclosure policies define the next level of the negotiation tree. When a node is added to the negotiation tree, it is associated to the label *OPEN* which means it is unknown whether it will be possible to successfully verify the resource condition represented by such node.

When a peer is requested a deliverable resource (see Definition 3.5.3), such resources is inserted into the negotiation tree with label $DELIV$. After that, the labeling function propagate the $DELIV$ state to the ancestors of the newly inserted node. This is done by checking the label of the children of the parent node. If all of the children have label $DELIV$ then the parent node is labeled as $DELIV$ too. Such procedure is applied to the parent of the currently modified node and so on. If the propagation of the $DELIV$ label reaches the root of the negotiation tree then the policy evaluation phase successfully ends.

Similarly, when a peer requests a resource not present in the $\mathcal{X}$-Profile of the other, then the corresponding (unsatisfiable) resource condition is added to the negotiation tree as a node labeled as $UNDELIV$. The $UNDELIV$ label is propagated by the labeling function as the $DELIV$ label with one exception: namely, a node is labeled as $UNDELIV$ if and only if all of its children are labeled as $UNDELIV$. Moreover, if a node labeled as $UNDELIV$ is part of a multi-edge then all its siblings in the edge are labeled as $UNDELIV$. If the propagation of the $UNDELIV$ label reaches the root of the negotiation tree then the policy evaluation phase fails making un-successfully terminate the whole trust negotiation.

In addiction, to maintain a complete log of the exchanged policies, each negotiator updates the tree with the local policies prior sending them to the counterpart.

The successful completion of the policy evaluation phase is signaled by a subtree of the tree rooted at the requested resource consisting only of nodes labeled as *DELIV*. We refer to such subtree as *Valid View*.

Definition 3.6.2 (Valid View)**.** *Given a negotiation tree* $\mathcal{NT} = \langle \mathcal{N}, \mathcal{E}, r, \psi \rangle$ *a* valid view *is a tuple of the form*

$$\langle \mathcal{N}', \mathcal{E}', r \rangle$$

where $\mathcal{N}' \subseteq \mathcal{N}$ *and* $\forall n \in \mathcal{N}', \psi(n) = DELIV$. $\mathcal{E}' \subseteq \mathcal{E}$.

An example of valid view in shown in Figure 3.3(c).

Note that more then one valid view may be found in a given negotiation tree. Again, the details about how to handle the presence of different valid view and some informations related to the different negotiation strategies which the negotiating peer may carry out are described in [100].

3.6.3 Credential Exchange Phase

The last phase of the trust negotiation protocol provided by Trust-$\mathcal{X}$ is the *Credential Exchange Phase*. During such phase the negotiating peers exchange the credentials in the valid view obtained by the successful termination of the policy evaluation phase.

Before exchanging resources, the negotiating peers must identify the order in which such resources must be exchanged. Such order can be extracted from the valid view. We refer to it as *trust sequence* or *credentials sequence*.

Definition 3.6.3 (Trust Sequence)**.** *Given a resource controller* **Con** *and a resource requester* **Req** *and the corresponding $\mathcal{X}$-Profile$_{\mathbf{Con}} = \langle \mathbf{Con}, \mathcal{R}_{\mathbf{Con}}, \mathcal{P}_{\mathbf{Con}} \rangle$ and $\mathcal{X}$-Profile$_{\mathbf{Req}}$ $= \langle \mathbf{Req}, \mathcal{R}_{\mathbf{Req}}, \mathcal{P}_{\mathbf{Req}} \rangle$ and a valid view $\mathcal{VV} = \langle \mathcal{N}', \mathcal{E}', r \rangle$, the corresponding* Trust Sequence *$\mathcal{CS}seq$ is a tuple of the form*

$$\langle r_1, \ldots, r_u \rangle$$

Where each r_i with $i \in \{1, u\}$ is a resource in $\mathcal{R}_{\mathbf{Con}} \cup \mathcal{R}_{\mathbf{Req}}$ and such that there exists among the resources an order satisfying the disclosure policies represented by the edges of the valid view.

The trust sequence $\mathcal{CS}eq$ can be built by traversing the view according to a specified order defined by the labeling function associated with the negotiation tree, beginning from the leaves.

Example 3.6.1. *As an example, the trust sequence computed on the valid view highlighted in Figure 3.3(c) is the following:*

$$\langle r_6, r_2, r_1, r \rangle$$

Hence, the resources are disclosed by both peers following the order dictated by the trust sequence. Finally, the trust negotiation protocols ends successfully if and only if all the exchanged resources actually satisfy the disclosure policies defined by the negotiators.

To summarize, a successful negotiation is typically defined as follows.

Definition 3.6.4. *Let p_1 and p_2 be two negotiation peers, and let $\mathcal{X}\text{-}Profile_{p_1}$, $\mathcal{X}\text{-}Profile_{p_2}$ be the profiles of p_1 and p_2, respectively.*

A trust negotiation protocol for a resource r is successful if and only if a trust sequence $\langle r_1, \ldots, r_n = R \rangle$ is found, where $r_i \in \mathcal{R}(\mathcal{X}\text{-Profile}_{p_1}) \cup \mathcal{R}(\mathcal{X}\text{-Profile}_{p_2})$, and such that for each r_i, with $i \in \{1, n\}$, the associated disclosure policy $DP_i = r_i \leftarrow \phi_i$ is satisfied by a set of resources $\{r_{j_1}, \ldots, r_{j_k}\}$ with $j_l < j_{(l+1)}$ and $\forall j_l, j_l < i$.

Chapter 4

Group-based Trust Negotiations

4.1 Introduction

As introduced in Chapter 1, a relevant feature of P2P systems is the support for the dynamic formation of peer *groups*.

Moreover, as shown in Chapter 3, we use a highly expressive negotiation language, called $\mathcal{X}$-RNL, able to support the specification of a large variety of conditions applying to single peers or groups of peers. We remark that $\mathcal{X}$-RNL supports group-based policies, in which the evaluation of policies may involve more than one peer (see Chapter 3.4). The specification of group-based policies is supported by the introduction of a limited form of quantification over the number of peers in a peer group.

A second important feature of $\mathcal{X}$-RNL is the possibility of expressing policies that include not only conditions on credentials but also on other types of resources.

Our negotiation-based join protocols include algorithms for policies exchange and resource verification. A unique feature of our resource verification algorithms is that the resources' properties are verified not only at the end of the join process, but throughout the peers' life-cycle in the group. In order to guarantee such degree of control over the group's resources, without falling into a centralized approach, we support event-based checking protocols. Peers propagate to special peers, referred to as group managers, event-based messages, upon detection of events of interest, such as unavailability of resources, non-responsive peers etc. The messages trigger checking protocols that verify disclosure policies which may be affected by the events, and not be any longer verified by the peers. As we show in our experimental evaluation, such protocols help in efficiently detecting free riders, misbehaving peers or even malicious peer groups.

To validate our framework, we have in fact developed a new type of JXTA [75] Membership Service. JXTA has a number of important features that make it suitable for our benchmark. First, it has a set of standard protocols to implement P2P applications. Second, it is independent from the programming languages, transportation protocols and system platforms being used. Therefore, by adopting JXTA, peers can communicate seamlessly across different P2P system and different communities. Our results show that our negotiation-based join approach is robust to malicious peers, which are detected both during the negotiation and during the peer group lifetime. Regardless of the peer group cardinality and interaction frequency, the peers always detect possible free riders within a small time frame. The peer finding a missing item notifies a group manager peer which promptly triggers a checking protocol and informs the peers of the group which indicated the missing resource as a strong requirement for their joining.

4.2 Application Scenario

We consider a file sharing P2P network called *ShareForFree* in which peers can exchange different types of resources such as books, music, movies and videos, software, photos, recipes etc or they can even sell stuff such as cars or post advertisements for rentals or house selling. Peers are organized in groups based on the type of the resources they are going to share, sell or buy. Specifically, we consider the following group of peers:

FreeBooks The peers belonging to this group exchange books spanning across different categories such as educational books, narrative books etc.

FreeMusic The peers in this group share mainly mp3 files.

FreeMovies The peers in this group exchange videos and movies.

FreeSoftware The peers of this group share open source software.

SharePhotos The peers of this group exchange pictures related to different topics.

CookForFun The peers of this group exchange recipes.

Housing The peers in this group share advertisements about houses for sale or rentals.

ForSale The peers post advertisement about any kind of item they want to sell or buy such as cars or electronics.

Figure 4.1 sketches the group organization of the considered P2P network.

To be a member of these groups, peers have certain properties or meet requirements related to the resources they provide to the other peers. For example, to be a member of the group **FreeSoftware**, a peer does not have to make available to the other peers in the group licensed software; or to be a member of the **FreeMovies** a peer has to share adult content movies only with adult peers, or to be a member of **FreeMusic**, a peer has to make available at least 1GB of mp3 files. We assume there is a peer hosting a service managing the memberships of peers in the group, and a peer that is responsible for monitoring changes that can occur in the group.

With respect to the policy definition language $\mathcal{X}$-RNL described in Chapter 3.4, an example of policy for the group described in the current section is what follows:

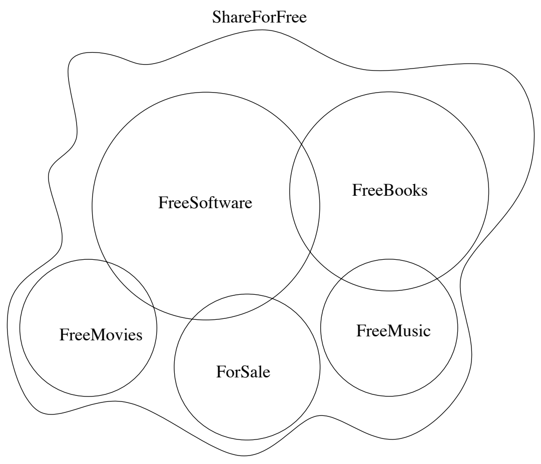


Figure 4.1: The *ShareForFree* P2P network.

Example 4.2.1. *Let us consider the* ShareForFree *file sharing networking and the group **FreeMusic**. Let assume that a peer p wants to join **FreeMusic**. The membership policy for **FreeMusic*** $FreeMusicMembership \leftarrow @p((Mp3(size > 500MB) \wedge State = California)$ *requires that p shares with the members of the group at least 500MB of Mp3 files and that it is from the state of California. On his side, p is only interested to join **FreeMusic** only if the group members provide a good selection of pop and jazz music. p requirements to join **FreeMusic** are expressed by the policy*

$$\begin{aligned} @p((Mp3(size > 500MB)) \quad \leftarrow \\ \exists^{>100} FreeMusic.x(@FreeMusic.x(Mp3(Singer = Madonna \vee \\ Singer = \text{Michael Bublé} \vee FrankSinatra \wedge size > 1GB)) \end{aligned} \tag{4.1}$$

*specifying that p will share at least 500MB of Mp3 files with the members of **FreeMusic** there are at least 100 peers in the group who make available a 1GB of Mp3 songs of Madonna, or Michael Bublé or Frank Sinatra.*

The satisfaction of a formula is evaluated with respect to the peer local resources, as given by the peer's advertisement and credentials, as discussed in the next sections.

4.3 Group Managers and Peer Members

4.3.1 Group Managers

In order to support our negotiation-based joining process, we introduce a new peer type, namely the *Group Manager* (GM for short). The GM controls the overall group state, by collecting and managing information about peers and peers' resources availability. The GM, besides acting as a super-peer, operates as a standard peer, and offers resources as any other peer member. In order to prevent reliability and security problems, one can assume that the GM is physically replicated on different machines. We refer to existing protocols [33, 66] for details regarding the management of GM replicas. We assume that the peer initiating the group becomes the first GM. The GM stores information about the group and its peer in a database. Such database can be accessed in a client/server fashion by any peer in the group as described in the next section, upon the retrieval of the access credential owned by the GM. Thus, the availability of the database does not depend on the availability of the current GM.

The GM plays an important role in the resource verification process, while ensuring a decentralized approach to the negotiation process. Specifically, its main role is to ensure the validity of the *strong formulae* over time.

To be able to do so, the GM collaborates with the peers in the group to be updated on the peer group changes, as further discussed in Section 4.4.2. Group members notify the GM about the occurrence of events that may affect the validity of strong formulæ.

Being the services offered by the GM very relevant components of our framework, it is necessary to ensure their reliability.

4.3.2 Group Manager Election

In case the current GM of a group becomes unresponsive or is unavailable for an extended time length, the other peers of the group initiate a protocol to replace it. Several sophisticated approaches for finding a super-peer – as those briefly described in Chapter 2 – can be deployed. However, since such approaches rely on a number of tools and strong assumptions, we here present a simple yet effective protocol for effective management of GMs.

Our protocol relies on the information stored in the GM database in order to identify the peer that represents the best candidate to take the role of the GM. As mentioned, the GM database is a GM-independent, persistent repository that stores several information about group members. Given the sensitive nature of the information stored in the GM database, usually the current GM only is allowed to access it. The right to access the GM database is encoded in an access credential, which is owned by the GM. The access credential contains the information required by a peer to access the GM database such as, for example, the database URL, a valid username and the corresponding password. Note that, additional information stored in the access credential depends on the type of database used.

In order to grant the possibility to access the GM database to other peers, the GM splits the access credential into shares, using a Secret Sharing Scheme (SSS) (e.g. [95]). The motivation of splitting the GM database access credential into shares is to prevent access to such database by unintended peers. In fact, shares are computed in such a way it is not possible to reconstruct the access credential from a number of shares less than a integer threshold M. Such threshold is an input parameter of the SSS, as the total number of shares, which is equal to the group cardinality (as registered into the GM database). Of course, the higher the threshold is, the harder for a set of peers is to collude.

After having computed the shares, GM gives each peer p_i a corresponding share s_i. In this way, for a peer (other than GM, of course) to access the GM database, it is necessary to obtain a number of shares larger than M.

The sharing of shares of the database access credentials enables the election protocol that is described as follows (see Algorithm 1 and 2):

- The request for identifying a new GM comes from a group member p_i that, after having sent a request to the GM – for example p_i wants to share a new resource r within the group – does not receive any answer from it. Thus, p_i assumes that GM is down and starts a procedure for electing a new GM.

- Peer p_i starts by requesting the shares of the credential for accessing the GM database to the other group members. The GM database stores relevant information about group members (e.g. group access time, number of advertised resources, along with corresponding lists of their features) that may drive the choice of the next GM (e.g. the oldest group member, or the member that advertises the largest amount of computational power, etc.). Such election criteria are a-priori fixed and shared among group members. All the messages exchanged among peers during the election protocol are signed with the key contained in the membership credential released by the Membership Service to each peer joining the group. A detailed description of the credential can be found in Section 4.4.1.

- The group members' positive answers to p_i's request depend on the verification of the signature of p_i on the request message and whether GM is actually unresponsive for a long enough (equal for all peers) period of time. If p_i signature is valid, then the other group members before disclosing the shares, check whether the GM is unavailable as indicated by p_i. In case the GM is actually down, the peers proceed by sending their credential shares to p_i. After having disclosed their shares once, the peers do not accept any subsequent shares disclosure request in the same election protocol run. In the unfortunate case that more than one peer initiate at the same moment the GM election protocol, it can be that none of them is able to obtain the sufficient access credential shares. In this case, when an initiating peer receives a share disclosure request from another initiating peer, one of the two agrees to release all its collected shares to the other, according to a previously agreed upon criterion (e.g., the shares are collected by the oldest peer). In this way, only one peer is granted the right to access the GM database. Afterwards, p_i reconstructs the credential and accesses the GM database.

- Once p_i has accessed the database, it identifies (according to the GM choice criteria) the new GM among the group members, as listed in the (old) GM database. Finally, the new GM is notified to the other group peers and begins to operate.

Notice that a malicious peer p_i wishing to access the GM database while the current GM is still up cannot retrieve all the required shares from the other peers. This is because every peer contacted by p_i checks whether the current GM is effectively down. If a sufficient number of peers succeeds in contacting the actual GM, the requested shares are not disclosed to p_i. The issue of colluding peers is still open, in that peers could possibly join the shares to access sensitive information. This aspect is however beyond the scope of the current work, and we do not further elaborate on it.

```
Input: $GM_{old}$: GroupManager identifier; $p_i$: initiator peer identifier; $s_i$:
       owned share; $M$, integer threshold; $Cert_i$: membership certificate
       released by the GM; $SK_i$: secret key of $p_i$
Output: New GM identification if the old one is unavailable
begin
    if IsGMOffline($GM_{old}$) then
        $ElectionInitiator = true$;
        GroupSend($p_i$, ``Share request'', Sig$_{SK_i}$(``Share request''),
        $Cert_i$);
        $Shares\_Set$ = GroupReceive();
        $Shares\_Set = Shares\_Set \cup s_i$;
        if $|Share\_Set| \geq M$ then
            $d$ = Interpolate($Shares\_Set$);
            Access Group Manager Database;
            Identify new GM $GM_{new}$ according to the choice criteria;
            Notify$GM_{new}$;
            Delete $d$;
```

Algorithm 1: New Group Manager Election: *ElectionInitiator*

4.4 The Negotiation-based Join Process

We now discuss how we extended JXTA in order to carry on negotiations. We present the peer's negotiation protocol, followed by a discussion on the formulæ verification process, which is one of the main features of our proposal.

4.4.1 Peer Negotiation

The join-based negotiation consists of three phases: the *advertisement discovery* phase, the *policy evaluation* phase, and the *verification* phase. Figure 4.2 reports the main steps of the protocol, for each of the negotiators. As reported, a follow-up component of the negotiation is given by the periodic verification of strong formulae, as further discussed in the remaining of the section.

```
Input: $GM_{old}$: GroupManager identifier; $s_l$: owned share
begin
    $\{Sender, Message, Sign, Cert_{Sender}\}$ = Receive();
    if VerifySign($Message$, $Sign$, $Cert_{Senter}$) then
        if $Message$ == "Share request" then
            if $ElectionInitiator == false$ then
                if $ShareAlreadySent == false$ then
                    if IsGMOffline($GM_{old}$) then
                        Send($Sender$, $s_l$);
                        ShareAlreadySent = $false$;
                    else
                        $p_k$ = IdentifyTheOldestElectorInitiator();
                        if $Sender == p_k$ then
                            Send($Sender$, $s_l \cup ReceivedShares$);
```

Algorithm 2: New Group Manager Election: $Peer_i$

The negotiation-based join process is carried out following the same interaction protocol used in the existing JXTA membership services [110]. The membership service (MS, for short) is a software component instantiated at the time the group is created. This service is physically located at the peer originating the group, which may be a GM. The group disclosure policies used by the MS to carry out the negotiation on behalf of the group are a combination of individual peers' policies, exchanged during their join procedure. Such policies are combined using conjunction operators for strong universal formulae, and using disjunction operators for all other formula types. Strong universal formulae are conjoined in order not to exclude any of the peer's resources. Note that, obsolete policies are discarded after a certain time interval set up by the group, or when peers leave.

The core phase is the *policy evaluation* phase, carried out to determine if the incoming peer p meets the peer group's joining requirements and, conversely, if p's requirements are satisfied by the peer group.

The actual policy exchange occurs between the incoming peer p and G's MS. Once the peer and membership service reach a mutual agreement on the policies to verify, the corresponding resources are identified. The verification of peers' resources availability is carried out according to the level of assurance of the policy formulae. *Weak formulae* express requirements that must be satisfied when p joins the peer group G. Therefore, the verification of weak formulae is performed one time only, at the end of policy evaluation phase. On the contrary, *strong formulae* express requirements that must also be satisfied for all the duration of p's membership. As a consequence, *strong formulae* are verified after the policy evaluation phase, and during p's membership. This type of verification is for critical resources,

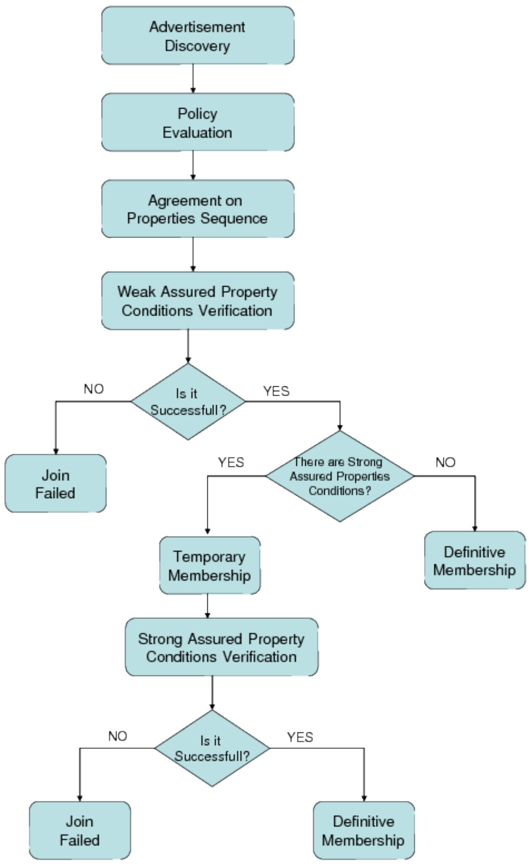


Figure 4.2: Main phases of the negotiation process

which the peers need to periodically access over time.

Advertisement Phase Peer p first discovers the advertisement of G. G's advertisement specifies the Membership Service that adopts resource negotiation as authentication method for the joining peers. Thus p, through MS advertisement, locates and runs an instance of the resource and, hence, starts the joining process.

Policy Evaluation Phase This phase starts when p runs requests to join G. It consists of a bilateral policy exchange between p and MS, where MS is hosted by a peer member. The goal of the policy evaluation phase is to authenticate both the joining peer and the group, based on the disclosure policies enforced by both p and MS.

The policy evaluation phase is carried out as follows. Upon p's request of joining group G, MS sends to p its policy. The policy specifies the trust requirements to meet, expressed by credentials' requests, as well as by the resources to make available to G, in case of a successful negotiation. At this stage, the policy is of the form $MembershipCred \leftarrow \phi(t_1, \ldots, t_u)$ where *MembershipCred* is a shorthand for the membership credential that p is requiring and $\phi(t_1, \ldots, t_n)$ is a formula that specifies conditions that p should satisfy. Moreover, *MembershipCred* is the credential which identifies a peer member of the group. It includes the following fields:

- the group identifier;
- the GM public key;
- a certificate issued by the GM containing the identifier and the public key of the joining peer. Such certificate is signed using the private key of the GM;
- the joining peer private key.

Note that the membership credential *MembershipCred* is sent through a secure channel, like SSL, at the end of a successful negotiation.

Peer p then verifies how to satisfy each term t_i in $\phi(t_1, \ldots, t_n)$, by checking its credentials and/or advertisement, depending on what is required. If p does not have the requested resources/credentials, and/or does not satisfy the conditions stated by the MS's policy, it informs the MS, that halts the negotiation process.

Otherwise, p checks whether there are policies regulating the disclosure of such resources to MS. If this is the case p replies by sending the counter policies to MS. p counter reply includes its own set of requirements that need to be satisfied by MS in order for p to join G. These requirements can be related to the specific policies that p received from MS, or they can express general trust requirements about the group that p wants to check before joining it.

Similarly, MS, upon receiving p's policies, verifies whether the received policies can be satisfied. Since MS carries on the negotiation on behalf of the members of the group, it access the GM database which stores information about resources that current peer members make available, so to be able to reason about quantified formulae along with the policies regulating the disclosure of these resources. Such disclosure policies are defined by the resource owners' peers, although managed and negotiated by *MS*. Information about resources of peers that are no longer members are removed from the cache as peers leave the group. Notice that this approach allows MS to release information about the peers that are members of G without contacting them (peers leaving the group notify the Membership Service). If the MS's local cache contains the resources corresponding to terms in p's policies and they all are available, MS checks whether the release of such resources is ruled by policies. To keep track of the policy exchange order p and MS maintain a negotiation tree (see Definition 3.6.1).

As described in Chapter 3.6.2, each negotiating party updates its local copy of the tree every time policies are received from the counterpart and evaluated as satisfiable. When neither p nor MS have additional policies to exchange and a sequence of logically related policies are satisfiable, the two negotiators traverse their local tree from a leaf node to the root to determine the sequence of formulæ that have to be verified. The sequence is then partitioned into two lists, one collecting the policies formed by the strong formulæ and the other one collecting weak formulæ. Recall that, according to our language, each policy corresponds to one single formula, which can either be strongly or weakly assured.

Example 4.4.1. *Let consider the* ShareForFree *file sharing networking and the group* ***FreeSoftware****. Let assume that a peer p wants to join* ***FreeSoftware****. The Membership Service of* ***FreeSoftware*** *group (denoted as* ***FreeSoftware****.MS) sends to p the membership policy for* ***FreeSoftware*** $FreeSoftwareMembership \leftarrow @p(Software(Licenced = False \wedge size = 1.5GB)$ *requires that p shares with the members of the group at least 1.5 GB of open source software (the XML representation of this policy is shown in Figure 4.3). On his side, p is interested to join* ***FreeSoftware*** *only if the peers in the group provides word processing, antivirus, and threats removal tools. Thus, p sends to* ***FreeSoftware****.MS the disclosure policy*

$$\begin{aligned} @p(Software(Licenced = False \wedge size = 1.5GB) \\ \leftarrow \\ \forall FreeSoftware.x(@FreeSoftware.x(Software(Type = WordProcessor \\ \wedge Type = Antivirus \wedge Type = ThreatRemover) \end{aligned}$$

specifying that p will share 1.5 GB of open source software if each peer in ***FreeSoftware*** *shares word processing, antivirus, and threats removal tools.* ***FreeSoftware****.MS checks if in his local cache there are for each peer in the group the advertisments for word processing, antivirus, and threats removal tools. If this is the case, then* ***FreeSoftware****.MS controls if each peer in the group has a disclosure policy for the release of these tools. The peers* p_1*,* p_2 *and* p_3 *have the following disclosure policies for the sharing of the word processing, antivirus, and threats removal tools:*

- *According to disclosure policy in Equation 4.2, in order for* p_1 *to share the tools, p has to provide a pdf editor.*
- *On other hand, according to the disclosure policy shown in Equation 4.3,* p_2 *provides the tools only if p makes available a JAVA IDE.*
- p_3 *shares the tools only if p shares a mobile browser simulator, see Equation 4.4.*

$$\begin{aligned} @p_1(Software(Type = WordProcessor \wedge \\ Type = Antivirus \wedge \\ Type = ThreatRemover)) \leftarrow \\ @p(Software(Type = PdfEditor)) \end{aligned} \tag{4.2}$$

$$\begin{aligned} @p_2(Software(Type = WordProcessor \wedge \\ Type = Antivirus \wedge \\ Type = ThreatRemover)) \leftarrow \\ @p(Software(Type = JAVAIDE)) \end{aligned} \tag{4.3}$$

$$\begin{aligned} @p_3(Software(Type = WordProcessor \wedge \\ Type = Antivirus \wedge \\ Type = ThreatRemover)) \leftarrow \\ @p(Software(Type = MobileSimulator)) \end{aligned} \tag{4.4}$$

Since p is not able to provide the pdf editor, the JAVA IDE and the mobile browser simulator, the policy evaluation phase fails and as a consequence also the joining process fails.

```
<?xml version="1.0" encoding="UTF-8">
<policies target="FreeSoftwareMembership">
<policy>
<term version="strong">Software(Licensed = False)</term>
<term version="strong">Software(size = 1.5GB)</term>
</policy>
</policies>
```

Figure 4.3: *FreeSoftware* Group Membership policy's XML representation

Notice that the negotiating parties can strengthen the security of the negotiation process as described above by adopting negotiating strategies. Several strategies can be employed, such as the trusting or the suspicious strategies [101] as supported by the Trust-$\mathcal{X}$ system. By carefully selecting the credentials and policies to disclose, parties can minimize the risk of information leakage and carry out safe and correct negotiations [117].

Resource Verification Phase Once the negotiation parties have reached an agreement about the resources to verify in order for p to join G, the verification phase begins. During such phase, resources' availability and their actual attributes are checked, in order to verify whether the requirements expressed in the exchange policies are met. We discuss the resource verification protocols in great details in the next subsection.

4.4.2 Event-Based Formulae Verification

In this section we illustrate the protocols for strong and weak formulæ verification. The main difference between strong and weak verification is for how long the validity of the credential or the availability of the resource is checked. As mentioned, a weak formula must be verified only at the end of the policy exchange phase, while a strong one must be verified true for all the temporal validity of peer p's membership.

The availability of the resources to which the formulae apply is determined according to the type of the resources. Specifically, in case the resource is a credential, once disclosed, its authenticity has to be verified. In case of a resource, its availability and main properties are checked, based upon the information contained in the resource advertisement. Although not implemented by us, an approach similar to [34] for resource verification can be actually employed.

Since the satisfiability of strong formulae must hold for peers' lifecycle in a peer group, there is a need for a mechanism that tracks the changes in peer group status. Events of interest relate to peers leaving the group or to peers that no longer make available the promised resources to the other members. To detect variations in the status of a peer group, we propose an event-based collaborative verification process.

Before describing such event-based collaborative process, we illustrate the main steps of the formulae verification process. The outcome of the policy evaluation consists of two lists of formulae, obtained from the peers' exchanged policies: $Weak\Phi\_List$, the list of *weak* formulae and $Strong\Phi\_List$, the list of *strong* formulae. Each formula in $Weak\Phi\_List$ ($Strong\Phi\_List$) is mapped onto actual resources that p or that the members of peer group G need to prove to possess. Specifically, for a formula of the form $\Phi(t_1, ..., t_n)$, the availability of the n resources $R_1, \ldots, R_n$ corresponding to terms $t_1 \ldots, t_n$ needs to be verified.

The pseudo code for the verification protocol is described in Algorithm 3. Algorithm 3 takes as input the identifier of the two parties (*Party* and *Counterpart*), which correspond to the incoming peer member and the peer that ran the negotiation on behalf of the group, along with the associated list of weak/strong assured formulae to verify, and *Form_Seq* the sequence of formulae they have agreed upon.

If the formula $\Phi(t_1, \ldots, t_n)_i$ must be satisfied by *Party*, the local party running the algorithm calls `RetrieveInfo` that returns the information to be provided to *Counterpart* in order to prove that $\Phi(t_1, \ldots, t_n)_i$ is satisfied by *Party*. The selection of the information to be sent to *Counterpart* varies according to the type of the formula $\Phi(t_1, \ldots, t_n)_i$ and to the type of the resources R_1,...,R_n. `RetrieveInfo` returns the set *Cred_Set* of credentials and the set *Adv_Set* of advertisement to prove the satisfiability of formula $\Phi(t_1, \ldots, t_n)_i$. By contrast, if $\Phi(t_1, \ldots, t_u)_i$ must be satisfied by the *Counterpart*, for each credential in *Cred_Set*, *Party* verifies the validity of the credential and for each Adv_i in *Adv_Set*, *Party* tries to access resource R_i.

If $\Phi(t_1, \ldots, t_n)_i$, is a strong assured formula, *Party* invokes the method `ToEnsure` to notify the GM that Φ is to be ensured as long as *Party* remains in the group. Once notified, GM tracks the changes of the availability of each resource R_i present in Φ during the temporal validity of *Party* membership.

Example 4.4.2. *Let us consider the* ***ShareForFree*** *file sharing networking and the following disclosure policies of Example 4.4.1 exchanged during the joining process between peer p and the membership service of* ***FreeMusic*** *group denoted as* ***FreeMusic****.MS:*

- $\forall FreeMusic.x(@FreeMusic.x(Mp3(Kind = Classic) \leftarrow @p((Mp3(kind = jazz)^{w}$
- $@p((Mp3(kind = jazz) \leftarrow \exists^{>100} FreeMusic.x(@FreeMusic.x(Mp3(Singer = Madonna \vee Singer = \text{Michael Bublé} \vee FrankSinatra \wedge size > 1GB)^{s})$

*The first disclosure policy is sent by **FreeMusic**.MS to p. During the formulae verification phase p has to send to **FreeMusic**.MS the advertisement of a jazz Mp3 file to prove that it satisfies the formula* $@p((Mp3(kind = jazz)^w$. *Since the formula is a weak formula, **FreeMusic**.MS has only to verify the validity of the advertisement and try to download the Mp3 file. If the download of the file starts, the formula is verified by p.*

*The second policy, instead, is sent by p to **FreeMusic**.MS which has to check that the strong formula* $\exists^{>100} FreeMusic.x(@FreeMusic.x(Mp3(Singer = Madonna \lor Singer =$ Michael Bublé $\lor\ FrankSinatra \land size > 1GB)^s)$ *is verified by group **FreeMusic**.*
***FreeMusic**.MS, in order to prove that the formula is satisfied, retrieves through a look up process the advertisements of Mp3 files of Michael Bublé, Madonna and Frank Sinatra of all the peers in the group. Then, it verifies that at least one hundred of them provide at least 1GB of Mp3 files of the selected singers. If this is the case, the **FreeMusic**.MS sends such advertisements to p, that tries to access the files. Since the formula is a strong formula, the **FreeMusic**.MS also notify the GM that the availability of the Mp3 files needs to be monitored. Thus, the GM will know about the violation of the requested formula and will promptly notify p, which will then leave the group and revoking the availability its resources.*

To be able to continuously verify the validity of strong formulae, the GM collaborates with the peers in the group to be updated on the peer group changes. Group members notify the GM to which they are associated about events occur that may affect the validity of strong formulae. The events handled by the GMs and the peers are the following:

Peer not found if a peer finds that another one is no longer reachable it communicates such condition to the associated GM. If the GM determines that such condition is permanent, it verifies the resources provided by such peer appear in any formulae of its strong formulae's list. Permanent in this context means the peer is unreachable for a significant time interval, where the length is defined during the peer group creation. If a given formula Φ is no longer valid, due to the missing peer, the GM notifies the peers who have negotiated their membership using Φ. Such peers are now free to leave the group or adjust to the new configuration.

Resource not found if a peer receives an error when trying to access a resource (e.g. an access failure message upon sending a legitimate request), it notifies the GM which will check if the resource was only temporary unavailable or if the peer is down; in the latter case, the dead peer will be canceled from the group. If the peer is only temporary unavailable, or it is up but not providing resources, its membership credential is revoked. This prevents from free-riders intentionally failing to provide resources they committed for.

New resource a peer willing to provide a new resource, notifies the associate GM and provides the identifier of the resource and the resource disclosure policy, making such resource available for subsequent negotiations. This case is relevant for the validity of universal disclosure policies (i.e., formulae with a $\forall$

operator), which require certain conditions to hold true for all the resources in the group.

Resource revoked a peer which is no longer providing a resource notifies the associated GM. The GM handles this event as the resource not found event.

Note that the GM, is a peer itself, besides handling the events triggered by other peers, it can directly trigger an event-message in case one of the events described above occur.

We now briefly discuss the issue of the robustness of our system. By *robustness*, we mean that it is not possible for a peer willing to join the group or a peer already in the group to incorrectly – possibly in a malicious way – report about the availability of a given resource, involved in the join negotiation. In order to argue in favor of the robustness of our system, we note that two kinds of controls are enforced upon the resources involved in a join negotiation. The former is weak verification, detailed above in this Section. According to it, if an outer peer (willing to join the group) declares a resource in the negotiation process, the effective availability of such resource will be checked by the GM upon entrance of the peer in the group, in case of successful negotiation.

The latter is strong verification. In this case, the resource availability is periodically checked. Thus, even in the case that the resource becomes unavailable once the peer has joined the group, such event is eventually detected by the GM.

4.5 The Negotiation-based Membership Service Architecture

In order to support the join-based protocol discussed in the paper, we have extended JXTA along several dimensions. First, we have deployed the JXTA negotiation service to implement a new version of Membership Service, referred to as *TrustMembership Service*. Second, we have introduced a *Group Membership Infrastructure*, which includes group members and group agents, to monitor and control in a collaborative manner the state of the peer group.

The JXTA membership service assigns an identity to a peer within a peer group by authenticating the peer through a negotiation process. The advertisement of peer groups that adopt trust negotiations to perform the join process, must contain the TrustMembership Service advertisement. The classes implementing the new Membership Service are described in Appendix B.10.

The JXTA negotiation service has been implemented in Java. The policies driving the group-based negotiation process, as well as credentials and the resource advertisements used to prove that policies can be satisfied are stored in a MySQL database. Since the verification processes described in Section 4.4.2 requires the instantiation of the Discovery Service whenever the local advertisements are to be retrieved – and the instantiation of the Discovery Service is memory and time expensive – we store all the advertisements that are in a peer′ local cache into the previously mentioned MySQL database. This is done by the GM that receives from the peers the advertisements in their local caches. This simplifies the discovery process, which is done by means of SQL queries. Peers' local policies and resources descriptions are stored in peers' local databases.

The system supports four operations:

1. `StartNegotiation`,
2. `PolicyEvaluation`,
3. `WeakVerification` and
4. `StrongVerification`.

`StartNegotiation` allows one to set the negotiation parameters such as the counterpart's peer identifier and to initiate the negotiation process with this peer. Strategies to conduct the negotiation can be selected, according to the peer's preferred mode [13]. `PolicyEvaluation` has a two-fold functionality. By invoking `PolicyEvaluation` a peer can send the policies that it wants to be satisfied by the counterpart and, in turn, it can verify whether it is able to satisfy the received policies.

As mentioned in Section 4.4.1, we keep trace of policies exchanged between the negotiators by using a tree structure. `WeakVerification` allows a peer to select the weak formulae it has to satisfy. It sends also the resource advertisements/credentials required by the counterpart. The resource advertisements and credentials to be sent to the counterpart are retrieved from the peer's local database. `Weak Verification` also enables verification of the counterpart's weak formulae, on the basis of the received resource advertisements and credentials.

Finally, `StrongVerification` provides the same functions of `WeakVerification` but it enable also the continuous monitoring of the resources required for verifying the strong formula.

The second important addition to JXTA, is the support of the *Group Management Infrastructure*, composed by a set of *Group Managers* and of *Group Agents*. The GM is a peer in charge of controlling the state of the group and of the resources provided and requested within strong formulae.

The Group Agents (GAs, for short) are agents residing in each peer and communicating with the associated GM by means of the event-based messages described below. The Group Agent component is automatically initialized for a peer using the TrustMembershipService, upon a successful join operation.

As mentioned, the resources' availability is monitored through various kinds of events that model the possible behaviors of the peers. Such events trigger the GAs to send corresponding messages to the GM, which performs suitable actions, as detailed below.

In case a peer identifies a missing peer or resource, it calls a method of its GA object. The GA sends a XML document to the assigned GM via `CommunicationLayer` which may be a TCP/IP Socket, a SSL Socket or a JXTA Socket. The XML document specifies the type of event generated and the relative information. The GM processes the events depending on the type. On a *Resource not found* event, as described before, the GM searches in the GM database whether the resource missing on peer p was requested by some other peers as prerequisite to the membership of p itself. In the affirmative case, the GM revokes the membership of p. Beside that, the GM checks whether the resource revoked r is required by some another peer p' within a quantified formula. If this is the case, the GM notifies p' about the violation of the formula and waits for the decision of p' about remaining or leaving the group. If the message is of the type *Resource Revoked*, the same procedure is executed for the peer whose GA is sending the event message. Upon receiving a message representing a *Peer not found* event, the GM searches in the GM database the resources provided by the peer p. For every such resource, the procedure for *Resource not Found* event is executed.

4.6 Experimental results

We carried out a set of experiments to evaluate the performance of our approach. In particular, we have evaluated the execution time required (1) to carry on a negotiation, (2) to detect free riders peers during the join, and (3) to detect changes in a peer group using our event-based mechanism.

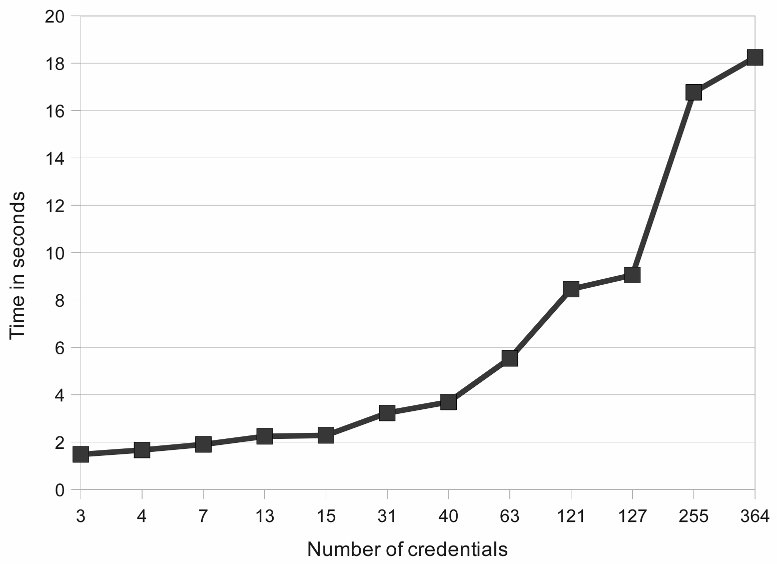


Figure 4.4: Execution time of a group-based negotiation by number of credentials involved

We conducted the first set of experiments, that is, the performance evaluation, by measuring the negotiation execution time with respect to two parameters: the negotiation length (that is, the number of credentials exchanged during a negotiation) and the negotiation tree height (which represents the number of negotiation

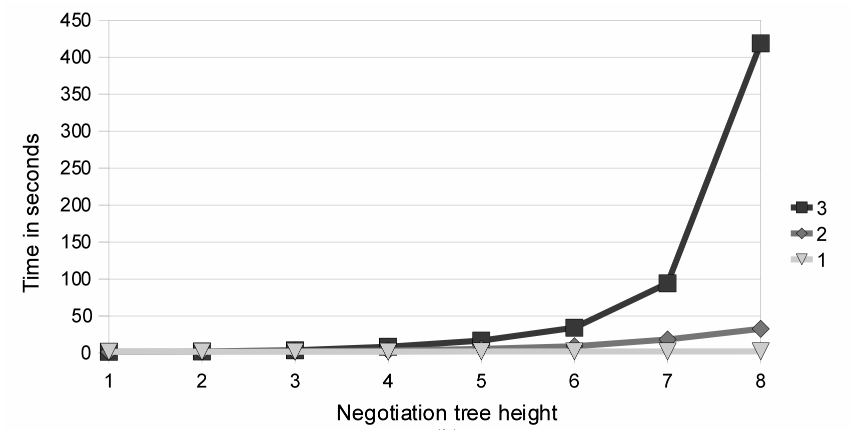


Figure 4.5: Execution time of a group-based negotiation, varying the number of rounds and the branching of the tree (1,2,3)

rounds).The results are shown in Figures 4.4 and Figure 4.5, respectively.

In both cases, we tested the worst case scenarios, in which the exchanged credentials create a complete tree, with fixed branching factors. A negotiation following this pattern results in a large number of credentials/policies to be evaluated.

Figure 4.4 shows the negotiation execution time while varying the number of credentials exchanged. Intuitively, the execution time grows with the number of credentials being exchanged. For realistic negotiations, which involve up to 40 credentials, the protocol adds a very low (around 4 sec.) overhead. Figure 4.5 reports the negotiation execution time while varying the number of negotiation rounds. To evaluate the impact on the number of negotiation rounds, we consider three test cases; we vary the number of credentials requested by each disclosure policy from 1 to 3. As in the previous case, the negotiation time significantly increases when several hundreds of credentials are involved in the negotiation, but it is still efficient when the number of credentials involved is reasonable. In the case of negotiation involving disclosure policies asking 3 credentials/resources each, the negotiation time begins to be significant with a negotiation of six rounds, i.e., when there are 127 credentials involved.

We have also compared our approach with the approach of Ramakrishna et al. [86], since it is the approach closest to ours. Our approach significantly outperforms the performance of Ramakrishna's protocols. For a negotiation of four rounds, the approach from Ramakrishna and colleagues takes about 8 seconds while our approach takes 1.734 seconds. We also notice that, in Ramakrishna's approach, most of the time is required to perform I/O operations. Figure 4.6 shows in detail the execution time of the negotiation according to the policy settings of [86], comparing their results with ours.

The second set of experiments, e.g. detecting possible free riders, describes a key feature characterizing our proposal. Free riders can be detected both during

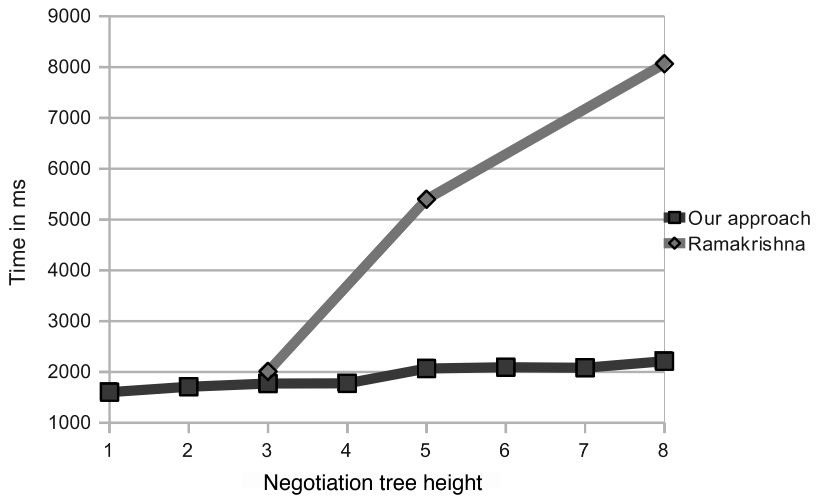


Figure 4.6: Comparison of performances of our negotiation approach with Ramakrishna's [86] one.

and after the negotiation process.

During the negotiation, a peer may detect malicious behavior while verifying some resources for an incoming member. Malicious behavior can also be detected after the negotiation, when multiple events related to a misbehaving or an unavailable peer are notified to a GM. The lack of previously available resources may impact the validity of strong formulae and, consequently, cause some peers to leave, altering the peer group configuration. Notice that for this experiment, resources availability is checked by comparing resources advertisements with requests.

To perform this test case, we consider the scenario of a peer group G and a peer p who wants to join G. We assume that p is asked to provide a resource r to join the group. Peers within G have a probabilistic behavior; that is, they try to randomly access with probability (or *interaction frequency*) α a resource specified in one of its strong formulae. The interval between each probabilistic access is defined by the *Time* parameter.

To evaluate the detection of free riders, we consider the case in which a free rider is detected during the policy evaluation phase and the case in which it is detected during the formulae verification phase. Then, we evaluate the promptness of event-based verification process with respect to three parameters: the cardinality of the peer group, the frequency time and the probability α.

To evaluate the impact of the group cardinality, we measure the time to detect an event "Resource Not Found" by keeping constant α while varying the cardinality of the group G from 10 to 200. This test also evaluates the scalability of our approach with respect to the number of peers. On the contrary, to evaluate the impact of the parameter α, we measure the time to detect "Resource Not Found" event keeping constant the cardinality of the group G while varying the value of

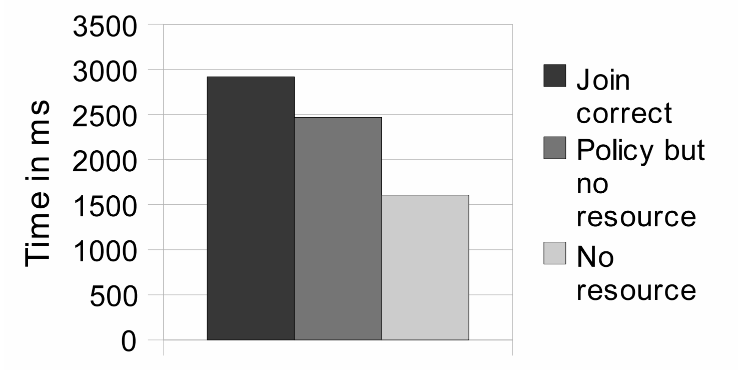


Figure 4.7: Execution time for detecting free riders

$\alpha \in \{0.025, 0.05, 0.075, 0.1\}$. The assumptions about frequency of interaction and cardinality of the group apply well in case of peer groups similar to the one described in Section 4.2.

As reported in Figure 4.7, free riders are always detected within a reasonable time period. In fact, in the worst case, that is a free rider is detected during the formulae verification phase, the detection time is around 2.5 seconds (the central column of Figure 4.7). Otherwise, as shown by the right column of Figure 4.7, during the policy evaluation phase a free rider is detected in around 1.5 seconds. In both cases the required time is less then the time required for a successful join.

Figure 4.8 shows the time required by the group to detect the event "Resource Not Found" by increasing the cardinality of the peer group G. For this case, we modeled the behavior of the peers using the frequency of the requests of a resource by means of different time intervals (denoted as *time*) and probability values (denoted as α). The time to detect a change in resource availability decreases for increasing values in the peer group cardinality regardless the value of probability α. In fact, in this case the number of interactions between the peers is very high and thus the time to detect that a resource is no longer available is shorter. However, the graph shows that also for peer groups with low cardinality (20-30) but with a high values of the interaction probability (0.1 and 0.025) the detection time dramatically decreases.

Figure 4.9 shows how α, denoting the peers' interaction frequency, influences the detection of the event "Resource Not Found" in a group. For this experiment, we evaluate different values of α for groups of different cardinality, ranging from groups of 10 peers to groups of 200 peers.

Interestingly, the interaction frequency of the peers α has a higher impact on the detection times than the cardinality of the group. In fact, for a value of α equal to 0.075 the detection time decreases regardless the group cardinality being considered. However, we notice that a peer group with a low interaction frequency has still a good performance as long as it as more than 50 operating members.

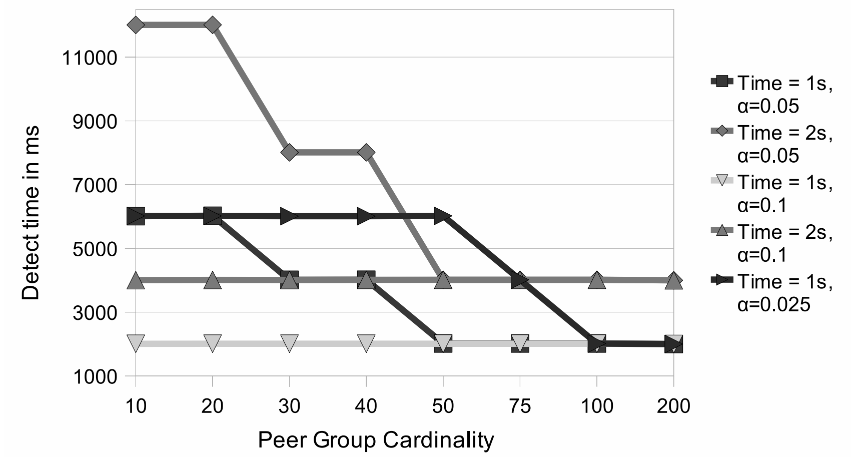


Figure 4.8: Time interval required to identify changes in the state of the group

From both set of tests, we notice that there is a minimum time required to detect the event, corresponding to 2000 ms. This time is the minimum time for a peer in the group to complete two tasks:

1. detect the event (notice that this takes some time as the peer will wait for some milliseconds for a reply before registering the lack of response as a "Resource Not Found" event) and

2. communicate the detected event to the GM.

From the experimental results obtained, we can conclude that the effectiveness of our approach depends heavily on the interactions between the peers. The higher is the frequency of interactions the shorter is the time to detect changes happening in peer groups. Notice that when a missing resource is detected by any of the peers, it is notified to a GM, which consequently updates the peer group data and informs the peers who relied on the resource for any of their strong formulae. Therefore, our approach proves that even if we employ peers with special roles, the detection of free riders is truly distributed, and relies on the group nature. As shown, this is an efficient and effective approach to protect peer groups from misbehaving peers, and is thus particularly suitable for P2P networks.

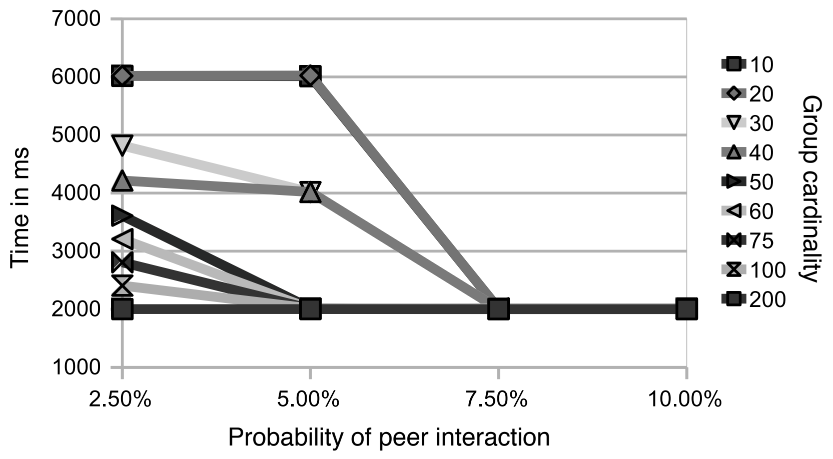


Figure 4.9: Impact of the peer group interaction in detecting changes of state in groups with constant cardinality

```
Input: $Party$ and $counterpart$ identifier; $\Phi\_List_{Party}$ and $\Phi\_List_{Counterpart}$
       $Form\_Seq$ the sequence of agreed formulae
Output: Verification completed
begin
    for $\Phi(t_1, ..., t_n)_i \in Form\_Seq$ do
        if $\Phi(t_1, ..., t_n)_i \in \Phi\_List_{Party}$ then
            $\{Cred\_Set, Adv\_Set\}$= RetrieveInfo($\Phi(t_1, ..., t_n)_i$);
            if $Cred\_Set == null \vee Adv\_Set == null$ then
                $Join\_failed$ = true;
                break;
            else
                Send($Adv\_Set$,$Cred\_Set$, $Counterpart$);
        else
            Receive($Adv\_Set$,$Cred\_Set$);
            for $Cred_i \in Cred\_Set$ do
                $val$ = VerifyValidity($Cred_i$);
                if $val == false$ then
                    $Join\_failed$ = true;
                    Notify($Join\_failed$,$Counterpart$);
                    break;
            for $Adv_i \in Adv\_Set$ do
                $result$ = AccessResource($Adv_i$);
                if $\Phi(t_1, ..., t_n)_i$ is a strong formula then
                    ToEnsure($Adv_i.R_i$, $Counterpart$);
                if $result == false$ then
                    $Join\_failed$ = true;
                    Notify($Join\_failed$,$Counterpart$);
                    break;
```

Algorithm 3: Formulae Verification

Chapter 5

Mobile multi-session Trust Negotiations

5.1 Introduction

As shown in Chapter 2, trust negotiation researches has mostly focused on the assurance of privacy and confidentiality with the goal of guaranteeing that no actual information about a negotiator's properties is disclosed to the counter-part [91, 93, 12].

Typically, these approaches rely on strong cryptographic assumptions, and are seldom applicable in many real-world scenarios, where properties, stated in digital credentials, actually need to be disclosed in clear and not only proved to be true. For example, just proving the possession of a valid credit card is not sufficient to complete a transaction, and actual account information is to be supplied in order to enable charging the amount spent. Additionally, protocols that rely on oblivious credentials or anonymous credentials do not allow the negotiating peers to follow the progress of the negotiation, since information regarding policies satisfaction is hidden for confidentiality purposes [64, 50].

In this chapter we introduce a novel approach to trust negotiations that offers a general solution to those issues by developing major extensions to previous approaches by us and others [13, 127, 105].

The core of our approach is a trust negotiation protocol supported by the Trust-$\mathcal{X}$ system, described in Chapter 3 and Appendix B.

The main innovative feature of the proposed framework, referred to as *multi-session* trust negotiation, is to support crash recovery and the possibility of completing the negotiation according to multiple sessions. To support the execution of multi-session negotiations, Trust-$\mathcal{X}$ extends the conventional steps characterizing a trust negotiation protocols. Savepoints are employed to save the negotiation state, and validity checks concerning events which may happen during the negotiation suspension and could possibly invalidate the negotiation steps executed before the suspension.

Examples of those events include credential revocation or expiration, or modification of disclosure policies by one of the peers.

To achieve such objective the presented framework exploits a highly configurable savepoint technique. Savepoints are copies of the current state of the negotiation, saved on stable storage. The frequency of the savepoints can be defined by the peers at the beginning of the negotiation.

Additionally, the possibility to safely suspend and resume an ongoing trust negotiation allow the peers to *migrate* such negotiation among different authorized peers, which means a peers is able to suspend an ongoing negotiation and resume it with a peer different from the one with which the negotiation started. We refer to such feature as *mobile trust negotiation*.

To support the secure transfer of negotiations, we defined an authentication protocol, based on a secret-splitting scheme combined with a zero-knowledge proof protocol to verify the identity of the peer recovering the negotiation and to assure the validity of the exchanged data (for further details, refer to Appendix A.2 and Appendix A.3).

5.2 Credentials' similarity.

Before presenting our new protocols, we begin introducing some basic concepts.

Credential types syntactically structure the information conveyed by the corresponding credentials, but they do not specify anything about the semantics of the attributes. Hence, it is impossible to automatically determine whether attributes belonging to different credentials are semantically related (e.g., they express the same information, one is the specialization of the other, and so on). This information is crucial during a multi-session negotiation, because it allows one to select among possible different similar credential attributes the one to be disclosed, thus enhancing flexibility and privacy. To deal with this issue, we borrow some ideas from work on ontologies. Ontologies represent an essential tool for sharing among distributed users and applications by providing a common understanding of a domain of interest. An ontology consists of a set of concepts together with relationships defined among these concepts. In this work, we rely on an OWL [119] specification. OWL is a standard language for ontology specification. Credentials similarity relies on mapping among concepts, based on the use of ontologies with attribute names or categorical values.

In what follows we assume the credentials involved in the trust negotiation process to be primarily categorizable on their semantics. Examples of credentials' categories are "ID", "Work life", "Financial". Such categories are used to classify the credentials, so to simplify the identification of the credentials eligible substitutes of an expired or revoked one. Beside a set of predefined categories, we support a "Misc" category that contains the credentials that cannot be be associated with existing categories.

Each category is associated with a predefined OWL model defining the core attributes a credential is supposed to contain in order to be classified with the category. The OWL model is obtained by the credentials themselves during a preprocessing phase. When Trust-$\mathcal{X}$ is initialized, and every time a new credential is inserted in the $\mathcal{X}$-Profile, the uncategorized credentials are processed.

5.2.1 OWL model extraction

Preprocessing the credentials contained in the $\mathcal{X}$-Profile consists of extracting an OWL model from the XML document encoding the credential. This extraction is automatically performed by sequentially applying a set of XSLT schema [18, 121] to the XML representing the credential. Such transformation is feasible since Trust-$\mathcal{X}$ credentials are strongly *data oriented* XML documents. *Data oriented* XML documents use XML as interchange format, to facilitate transmission, and their structure is decided upon the content, rather than their readability, as for the document-oriented XML documents. Such characteristic greatly improves the accuracy of the obtained OWL model, preserving the content of the credential with respect to the representation of the concepts corresponding to attributes, and the relationships among them.

The process for extracting the OWL model consists of two steps. In the first step, an XSLT is applied transformation to extract a XSD schema [120] from the XML document encoding the credential. In the second step, the XSD is transformed into a OWL model by applying another XSLT transformation.

We remark that other tools and approaches for extracting OWL models from a set of structured documents exist in literature, such as GRDDL [122, 123].

5.2.2 Credential categorization

After the extraction of the credential's OWL model from the XML document, the model is processed by a matching algorithm which evaluates the credential's OWL model against different metrics. Such metrics have been chosen to both identify syntactic and semantic similarities among the credentials' OWL model and the categories' ones. The final result of our matching algorithm is the normalized and weighted composition of the results of the comparison of the two models using different algorithms that rely on various metrics.

Similarity metrics

We used different metrics in order to identify syntactic and semantic similarities among credentials. These metrics are implemented by algorithms provided by the Falcon AO [54] system and have been used in a number of different domains [76]. In our system, we use the following algorithms:

V-DOC [85]. V-DOC takes a linguistic approach to ontology matching. Basically, as a collection of weighted words, the virtual document of a domain entity (e.g., a class or a property) in an ontology contains not only the local descriptions, but also the neighboring information to reflect the intended meaning of the entity. Document similarity can be calculated via traditional vector space techniques, and further be used in certain similarity-based approaches to ontology matching.

GMO [52]. GMO is an interactive structural matcher that uses RDF bipartite graphs to represent ontologies and computes structural similarities between domain entities and between statements (triples) in ontologies by recursively propagating similarities in the bipartite graphs. GMO takes a set of external alignments as input, which are typically found previously by other matchers, and incrementally generates extra alignments as output. The performance of GMO improves as the precision of external alignments increases.

The matching algorithm computes the matching between two OWL models received (o_i and ω_j, respectively) in input using the described algorithms. The returned similarity measure, computed according to expression (5.1) below, is the result of a weighted sum of the similarity measures computed by algorithms V-DOC and GMO.

$$\mu(o_i, \omega_j) = w_{V-DOC} V - DOC(o_i, \omega_j) + w_{GMO} GMO(o_i, \omega_j) \tag{5.1}$$

where ω_j is the OWL model representing the j category. Therefore $\omega_j \in \Omega$, with $i < j < l$ where l is the number of categories in which we want to classify the credentials of the $\mathcal{X}$-Profile. Each w_t, with $t \in \{V\text{-}DOC, GMO\}$ is a weight associated to the corresponding algorithm. Such weights are used, depending on the context, to emphasize the importance of the metrics they apply to and normalize the different results of the metrics on a common scale.

5.2.3 Application of the matching algorithm

Once the matching algorithm is applied to a credential c, c is associated with the most similar category, referred to as *primary category*, and with a set of other categories, referred to as the *set of secondary categories*. Given a credential c, its primary category is denoted as $Pri(c)$, while the set of secondary categories as $Sec(c)$. The secondary categories are the categories that with respect to c have a similarity value lower than the similarity value of the primary category and higher than a threshold value β. To reduce the number of matching categories, we introduce a threshold value α, with $\alpha > \beta$. The thresholds α and β are defined in the system but could be refined by the end-user to better suit the categories and the credentials contained in its $\mathcal{X}$-Profile.

We formally define primary and secondary categories in what follows.

Definition 5.2.1 (Primary category). *Given a credential c, a set of categories $\Omega = \{\omega_1, \ldots, \omega_k\}$ and a matching algorithm μ, we define the* primary category *for c, denoted by $Pri(c)$, as $\omega_i \in \Omega$ where i is $max(\mu(\omega_i, c) > \alpha)$.*

Credential	**Primary**	**Secondary**
Passport	*Personal Information*	*Working Life*
IdentityCard	*Personal Information*	*Working Life* *Financial*
Work ID	*Working Life*	*Personal Information*
CreditCard	*Financial*	*Working Life* *Personal Information*
BankAccount	*Financial*	*Personal Information*

Table 5.1: Example of credentials' categories

Definition 5.2.2 (Secondary category)**.** *Given a credential c, a set of categories $\Omega = \{\omega_1, \ldots, \omega_k\}$ and a matching algorithm μ, we define the set of the* secondary categories *of c, denoted by $Sec(c)$, as $\{\omega_1, \omega_2, \ldots, \omega_s\}$ where $\mu(\omega_i, c) > \beta$.*

Note that some credentials may not meet the requirements of any category. In such case, the credential c is associated with a special category called *Misc*.

Example 5.2.1. *Consider the following categories:*

1. Personal Information*: the model of this category has attributes* `name`*,* `surname`*,* `address`*,* `date of birth`*, and SSN;*
2. Work Life*: the model of this category has attributes* `name`*,* `surname`*, and* `employee number`*;*
3. Financial*: the model of this category has attributes* `name`*,* `surname`*,* `bank name`*,* `bank address`*, and* `bank account`*.*

A credential representing the passport of a user, called Passport *and with attributes* `name`*,* `surname`*,* `address`*,* `date of birth`*, has* Personal Information *as primary category and the other two categories as secondary ones.*

Given a credential c we define $\mathcal{S}_c$, the set of *similar credentials* of c, based on the categories it belongs to. More formally:

Definition 5.2.3 (Similar credentials)**.** *Let c be a credential and* Pri(c) *be its primary category as by Definition 5.2.1, and $Sec(c)$ be the set of its secondary categories, as by Definition 5.2.2. The set of similar credentials $\mathcal{S}_c$ for c is defined as:*

$$\mathcal{S}_c = \{c_i | Pri(c_i) = Pri(c) \vee Pri(c_i) \in Sec(c) \vee c_i \in \text{Misc}\}.$$

Example 5.2.2. *Consider again the categories defined in Example 5.2.1. Let the following credentials be classified according to Table 5.1.*

According to Definition 5.2.3, the credentials similar to the credential Passport *are $\mathcal{S}_{Passport} = \{$IdentityCard, Work ID$\}$. The credentials* CreditCard *and* BankAccount *are not similar because they are classified in categories different from the ones of* Passport.

5.2.4 Credential substitution

Based on the notion of credential similarity, we are now able to identify a substitute for a credential c if c has been revoked or has expired.

Definition 5.2.4 (Substitute credential). *Given a credential c and a policy* $pol = C_i \leftarrow \phi(t_1, \ldots, t_n)$*, the* substitute *for c with respect to p is defined as the credential, denoted by* s_c^{pol}*, such that:*

1. $s_c^{pol} \in \mathcal{S}_c$ *and*

2. *if* $\phi(c, c_1, \ldots, c_n)$ *holds true, then* $\phi(s_c^{pol}, c_1, \ldots, c_n)$ *holds true.*

Example 5.2.3. *Let* $\phi(Passport, DrivingLicence) \equiv Passport(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"}) \wedge DrivingLicence(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"}, ExpiringYear > 2010)$. *If the credential Passport is expired, we search within the primary category it belongs to (let it be "PersonalInformation") a substitute which will be able to satisfy the above disclosure policy. For example, assume that the credential IdentityCard contains similar information. According to Definition 5.2.4, we substitute* $Passport(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"})$ *with* $IdentityCard(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"})$ *obtaining:*

$$\phi(Passport, DrivingLicence) \equiv IdentityCard(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"}) \wedge \\ DrivinLicence(Name = \text{"}John\text{"}, Surname = \text{"}Doe\text{"}, ExpiringYear > 2010)$$

Example 5.2.4. *Consider the resource condition* $t = IdCard(YearofBirth = 1980)$. *Credential* C_1 *=IdCard(Pennsylvania, Mary Mork, 322 State Road, Altoona, 1/11/1980) satisfies t. An equivalent credential is IdCard(Indiana, Mary Mork, 1980). In case of a resource condition of the form* $t2 = X(YearofBirth = 1980)$ *a credential with the following structure DriverLicense(Issuer, Name, Address, YearofBirth, ExpirationDate) is equivalent to* C_1 *provided that YearofBirth be equal to 1980.*

5.3 Negotiation tree switching

In this section we provide the core contribution of our approach to multi-session. We begin with presenting a high-level discussion of the approach, followed by a detailed presentation of the new Trust-$\mathcal{X}$ algorithms.

5.3.1 Overview of the approach

In order to support long lasting negotiations, we introduce two major features to trust negotiation protocols. First, we support multi-session negotiations, that is we allow negotiations to be conducted within multiple separate sessions. We depart from the assumption of atomic trust negotiations in order to make the negotiation also suitable for peers with heterogenous capabilities. In the multi-session protocol, we do not require both parties to maintain an up to date copy of the negotiation state at the time of suspension. Relaxing such assumption does not imply going back to a client-server architecture. Rather, parties are still peers, and therefore able to control the negotiation process; however the task of storing the negotiation data at suspension time can be assigned to one of the two parties. This approach makes trust negotiation protocols suitable for a large variety of environments, in that it may often be the case that one of the negotiating peers is connected via a device with limited connection power or small memory. A second important extension is to allow negotiations to be completed by multiple peers. Essentially, we now allow the negotiations between two peers, say P_1 and P_2, to be suspended and then resumed by different peers. For example, P_2 can be replaced by P_3, provided that the replaced –or *delegated* – peer (e.g, P_3) has the ability to complete the previously started negotiation. We do not expect peers to be replica one of another. Hence, some interesting security challenges arise, related to the trustworthiness of the peers and the compatibility with the credentials and policies of their predecessors. To implement these extensions, we introduce a protocol allowing peers to:

- suspend and resume negotiations with a peer different from the original negotiator;
- share with a different peer in a privacy-preserving way the negotiation tree so far constructed.

A brief description of the main steps of the multi-session negotiation (MS, for short) protocol is reported in what follows, and summarized in Figure 5.1.

We begin with focusing on the case of a negotiation that was suspended during the policy evaluation phase. We assume, without loss of generality, that P_2 is keeping track of the advance of the negotiation (since, e.g., P_1 is a mobile, low-end device).

1. During the execution of the policy exchange phase, P_1 and P_2 exchange credentials as soon as their corresponding nodes in NT reach the DELIV state. Since it may be not the case that all DELIV nodes are contained in a valid view (and hence they should not be actually exchanged), the corresponding credentials are sent in an encrypted form, thus preventing information disclosure (see Section 5.3.2).

2. P_1 and P_2 suspend the negotiation. P_2 processes the NT tree in order to hand it to another peer, which will resume the negotiation, by pruning the delivery nodes from NT.

3. Subsequently, for every term in the pruned negotiation tree, P_2 computes the corresponding commitments (see Section 5.3.2).

4. P_2 linearizes the pruned tree, thus obtaining another tree S. P_2 applies the secret sharing scheme to S, obtaining n, $n \geq 3$, shares; let $S_1, S_2, \ldots, S_n$ be such shares. P_2 sends S_1 to P_1, keeps S_2 for itself, and distributes $S_3, ..., S_n$ to a

subset $\mathcal{P}'$ of peers in $\mathcal{P}$, where $\mathcal{P}$ corresponds to the set of peers known to P_1 as potential delegates ($\mathcal{P}$ and $\mathcal{P}'$ may coincide). In addition to S_2, P_2 sends the keys for deciphering the encrypted versions of the credentials corresponding to DELIV nodes (already exchanged with P_1).

5. Suppose that P_1 has contacted and authenticated peer $P_3 \in \mathcal{P}'$. Upon receiving S_3 from P_1, P_3 reconstructs the node-by-node committed negotiation tree (see Section 5.3.3). Now P_3 has to build its version of the negotiation tree starting from the committed tree and from its available credentials. For every node in the tree, P_3 (by using a non-interactive zero-knowledge proof of knowledge) checks whether its credentials match the committed terms (see Section 5.3.4).

6. Having built its version NT'_{open} of the negotiation tree, P_3 sends it to P_1 and they can restart the previously suspended negotiation. In the case of successful termination and in case that the valid view contains credentials sent in the first phase of the negotiation, P_3 sends the corresponding keys to P_1.

Notice that the case of interruptions is essentially an extension of the above described protocol. Unlike suspensions that are voluntarily decided by the negotiating peers, interruptions are unforeseen by the peers and are due to external events, like network and system crashes. To support recovery from interruptions, the steps (2) (3) and (4) above are executed periodically, asynchronously by either peer, with the only difference that at step (2) the negotiation is *not* suspended. Each peer saves a version of the tree and generates a secret for the sharing (step 3). The secret shares are then distributed to $\mathcal{P}'$, that will not use them unless necessary. If neither an interruption nor a suspension occur for a few negotiation rounds, the peers will periodically update their committed versions of the tree (this can be done incrementally for the new nodes to avoid unnecessary overhead) and create new secrets shares. The receiving peers in $\mathcal{P}'$ will simply replace the old secrets with the new ones. If an interruption occurs, one of the two peers, say P_1 will simply stop sending and receiving messages. When P_1 recovers, the most recent shares sent by P_1 will be used to reconstruct the intermediate state of the negotiation.

5.3.2 Negotiation, Suspension, and Nodes Commitment

As outlined in Step (1) above, peers P_1 and P_2 exchange credentials as soon as their corresponding nodes in the negotiation tree NT enter the $DELIV$ state. This approach is not conventional to trust negotiation strategies that usually postpone credentials disclosure after a successful path is found. The purpose of anticipating the disclosure is twofold: first, it provides partial assurance to the negotiating parties regarding their opponent's commitment to negotiating; second, it reduces the cost of the subsequent phases and removes the already processed nodes from the tree.

Notice that, in order to prevent information leakage, what is actually exchanged is an encrypted version of the credentials, computed by P_2 with a (previously agreed

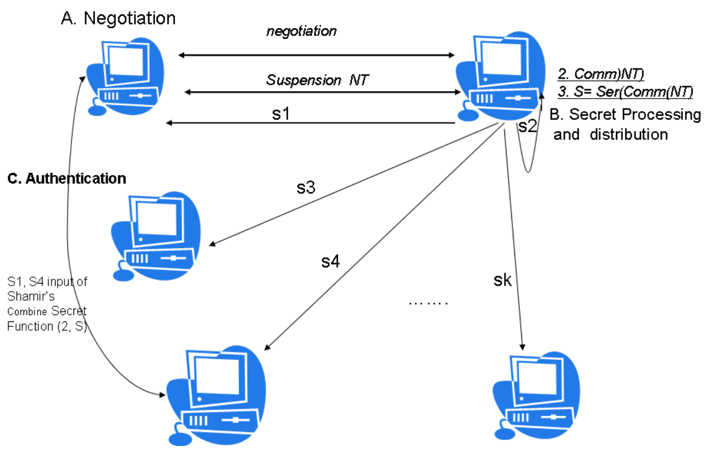


Figure 5.1: The three macro phases of a long lasting negotiation: 1. Negotiation 2. Suspension and preprocessing 3. Tree sharing upon resume.

upon) symmetric encryption scheme Enc (e.g. AES). Each credential is encrypted with a different key; all these keys are unilaterally chosen by P_2. Upon negotiation suspension (Step (2) above), P_2 prunes from the negotiation tree NT all the $DELIV$ nodes, obtaining the tree NT_{open}, which contains open nodes only.

Subsequently, at Step (3), peer P_2 computes for each node in NT_{open} the corresponding commitments using the Damgård-Fujisaki commitment scheme (see Appendix A.3).

Given a node in NT_{open}, suppose that the associated resource condition has the form

$$cred[att_1, pred_1, val_1, conn_1, \ldots conn_u, att_u, pred_u, val_u]$$

where $conn_i \in \{\vee, \wedge\}$. Then, P_2 computes[1]:

$$\mathsf{commit}(cred)[\mathsf{commit}(att_1)\ pred_1\ \mathsf{commit}(val_1)$$
$$conn_1 \ldots conn_{u-1}\mathsf{commit}(att_u)\ pred_u\ \mathsf{commit}(val_u)].$$

We denote with $\mathsf{commit}(NT_{open})$ the negotiation tree in which every open node has been committed.

5.3.3 Tree Splitting and Sharing

If the negotiation is suspended during the policy evaluation phase, the tree is used for checkpointing. An encrypted version of the tree is generated. Subsequently, P_2 serializes it for improved efficiency. We denote the serialized version of the (pruned and committed) negotiation tree as $S = \mathsf{ser}(\mathsf{commit}(NT_{open}))$. Once serialized, the

[1]We omit to explicitly indicate the random value r. It is implicit that it changes for every execution of the commitment algorithm.

tree, being now encoded by a (large) string S, is used as input for the Shamir secret sharing scheme: P_2 generates n secret shares, denoted by S_1, S_2, and S_n from S. P_2 then sets $k = 2$ and distributes the share to $P_3, P_4, \ldots, P_n$, all belonging to $\mathcal{P}$ and being trusted by P_2. P_2 will, by default, keep one share for itself, and gives S_1 to P_1. When P_1 recovers and/or restarts, it can select among the peers owning the other share the one to resume the negotiation with. Notice that the notion of trust here is loosely used; the peers with whom P_2 shares the secret are not peers which can immediately access the negotiation status in all its details, but peers who are trusted enough to adhere to the negotiation protocol and complete the suspended process. P_1 therefore will continue the negotiation with any peer among $P_2, \ldots, P_n \in \mathcal{P}$. Suppose that P_3 is selected. Then, the partial secrets of P_1 and P_3 must be combined. Shares are then pooled by P_3. The two shares provide two distinct points $(x; y) = (i; S_i)$ making it possible to compute the coefficients of $f(x)$ by Lagrange interpolation. The secret is recovered by noting that $f(0) = a0 = S$. For increased security, k can be set to three, so that the authentication of peer P_3 is 'witnessed' by a third party.

5.3.4 Tree Recovery

After peer P_3 has recovered the tree $\mathsf{Commit}(NT_{open})$, it builds its version of the negotiation tree starting from $\mathsf{Commit}(NT_{open})$ and its own credentials. P_3 builds its tree by executing knowledge proofs on the commitments. For every committed node in $\mathsf{commit}(NT_{open})$ and for every credential $cred'[att'\ op\ val']$, P_3 performs the following non-interactive zero knowledge proofs. Recall that a committed node has the form

$$cc = \mathsf{commit}(cred)[\mathsf{commit}(att_1)pred_1\mathsf{commit}(val_1)$$
$$conn_1 \ldots conn_{u-1}\mathsf{commit}(att_u)pred_u\mathsf{commit}(val_u)]$$

We denote with c_{P_3} the credential set owned by P_3. Algorithm 4, reported in appendix, checks whether, for every committed term cc there is a credential in C_{P_3} with matching credential and attributes names that contains values satisfying the term predicates.

If every zero-knowledge proof succeeds, P_3 has constructed its own negotiation tree NT'_{open} upon which the negotiation with P_1 can be restarted.

As a final step (Step (6) in the above outline), P_3 sends the novel negotiation tree to P_1 and they resume the negotiation. In the case in which the trust negotiation succeeds and some encrypted credentials $\mathsf{Enc}(Cred_i, k_i), \ldots, \mathsf{Enc}(Cred_j, k_j)$ (exchanged during the negotiation between P_1 and P_2) belong to the valid view, P_3 sends the corresponding secret keys $k_i, \ldots, k_j$ to P_1, which finally deciphers the encrypted credentials and terminates the negotiation.

```
Data: commit($NT_{open}$), $C_{P_3}$
Result: $NT'_{open}$
$NT'_{open} = \emptyset$;
for $cc \in commit(NT_{open})$ do
    for $c' \in C_{P_3}$ do
        if $PK\{(m, r_1) : c = g^{cred} h^{r_1} \wedge cred = cred'\}$=true then
            for $att_i \in \{att_1, \ldots, att_u\}$ do
                if $PK\{(att_i, r_i) : c = g^{att_i} h^{r_i} \wedge att_i = att'_i\}$=true then
                    if $pred_i = \geq$ then
                        if $PK\{(val_i, r_i) : c = g^{val_i} h^{r_i} \wedge 0 \leq val_i \leq val'_i\}$=true then
                            $NT'_{open} := NT'_{open} \cup cred'[att'_i pred_i val'_i]$;
                    else if $pred_i = \leq$ then
                        if $PK\{(val_i, r_i) : c = g^{val_i} h^{r_i} \wedge 0 \leq val_i \leq val'_i\}$=false
                        then
                            $NT'_{open} := NT'_{open} \cup cred'[att'_i pred_i val'_i]$;
```

Algorithm 4: Checking algorithm

5.3.5 Sharing the Trust Sequence

A suspension may occur during the verification of the set of credentials identified as necessary and sufficient to complete the negotiation. This situation presents some security challenges, because of the sensitive contents exchanged during the credential evaluation phase. While in the case of policy evaluation phase, only uncertified statements are exchanged, here the exchange often involve confidential digitally signed documents. Releasing a credential to a third party for negotiation may not be desirable, even if this third party is considered trustworthy. Additionally, in order for the verification to be meaningful, both parties should be able to verify the credentials against previously exchanged policies. Clearly, this is a not trivial step, given that one of the two peers is, in case of resume, a different one, and may thus have different credentials and policies. Our approach to address these issues is based on carefully preprocessing the sequence generated by the valid view, and on a modified verification protocol. The MS protocol for credentials' exchange is the following:

1. P_1 and P_2 agree to suspend, after a sequence of terms $\mathcal{CS}eq$ is determined and partly exchanged[2]. P_2 encrypts the local credentials corresponding to terms in $\mathcal{CS}eq$ that have not been exchanged yet.

2. P_2 sends to P_1 a single session encryption key k_s (upon agreement on a sym-

[2]Note that the approach is the same even if none of the credentials has been exchanged yet.

metric encryption scheme Enc) to open the encrypted credentials required by the credential sequence $\mathcal{CS}eq$

3. P_2 generates an encrypted version of $\mathcal{CS}eq$ as follows: it leaves in clear the terms of P_1, while it replaces with encrypted credentials its own credentials. Then, it applies the secret splitting protocol on the modified $\mathcal{CS}eq$, obtaining shares $s_1, \ldots, s_k$.

4. P_2 sends to the possible delegates (i.e, $P_3, \ldots, P_k$) the generated secret shares $s_1, \ldots, s_k$, and leaves.

5. Assume that P_3 is selected. P_1 and P_3 can now reconstruct the sequence, and exchange the remaining credentials.

6. During the exchange, P_3 receives P_1's credentials $Cred_i, \ldots, Cred_k$ in order and in clear. Based on the corresponding terms stored in the $\mathcal{CS}eq$, P_3 is able to verify the validity of each credential and whether or not it satisfies the corresponding term. P_3 at its end sends sequentially P_2's encrypted credentials to P_1, which can verify them by using the key k_s obtained by P_2.

This protocol has a number of desirable features. First it allows P_1 and P_3 to complete the negotiation without requiring that P_2's credentials be disclosed to P_3. Second, it protects from possible changes of the sequence either by P_1 or P_3. Because the secret is split, P_1 cannot decipher with k_s P_2's credentials. Finally, it allows P_3 to verify the credentials from P_1 by using the terms in the sequence. For highly confidential credentials, it is also possible to replace the exchange of the credentials with their committed versions, so that only minimal information is disclosed.

An issue of the protocol is that it is not able to handle the case in which a credential within the sequence expires during the suspension and the subsequent resume is operated by a different peer. There are two different – and mutually exclusive – strategies to address this issue:

1. P_2 includes within the modified and encrypted $\mathcal{CS}eq$ up to d substitute credentials for each credential $Cred_i$ in $\mathcal{CS}eq$. Substitute credentials are identified according to Definition 5.2.3. Each encrypted credential $\mathsf{Enc}(Cred_i, k_{s)}$ will thus be associated with a timestamp defining the time validity period of credential $Cred_i$. After the recovery of the negotiation, the delegated peer P_3 checks whether the credential being sent to P_2 is still valid by verifying if the timestamp has not expired. If this is not the case, P_3 will select a substitute from the list.

2. The resuming peer finds a valid substitute credential. This task is performed by using the credential categories introduced in Section 5.2. The suspending peer P_1 adds to each encrypted credential $C = \mathsf{Enc}(Cred_i, k_s)$ in the modified $\mathcal{CS}eq$ the corresponding OWL model M_C. By exploiting M_C, P_3 identifies a substitute credential among its $\mathcal{X}$-Profile following the steps described in

Section 5.2. Obviously, such steps are performed if and only if the credential $Cred_i$ has expired.

The proposed strategies do not, however, address the problem of revoked credentials. P_1 is the only peer able to discover such case. If a credential has been revoked, P_1 notifies P_3, which will provide a valid substitute credential (using a credential provided by P_2 or using one of its own), if it exists.

5.4 Illustrative Example

In this section we illustrate the features of our approach by an example related to we discuss a simple yet typical scenario where the unique features introduced by our framework allow two negotiating peers to establish trust on the fly. We focus on a distributed mobile environment involving temporary and one-time credentials. In these domains, there is a strong need of secure and flexible approaches that allow mobile devices to access services provided by distributed systems in a efficient and reliable way. Mobile devices are becoming nowadays more and more common and they need to be able to seamlessly migrate from different physical servers belonging to the same service provider. Therefore we used such ideas to refine a framework which could provide the required features in the most general way.

Suppose that user Alice (A, from now on) would like to buy from BestBuy (B, from now on) a DRM-protected digital movie using a coupon allowing her to obtain a discount on the movie price. We refer to Figure 5.2 for a graphical representation of the example.

A connects to B from her PDA, and initiates a negotiation with one of the servers operating for B and identified as in charge of the negotiation for B's. Let this server be denoted by B_1, note that, in Figure 5.2 we use the double stroke to indicate the active server of B.

In order to provide the required movie, B_1 requests from A the coupon and the amount of e-cash required to buy the movie. This policy is encoded by a rule of the form: *Movie(Discount=coupon,title=sometitle)*$\leftarrow$ *Coupon(Issuer=BestBuy,Object=Movie)* $\wedge$ *Cash(amount=10)*.

Before sending her credentials, as required by B_1's policy, A requires some credentials from B_1. In particular, in order to release her coupon, A

requires a credential attesting that B_1 is a BestBuy server. Moreover, A requires a ticket which allows her to examine B's bank privacy policies. B_1 replies to the first policy by asking that A presents her BestBuyAccount in order to be authorized to access the credential showing that B_1 is a BestBuy. To disclose the temporary ticket which will let A access its bank policies, B_1 requires to identify a bank account to which to refer the temporary ticket itself. A is not registered at BestBuy so she asks to B_1 for a temporary suspension of the negotiation – Suspend(1) in Figure 5.2 –, since A is connected from a limited device that does not allow multiple transactions at the same time.

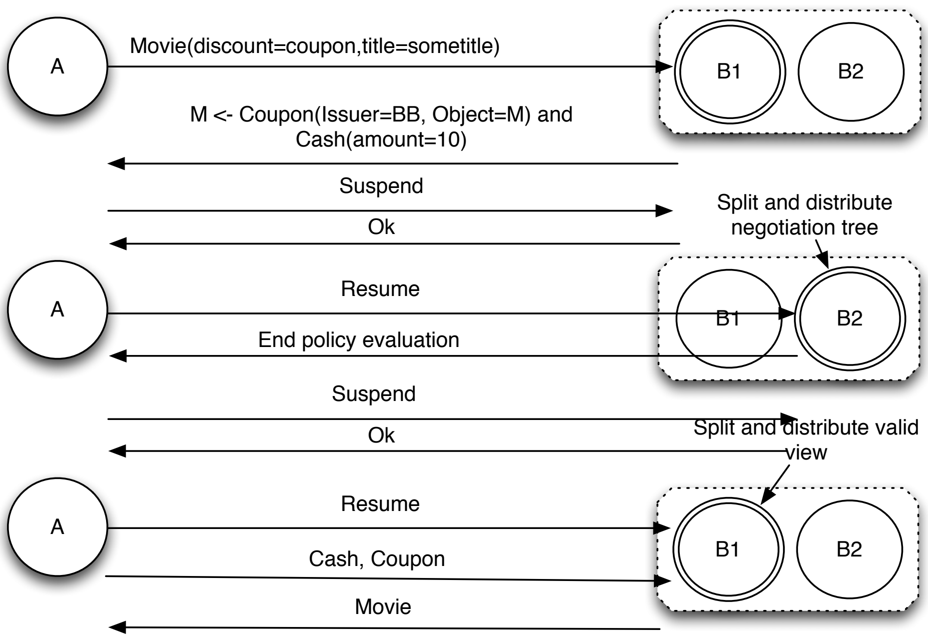

A
Movie(discount=coupon,title=sometitle)
B1
B2
M <- Coupon(Issuer=BB, Object=M) and Cash(amount=10)
Suspend
Ok
Split and distribute negotiation tree
A
Resume
B1
B2
End policy evaluation
Suspend
Ok
Split and distribute valid view
A
Resume
B1
B2
Cash, Coupon
Movie

Figure 5.2: A running example of the multi-session trust negotiation protocol

Before the suspension of the negotiation B_1 creates the information required to save the intermediate state of the negotiation (see the protocol in Section 5.3.4) and sends them to every other server in B's pool.

When A resumes the negotiation, B_1 is temporarily offline for maintenance; thus another server B_2 (in B's pool) continues the negotiation with A. Since the negotiation was suspended during the exchange of the policies, B_2 checks the committed negotiation tree initially received from B_1 and because B_2 has the same policies referred in the tree, it is able to resume the negotiation with A with no changes. A decides that the credential required in the current round by B_2 are not sensitive and thus does not impose any additional policy on the disclosure of this credential. The decision made by A ends the policy evaluation phase of the negotiation because both A and B_2 are now able to identify a valid sequence of credentials that can lead to the negotiation success. This sequence consists of every credential involved in the negotiation. This means that every credential needs to be exchanged and verified in order to successfully complete the negotiation. Both peers switch to the next phase of a Trust-$\mathcal{X}$ negotiation, that is, the credential exchange phase. A asks again to suspend the negotiation because she discovers that her credential attesting the bank account is expired, indicated as Suspend(2) in Figure 5.2. B_2 operates the suspension creating the split version of the credentials' list and sends it to the other servers of B's pool along with the involved credential. After A has renewed her credential, she resumes the negotiation. This time, B_1 is available and being the server nearest to A, it is in charge of continuing the negotiation. This time it operates on behalf of B_2 because B_2 is the issuer of the temporary ticket required by A. The exchange of the credential is then completed by B_1 with credentials by B_2, as described in Section 5.3.5.

This simple example shows that events that are likely to often arise, such as short term validity of credentials or temporary inability of a peer to continue the negotiation, do not impact the negotiation under Trust-$\mathcal{X}$.

5.5 Security analysis

In this section we present a detailed security analysis of the most critical algorithms characterizing the Trust-$\mathcal{X}$ multi-session negotiation, along with the important security properties satisfied by our approach. We also informally discuss the ability of our approach to withstand malicious peers and colluding parties.

5.5.1 Tree Sharing Protocol Analysis

In order to ensure the validity and correctness of multi-session negotiations, three properties must be verified by our protocols. Given a negotiation started between P_1 and P_2 and completed on behalf of P_2 by a delegate P_3, the properties are as follows:

- *Completeness*: P_3, upon successful completion of Protocol 4, obtains a negotiation tree NT'_{open} whose nodes satisfy the corresponding committed terms in $\mathsf{commit}NT_{open}$.
- *Soundness*: P_3, upon successful proof of knowledge for a credential *cred*, can safely omit the names and values contained in *cred* in the construction of NT'_{open}.
- *Zero-knowledge*: P_3 does not gain any information about the committed values in $\mathsf{commit}(NT_{open})$, upon checking its values against them using Protocol 4.

Peer P_3 builds its version of the negotiation tree, NT'_{open}, by a node-by-node execution of proofs of knowledge of committed values obtained by P_2 by application of the Damgaard-Fujisaki commitment scheme. Thus, soundness and completeness properties are guaranteed by such commitment scheme, whose security is based on the following well-known cryptographic assumptions (see Appendix A and in particular [35]):

- *Computational Diffie-Hellman (CDH) assumption.* Given a finite cyclic group G, a group generator g and group elements g^a, g^b, there exists no polynomial time algorithm that computes g^{ab}, with non-negligible probability.
- *Strong RSA assumption.* Given an RSA modulus n and a random value x in $\mathbb{Z}_n^*$, there exists no polynomial time algorithm that computes $e > 1$ and $y \in \mathbb{Z}_n^*$ (with non-negligible probability) such that $y^e = x \mod n$.

- *Random oracle hypothesis.* Informally, a random oracle is a function $H : X \to Y$ uniformly chosen in the set $\mathcal{H}$ of all function from X to Y. It models a real-world cryptographic hash function, such as SHA-1.

We further note that in our proof Protocol 4 the repeated execution of the proof of knowledge does not leak information, since all committed values are independent from each other. That is, P_2 uniformly draws each secret values r to be employed in the commitment scheme. Regarding the zero-knowledge property, assuming the random oracle hypothesis, the non-interactive proofs obtained employing the Fiat-Shamir heuristics are zero-knowledge, because the interactive proofs are zero-knowledge [95].

Regarding the security of the resume in the credential exchange phase, described in Section 5.3.5, we note that P_3 receives an encrypted version $\mathsf{Enc}(Cred_i, k_s)$ of credentials $Cred_i$ for exchange with P_1. However, P_3 is not able to decipher them, since the symmetric key k_s has been sent by P_2 to P_1 only.

Another important aspect of our security model is the security of the delegation process. Essentially, two important properties must be guaranteed. First the delegate must obtain all and only the data necessary to successfully carry on the negotiation. Second, and equally important, the delegated party must not leak information regarding the peer that originally started the negotiation process. The first property is guaranteed by the results stated by Theorem 5.5.5 and Theorem 5.5.1. Based on these results, we can guarantee that the secret share disclosed to the delegate, once combined and opened for the secret disclosure, provides all necessary information to advance the negotiation. The non-disclosure property, instead, guarantees that only the information needed to advance the negotiation is provided to the delegate.

The zero-knowledge property, introduced in Section 5.5.1, guarantees that the delegates cannot infer any information concerning the credentials' content nor the disclosure policies in that the content of the disclosure policies is hidden.

5.5.2 Formal Properties of the Multi-Session Protocol

We begin by introducing a *correctness* property.

Definition 5.5.1 (Trust Negotiation correctness). *A trust negotiation protocol is correct if it successfully completes all and only the negotiations satisfying Definition 3.6.4.*

In our context, correctness refers to the fact that despite suspension and possible changes to the originally exchanged credentials and/or policies, the success of the negotiation is guaranteed if and only if a set of valid credentials satisfying the policies of both parties is identified.

Theorem 5.5.1. *The MS protocol satisfies correctness.*

Proof of Theorem 5.5.1 We prove Theorem 5.5.1 by contradiction. Let $\mathcal{CS}eq = (C_1, \ldots, C_n = R)$ be a sequence obtained at the end of the trust negotiation that does not satisfy Definition 5.5.2. Let $T = \langle \mathcal{N}, R, \mathcal{E} \rangle$ be the negotiation tree from which S has been generated.

There are two possible cases:

1. The sequence is of the form $S = (C_1, \ldots, C_j, \ldots, C_k, \ldots C_n = R)$. Assume without loss of generality that C_j is protected by $pol_j = C_j \leftarrow \phi(t)$, where t^{C_k}. According to [13], pol_j corresponds to a multi-edge $e = (t^{C_j}, t^{C_k}) \in \mathcal{E}$.

   Since $\mathcal{CS}eq$ is given by traversing the valid view associated with T [13], the first credential in the list is C_k, followed by C_j. $S = (C_1, \ldots, C_k, \ldots, C_j, \ldots C_n = R)$ is created by Algorithm 2 presented in [13]. The same criteria is followed by the resume protocol to replace credential, thus preserving the order given by the policies. However, this is in contradiction with our hypothesis of having sequence $S = (C_1, \ldots, C_j, \ldots, C_k, \ldots c_n = R)$, where C_j precedes C_k and the thesis holds.

2. The sequence is of the form $\mathcal{CS}eq = (C_1, \ldots, C_j, \ldots C_n = R)$. Assume without loss of generality that C_j is protected by $pol_j = C_j \leftarrow \phi(t)$, where t^{C_k}. However, C_k does not appear in S. Three possible cases arise:

   (a) C_j is being introduced during the policy evaluation phase. However, if $\nexists n \in N$ s.t. n's term is equal to t^{C_k}, that means that t^{C_j} is carried by a node $n' \in N$ labeled as *open*. Since S is obtained backtracking tree nodes all labeled as *deliv*, $C_j \notin S$. But this is in contradiction with our hypothesis and the thesis holds.

   (b) C_j is being introduced during the credential exchange phase. However, $\mathcal{CS}eq$ is obtained by the terms derived from the tree, and it follows from the previous case that this is not possible. Consider the case in which a credential C'_j is replaced by a substitute s_c^{pol} during the credential exchange phase. If C'_j is selected then$pol_j = s_c^{pol} \leftarrow \phi(t)$, and $C'_j \leftarrow \phi(t')$, where $t = t'$ and thus the substitute should be satisfied by C_k, i.e. $t'^{C_k} =t^{C_k}$. Thus C_k belongs to S. But this is in contradiction with our hypothesis, and the thesis holds.

We notice that there is an exceptional case to this result, that occurs when the peers anticipate the disclosure of credentials, due to the strategy adopted during the negotiation or due to the anticipated disclosure in light of the upcoming switch of the negotiator (see Section 5.3). In this case, the correctness result is still valid, but the credential sequence to be applied is the sequence obtained by adding all the credentials disclosed between P_1 and P_2, say $\{C_1, \ldots, C_K\}$ and between P_1 and P_3, $\{C_{K+1}, \ldots, C_n\}$. That is: $CSeq' = \{C_1, \ldots, C_K\} \bigcup \{C_{K+1}, \ldots, C_n\}$. It is straightforward to apply the above proof to $\mathcal{CS}eq'$. $\Diamond$

Validity is informally defined as the property of using only valid credentials to determine success of the negotiation.

Definition 5.5.2 (Trust negotiation validity). *Let $S = (C_1, \ldots, C_k = R)$ be a credential sequence and $pol_1, \ldots, pol_k$ be the policies protecting $C_1, \ldots, C_k$, respectively. A trust negotiation protocol is valid if all the credentials in S are valid.*

Theorem 5.5.2. *The MS protocol satisfies the validity property.*

The proof is straightforward. Validity of the negotiation is guaranteed by the multiple controls about credentials validity that are an integral part of the negotiation process. Our approach prevents the usage of expired credentials by introducing a preprocessing phase that identifies the critical credentials (collected in the $\mathcal{PSC}$ list), subject to expiration, and by carefully replacing credentials as needed, while preserving the correctness and minimality of the process.

Another important security property is minimality ensuring that regardless of the number of exchange and intermediate steps in the negotiation, only credentials required for the purpose of the negotiation are disclosed. The minimality property is analyzed with respect to the negotiation final outcome and the overall negotiation data exchange. First, we consider minimality with respect to the success of negotiation.

Definition 5.5.3 (Trust negotiation minimality). *Let $S = (C_1, \ldots, C_k = R)$ be a credential sequence and $pol = \{pol_1, \ldots, pol_k\}$ be the policies protecting $C_1, \ldots, C_k$, respectively. The credential sequence is minimal if:*

1. *either for every credential $C_i \in S$ there exists a policy $pol_j = C_i \leftarrow \phi(t_{1,\ldots,t_k})$ such that there exists a term $t \in \{t_1, \ldots, t_k\}$ such that t^{C_i} holds, or $C_i = R$;*

2. *for every policy $pol_i \in P$, there exists $C \subseteq S$ such that pol_i^C holds.*

Notice that a credential C used for completing the negotiation may be a substitute of previously committed credentials. By definition, as long as policies satisfaction is guaranteed, the minimality property is preserved.

Theorem 5.5.3. *The credential sequence $\mathcal{CS}eq$ is minimal.*

Proof of Theorem 5.5.3 The credential sequence $\mathcal{CS}eq$ satisfies Property 1) in Definition 5.5.3: $\mathcal{CS}eq$ contains all the credentials in the valid view as obtained by a run of MS Trust-$\mathcal{X}$ protocol. By definition of valid view, credential C_i is contained in the valid view (or, equivalently, $\mathcal{CS}eq$) if and only if $C_i = R$ or there exists a policy pol_j such that $pol_j^{C_i}$ holds.

The credential sequence $\mathcal{CS}eq$ satisfies Property 2) in Definition 5.5.3: Suppose by contradiction that there exists a policy pol_j which is not satisfied by any subset of $\mathcal{CS}eq$. This implies that there exists a credential C_i whose release is controlled by pol_j, which is not contained in $\mathcal{CS}eq$. Therefore pol_j is not in P, which contradiction.

$\diamondsuit$

The second definition related to minimality is given with respect to the success of the negotiation process itself. This property implies that all credentials and policies exchanged during the negotiation, although they may not be used for the negotiation final sequence of credentials, are required to ensure the negotiation correctness. By nature, the trust negotiation may involve multiple rounds to explore the various alternatives, and thus not leading toward the negotiation success. However, needless rounds are to be prevented, to avoid information leaks and covert channels. Hence, in order to ensure that *minimal* and *correct* negotiations (see Def. 5.5.1) are supported by our protocols, the trust negotiation minimal disclosure must be preserved. Trust negotiation minimal disclosure is defined in what follows.

Definition 5.5.4 (Trust Negotiation Minimal Disclosure). *Let P_1 and P_2 be two peers negotiating for a resource $\mathcal{R}$. Let $\{m_1, \ldots, m_k\}$ be the set of messages exchanged between P_1 and P_2. Let P_3 be P_2's delegate and $\{m_k, \ldots, m_{k+n}\}$ be the second set of messages exchanged by P_1 and P_3, where $m_{k+n} = \mathcal{R}$ or $m_{k+n} = Fail$. A trust negotiation has the minimal information disclosure property if and only if, given the negotiation tree $NT = \langle \mathcal{E}, R, \mathcal{N} \rangle$, and the related credential sequence $\mathcal{CS}eq$ each m_i in $\{m_1, \ldots, m_k\} \cup \{m_k, \ldots, m_{k+n}\}$ is of one of the following types:*

1. *$m_i = ack$ or $m_i = deny$*
2. *$m_i = \{pol_1, \ldots, pol_k\}$ and $\forall\ pol_i = R \leftarrow \phi(t_i, \ldots, t_n) \in m_i \exists$ a corresponding node $n \in \mathcal{N}$*
3. *$m_i = \{c_1, \ldots, c_m\}$ and $\forall c_i \in \mathcal{CS}eq$*
4. *$m_i = S$, that is, a share of a serialized tree*
5. *$m_i = \{c_1, \ldots, c_m\}$ and each $c_i \in m_i$ is s.t.*
   $c_i = s_{c'}^{pol} \wedge c' \in \mathcal{CS}eq$.

Under the assumption that peers are honest and faithfully follow the negotiation protocol, the following property holds.

Theorem 5.5.4. *The MS protocol guarantees minimal disclosure.*

Proof of Theorem 5.5.4 The proof of Theorem 5.5.4 is performed analyzing the design of the protocol. The original Trust-$\mathcal{X}$ protocol described in [13] extended accepts only messages of type 1), 2) and 3) identified in Definition 5.5.4. The features defined in Section 5.3 extends the set of accepted messages with types 4) and 5) making them valid during the suspension and resume of a trust negotiation. Moreover, the negotiation phase refines the types of messages allowed. During the policy evaluation phase only messages of type 1) and 2) are allowed while during the credential exchange phase are the messages of type 1) and 3) the ones allowed to be exchanged.

Any other type of message received will cause the negotiation to be aborted. The same will happen if a message of the wrong type is received during any phase. $\Diamond$

Finally, our protocols satisfy the following non-disclosure property. Given a negotiation, a corresponding re-awakened negotiation does not disclose any additional information other than the one intended by the negotiation before suspension. We notice that this property holds even in case of replaced negotiations.

Theorem 5.5.5. *The resume of a MS negotiation satisfies the non-disclosure property.*

Proof of Theorem 5.5.5 In order to prove that a re-awakened negotiation in which a credential has been substituted with an equivalent one does not disclose more information than the original negotiation, we have to extend the definition of equivalence to negotiations. We assume that the information released by a data item x and its corresponding semantic mapping $\theta(x)$ are the same. We formalize such assumption as follows: we say that two data items x, $\theta(x)$ are *similar* if for every probabilistic, polynomial time bounded algorithm $\mathcal{O}$ (referred thereon as the *observer*) there exist a $\epsilon > 0$ such that the following inequality holds

$$|\Pr\left[\mathcal{O}(x) = 1\right] - \Pr\left[\mathcal{O}(\theta(x))) = 1\right]| \leq \epsilon \tag{5.2}$$

Next, we define two negotiations N_1 and N_2 (intended as the lists of the exchanged credentials) as *equivalent* in the case that, for every probabilistic, polynomial time bounded algorithm $\mathcal{O}_{neg}$ there exist an $\epsilon > 0$, such that the following inequality holds

$$\left|\Pr\left[\mathcal{O}_{neg}(N_1) = 1\right] \Pr\left[\mathcal{O}_{neg}(N_2) = 1\right]\right| \leq \epsilon. \tag{5.3}$$

The proof amounts to show that when substituting a credential in N with an equivalent one, we get a negotiation N' equivalent to N.

In order to show this, consider the set of exchanged credentials $\{C_1, \ldots, C_u\}$ during negotiation N. By hypothesis, such credentials are exchanged using the selective disclosure protocol presented in [106].

Now consider the negotiation N', which differs with N in a single credential C_i'. Suppose further that credential C_i' is such that the non-committed attributes are similar according to Definition 5.2.3. Credential C_i' is equivalent to C_i (with respect to the disclosure policy under which C_i was released). In fact, given that the commitment scheme used in the selective disclosure protocol satisfies the hiding property, we have that two committed values are similar as well. This means that there exist no computationally bounded observer able to discern C from C' (up to a negligible quantity ϵ). This implies such an observer does not exists for N and N' either, given that C and C' is the only spot in which they differ. $\Diamond$

5.5.3 Protocols Resiliency To Malicious Or Colluding Parties

Our protocol does not explicitly deal with protection of data against hacking and tampering. Such threats are prevented by standard encryption techniques [107]. Another form of attack the protocol might be vulnerable to is the replay attack. However, protection from such an attack can be easily achieved by using timestamps when messages are exchanged.

We now briefly discuss how our protocol operates if assumptions about the honesty of the peers are relaxed. Under our approach, the resume procedure may generate a tree view containing nodes referring to non-existing terms because of a peer lying about the saved tree. In such a case the negotiation process will fail because of the lack of the corresponding credentials to disclose during the credential exchange phase. Even if the honest peer did not have the complete state of the negotiation handy, because of the assumption of unilateral negotiations, after the suspension phase both the peers will have reconstructed the intermediate state of the negotiation, which provides enough information to detect malicious requests. Finally, in case the negotiation was never suspended and one party had the whole process on its side, we assume that critical information related to the credentials involved and the corresponding policies will be available to it, and sufficient to detect the malicious behavior.

Similarly, fake credentials or terms may be inserted in a $\mathcal{CS}eq$. In such a case the negotiation process will fail because of the lack of the corresponding credentials to disclose during the credential exchange phase, as indicated by the $\mathcal{CS}eq$. Whether the malicious peer is the original peer participating in the negotiation or the delegate, a mismatch of the exchanged credentials with respect to the shared credential will be determined. If the originator tries to alter the content of one of its credentials, the delegate will not find the matching item in the list. If it rejects a decrypted credential belonging to P_2, it may obtain sensitive information and then purposely failing the negotiation to fail. The usage of selective disclosure actually defeats such tedious attacks, since the attacker can only gain very limited information.

Finally, we remark that the secret sharing protocol and the tree protocols ensure that any possessor of the secret share cannot infer any information regarding the negotiation process. Even in case two delegated peers, say P_3 and P_4, collude to gather information regarding the negotiation, they cannot gather any private information. If $S = \mathsf{ser}(\mathsf{commit}(NT_{open}))$ is reconstructed, the only data gathered by the malicious peers is a committed version of some of the tree nodes. In case $S = \mathcal{CS}eq$, the colluding peers can only obtain a sequence of encrypted credentials, which security relies on the strength of the adopted encryption protocol.

5.6 Complexity Analysis

In this section we analyze the complexity of our protocols, in terms of the time and space and in terms of the number of exchanged messages. We consider the case in which one of the two peers may change, during the resumption of a trust negotiation. For the case in which peers do not change, we refer to [105] for results about complexity.

5.6.1 Policy Exchange Phase Complexity

Unsurprisingly, the nodes commitment, and tree sharing phases (along with the non-interactive proofs associated with the commitments) are the most computing-intensive parts of our approach. We start by analyzing them separately, by evaluating the number of required multiplications.

Nodes commitment. Suppose that the negotiation tree NT_{open} contains n nodes. Suppose further that the maximum number of predicates contained in a term is p_{max}. Then, the number of commitments to be computed is $n(p_{max}+1)$. Each commitment computation entails two modular exponentiations and each of them can be computed in $O(log(\max\{m,r\}))$ multiplications. Thus the total number of multiplications needed to commit all the nodes in NT_{open} is $O(n \cdot log(\max\{m,r\}))$. In real-world settings, the value of p_{max} usually does not exceed two or three.

Tree sharing. (see Section 5.6.2 and [58]). The modified Shamir's secret sharing scheme does not add significative computational overhead and very efficient polylog algorithms for polynomial evaluation and interpolation are available [3].

Commitments proofs. For each committed value, an equality zero-knowledge proof of knowledge requires P_2 and P_3 to compute each a modular exponentiation (other than those required for computing the commitments themselves, of course). For a zero-knowledge proof that a committed value lies in a given interval, P_2 must compute two additional commitments and five zero-knowledge equality proofs [22].

Hence, the overall time complexity, measured by the number of multiplications needed to execute the commitment, sharing, and zero-knowledge proofs protocols, linearly depends from the number of nodes contained in NT_{open}.

5.6.2 Communication Complexity

We measure the communication complexity in terms of the number of exchanged messages and in their size. The total number of messages that need to be exchanged is five, assuming that P_2 supports multicast and without taking into account the authentication between P_1 and P_3, as detailed in the following. Once one peer among P_1 and P_2 requests a suspension (1st message), P_2 computes the two shares of the committed version of NT_{open} and sends them to P_1 (2nd message) and $P_3, \ldots, P_n$ (3rd message, assuming a multicast mechanism for the peers in P_2's pool), respectively. After that, P_1 sends a resume request to P_3 (4th message) and, in case of its correct authentication, P_1 sends its share to it (5th message). Once P_3 has correctly built its version of the negotiation tree NT'_{open}, P_3 sends it to P_1 (6th message).

The upper bound on the size of exchanged messages is the size of a share of the negotiation tree. Recall that the space-efficient version of the Shamir's secret sharing scheme operates on a linearized version of the (node-by-node committed) tree NT_{open}, denoted by $\mathsf{ser}(\mathsf{commit}(NT_{open}))$. The size of a committed node linearly depends on the size of security parameter s, typically an integer with a number of bits between 512 to 1024. Hence, the size of $\mathsf{ser}(\mathsf{commit}(NT_{open}))$ linearly depends on the number of nodes in NT_{open}. Finally, note that the space-efficient version of the Shamir's secret sharing scheme yields two shares whose length is half the length of $\mathsf{ser}(\mathsf{commit}(NT_{open}))$ [58].

5.6.3 Credential Exchange Phase Complexity

The complexity of the operations performed in the credential exchange phase is significantly smaller than the complexity of the policy evaluation phase.

According to the protocol presented in Section 5.3.5, during the suspension phase, 3 messages are exchanged:

Message 1: the request from P_1 to P_2 to suspend the credential exchange;

Message 2: the message from P_2 to P_1 containing the credential sequence and the corresponding encryption keys;

Message 3: the message containing the credential sequence along with the encrypted credentials belonging to P_2, which are substituted to the proper terms. This message is sent from P_2 to $P_3, \ldots, P_n$ (assuming, again, a multicast mechanism for the peers in P_2's pool).

In addition, one additional message is sent by P_1 to P_3 to resume the negotiation. Hence, in case of correct authentication, the credential exchange phase resumes. The computational complexity of the second message depends on the symmetric encryption scheme chosen. Since standard symmetric encryption schemes (e.g. AES and/or DES) have linear time complexity, we can safely assume that the time complexity of suspension and resume of the credential exchange phase is linear in the number of P_2's credentials contained the credential sequence.

The upper bound on the size of exchanged messages coincides with the size of the 3rd message. The reason is that such message contains the credential sequence *and* the encrypted credentials. The number of encrypted credentials linearly depends on the number of P_2's credentials contained in the credential sequence as well as on the strategy adopted on how to deal with the substituted credentials.

Chapter 6

TN for information sharing in critical infrastructures

6.1 Introduction to the application environment

The first application scenario in which we applied the techniques presented in the previous chapters is the Critical Infrastructure environment. In particular, the context we are describing is related to a community of Energy infrastructure actors.

The peculiarities of the community of Energy Infrastructure actors are several:

- It is a geographically sparse community
- The involved actors can be clustered in categories with widely different needs, characteristics and roles
- The kind of information which flows among the community goes from completely unclassified to highly reserved, and different actors can have different disclosure rights on such information
- The relationships between the different actors of the community can be extremely complicated and controversial, having to deal for example with governing authorities, market competitors etc.
- The effects of a failure in the information sharing process could have different level of impact (market impact, citizen impact etc.)

Roughly speaking we are talking of a community composed by different level of entities (customers, authorities, hardware providers, energy producers, energy deliverer etc.), which traditionally are not used to collaborate in order to share security information.

This lack of collaboration is mainly due to three reasons:

1. **Economical**: making other competitors aware of internal vulnerabilities, could put a certain company in a critical situation from a pure market point of view.

2. **Political**: the interaction among governing authorities and/or between authorities and other entities, have always been difficult and time/resource consuming.

3. All the aspects related with the confidentiality of the data and the trust level of the different entities have always acted as a barrier, discouraging any attempt of collaboration in this direction.

Here we concentrate our attention to the last of these problems since once solved the problem of confidentiality and trust also the relevance of the other two will drastically diminish.

In the following a brief description of the different actors involved in our motivating scenario is given:

Government Authorities a Government authority is the national entity directly dependent from the national government entitled to control the national operators, to regulate the Energy normative and legislation, to interact with Foreign Government authorities in order to coordinate actions at international level. In our scenario different Government authorities can exist in the national context. Each one can directly interact with the other entities of the same country and only indirectly with entities of other countries. There exists an exception: government authorities of different countries can interact. Moreover, every Government authority maintains a table of trust related to the authorities of other countries. Such a table obviously reflects the foreign politics of every country. On the basis of such table, the different authorities may have or not the access to selected portion of the information shared/requested.

Producers in this class fall all the industries producing software and hardware for Energy (or more in general for industrial) infrastructures (SCADA systems, field actuators etc.). Such Producers, can share information – if they have some sort of common agreement –, or with the final users – if the product the user have bought are under guarantee. Other details about the information policies will be explained in the proper section.

R&D under this class fall all the research centers, universities etc. involved in the research in the field of critical infrastructure. Also in this case R&D can be groped into consortiums which can share all the information or only portions of information about their results. The R&D can be associated into mixed consortiums having partners like Producers, final Users etc.

End Users the End users are the owner of the critical infrastructures. They can share information with R&D, with other E.U., with producers.

Generic Entities , that is entities which cannot be classified in one of the previous classes. Their level of information disclosure can be considered minimum.

The possible interactions between the actors described can be extremely complex. The core of such interaction is obviously an exchange of information which can have different level of sensibility and criticality. In the following, some example of such flows are shortly described:

- *Security information*: exchange of security knowledge related to new vulnerabilities – like hardware, software, architectural etc. –, new attacks, new threats. Such information can be exchanged between end users, between end users and producers, etc. In any case, is evident how the same information can be posed under different disclosure constraints depending from the identity of the actor requiring the information
- *International threats*: Government Authorities can for example exchange information about international threats, and then disclose partially such information to the companies based inside their national borders.
- *Common Policies*: Government Authorities could, for example, exchange information about common policies, reach new agreements and then inform the national companies –producers and end users)

6.2 Information sharing in critical infrastructure environment

The information sharing problem is, first of all a problem of *modelling*, in other words, before being shared the information has to be modeled. The Infrastructure Modeling and Security modeling has been treated mainly for operational and security analysis purposes. Different approaches have been suggested in order to model systems in light of security. In the work presented by Alberts & Dorofee [5], the authors propose a first idea of system description to be used for security purposes. However such a description lacks mainly in two points: it is not formal and, more relevant, it cannot deal with complex System-of-Systems. Folker den Braber et al. present in [37] a risk assessment approach partially based on a system description. It tries to partially capture the concept of adverse environment by introducing (using UML) the concept of Threat Scenario, of course, is an advance in the representation of systems that could be adapted for the description of several interacting systems. Nevertheless, this was not the intention of the authors and represents only a possible adaptation of the methodology. Masera and Nai Fovino in four correlated works [71, 70, 72, 67] present an approach based on the concept of system of systems, that preserves the operational and managerial independence of the components while capturing at the same time the concept of relationship among components, services and subsystems. In the present work we adopt this modeling approach for describing information related to the architecture of a critical infrastructure. Another relevant information to be shared is related more strictly to security issues like attack scenarios, threats and vulnerabilities. Again, in the scientific literature, there exist several methods used to describe security information related to malicious acts. Historically the first approach in that sense was related to the creation of vulnerability databases (of which Bugtraq [94] is an example). However, they are basically focused on the description of vulnerabilities, lacking completely (but that isn't their goal) the description of the means and ways by which they can be exploited by attacks. The most promising approach capturing the latter characteristic is the Graph Based Attack Models [108]. In this category two can be considered the main modeling approaches: Petri Net based Models and Attack Trees models. A good example of the first category is the Attack Net Model introduced by McDermott [68] in which the places of a Petri Net represent the steps of an attack and the transitions are used to capture precise actions performed by the attackers. The second approach (attack trees), proposed originally by Bruce Schneier [90] is based on the use of expansion trees to show the different attack lines that could affect a system describing their steps and their interrelationships. Such an approach has been extended by Masera and Nai Fovino [72] by introducing the concept of Attack Projection. In the present work, to represent attacks and threat information, we have adopted this approach.

6.2.1 Working example

Consider the following example: a Power Producer Company discovers a major vulnerability in its infrastructure. To achieve the main scope of the information sharing network, such company should share in detail all the information related to the discovered vulnerabilities with all the other companies, with the government authorities and, if needed, with the hardware producers.

Let considers now the implication of the diffusion of such information: the hardware producers, in order to fix completely vulnerability may need to know in detail the infrastructure of the company in which such vulnerability has been discovered. On the other hand, since the detailed infrastructural information may represent an added value for the company in the sense that it could be for example the results of huge economic investments for optimizing the production. In that case, to provide the same information to a direct competitor (i.e. the others Power companies), represents, indirectly, a loss of money. Moreover, this information could cause an image and an economic damage to the company if used in a malicious way.

These few example are sufficient to argue some desirable features a framework for information sharing and exchange should provide:

1. A language allowing to express the policies and, more generally, the constraints which must be satisfied in order to request and obtain information

2. A set of mechanism allowing to define different level of trust, on the basis of which select which portion of the information can be released

3. An architecture implementing a negotiation mechanism for the exchange of the information allowing to define such levels of trust

4. The possibility, for a group of actors, to elect a member able, in certain cases, to conduct the negotiation for the entire group, for example in case of negotiation between government authorities.

The framework here presented addresses such features.

6.3 Knowledge Characterization

In the current application scenario, the framework has been adapted in order to share information about Power Grids, under a security and safety perspective. A relevant issue in an Information Sharing Framework, is related to the kind and format of the information to be shared. [71, 70, 67], presented a comprehensive framework allowing to model both systems (from an architectural, service oriented point of view) and security knowledge. We decided to adopt such compact modeling approach to model the information to be shared. A system or an architecture is then modeled in term of components, services provided, relationship among components and subsystems. At the same way, according to [72], the Security Knowledge is modeled in term of *vulnerabilities*, *attacks* and *threats*. For major details about system and security knowledge modeling please refer to [71, 72]. In Figure 6.1 and Figure 6.3 are shown two examples of XML representation: of an attack tree and of a system component . Moreover, in Figure 6.2, the graphical representation of an attack tree is provided.

6.4 Language extensions

The application scenario presented in the current Section extends the definition expressed in Chapter 3. Such extensions are conservative with respect of the original language in the sense that, if there is no match between the disclosured policies in $\mathcal{X}$-RNL, then the negotiation fails, while in the newly proposed language it is possible to re-negotiate the originally requested resource and to exploit a successful negotiation to obtain more informations related to the previous one.

6.4.1 Resource level negotiation

Up to now, $\mathcal{X}$-RNL provides only all-or-nothing disclosure: That is, a resource R can only be disclosured or not. As shown in Section 6.2.1, there are very natural scenarios motivating the introduction of a more flexible way of disclosing complex resources, as found in a CI setting.

To achieve that we need to introduce some formal definition about the level of disclosure and to grant to the parties involved in the negotiation the required tools.

To begin, we introduce the concept of *level of (resource) disclosure*.
We begin refining the Definition 3.2.2 presented in Chapter 3.

Definition 6.4.1 (Atomic resource). *An* atomic resource *is a resource that can't be further divided in smaller pieces.*

Definition 6.4.2 (Composite resource). *A* composite resource *is a resource of the form $R = r_1, r_2, \ldots, r_u$ where each r_i is an atomic resource.*

Using the what just explained, is finally possible to define the concept mentioned above.

Definition 6.4.3 (Levels of (resource) disclosure). *Given a resource, or service, R, we define a series of* levels of disclosure *over R, denoted by R^0, R^1, R^2, R^3, ..., R^v, as follows:*

Considering the composite resource $R = \{r_1, r_2, \ldots, r_t\}$, where each r_i is an atomic resource, the levels of disclosure can be seen as coverings over R where $R^0 = R$ and $R^v = \varnothing$.

The remaining levels are chosen in a way to define a partial order among them.

How this association is performed is mandated to the users. In Examples 6.4.1 and 6.4.2 are shown two of the possible levels definitions.

Example 6.4.1. *Considering the resource R, defined in Figure 6.1, identified with the term $Library(Type = \text{“4”})$ it is possible to define on it, for example, 5 levels of disclosure:*

- $R^0 = Library(Type = \text{“4”})[1,2,3,4] = Library(Type = \text{“4”})$
- $R^1 = Library(Type = \text{“4”})[1,2,3]$
- $R^2 = Library(Type = \text{“4”})[1,2]$
- $R^3 = Library(Type = \text{“4”})[1]$
- $R^4 = \varnothing$

Note that R^0 is also equivalent to $Library(Type\text{“4”})$.

Example 6.4.2. *Again, consider the resource R, defined in Figure 6.1, it is possible to define a different set of levels of disclosure, composed that time of 8 levels, as follows:*

- $R^0 = Library(Type = \text{“4”})[1,2,3,4] = Library(Type = \text{“4”})$
- $R^1 = Library(Type = \text{“4”})[1,2]$
  - $R^2 = Library(Type = \text{“4”})[1]$
  - $R^3 = Library(Type = \text{“4”})[2]$
- $R^4 = Library(Type = \text{“4”})[3,4]$
  - $R^5 = Library(Type = \text{“4”})[3]$
  - $R^6 = Library(Type = \text{“4”})[4]$
- $R^7 = \varnothing$

Again, as in Example 6.4.1, R^0 may be simply written as $Library(Type\text{“4”})$.

6.4.2 Shrink the resource level

We now show how to express the resource level to be actually negotiated. In order to do this, we introduce a new kind of formulae, called *rebate formulae*.

A rebate formula is an expression of the form $\rho(\tau_1, \ldots, \tau_k)$ where each τ_i is a term referring to the corresponding one in a previously exchanged disclosure policy, that means, given a previously exchanged disclosure policy $R \leftarrow \phi(t_1, \ldots, t_u)$ where $\phi(t_1, \ldots, t_u) \equiv t_1 \wedge \ldots \wedge t_u$, $k < u$ and $\forall i$, $1 \leq i \leq k$, $\exists j$ such that $\tau_i = t_j$. The meaning of such expression is that the peer is offering a subset of the terms included in the formula of the disclosure policy, in to access the term corresponding to the (sub-)resource protected by the policy itself.

With the adoption of rebate formulae it is possible to negotiate the level of the resource, introducing thus a new, key phase in the trust negotiation.

If the reply to the initial request of the resource R is of the form

$$R \leftarrow \phi(t_1, t_2, \ldots, t_u)$$

where the formula $\phi(t_1, t_2, \ldots, t_u) \equiv t_1 \wedge t_2 \wedge \ldots \wedge t_u$, then it is possible for the initial requester to provide a subset of terms contained in ϕ in order to obtain at least a subset of the atomic resources composing R that the provider define as appropriate for the disclosure terms. The requester rebate with the rebate formula

$$\rho(\tau_1, \tau_2, \ldots, \tau_k)$$

constructed as described before.

If the controller of the resource R accepts the offered terms as enough to disclosure the resource, it replies with a disclosure policy of the form $R \leftarrow \rho(\tau_1, \ldots, \tau_k)$ and the negotiation continues in the normal way. If the controller decides that the offered terms are not enought, it replies with a disclosure policy of the form $R^l \leftarrow \rho(\tau_1, \ldots, \tau_k)$ where R^l is the level of disclosure associated by the controller with the offered terms $\tau_1, \ldots, \tau_k$.

Such process is called *level negotiation*.

To be able to identify the level associate to a give subset of terms, we need to define some tools function.

Definition 6.4.4 (Term evaluation function). *A* term evaluation function *is a function defined as follows:*

$$w : \mathcal{T} \rightarrow \mathbb{N} \tag{6.1}$$

Where $\mathcal{T}$ is the set of terms known by a party. That function associates to each term t a value n, expressing the importance each party associates to the term t.

Definition 6.4.5 (Level of disclosure threshold function). *A* level of disclosure threshold function *is a function defined as follows:*

$$s : \mathcal{L} \rightarrow \mathbb{N} \tag{6.2}$$

Where $\mathcal{L}$ is a set of all disclosure levels. That function associates a value to each disclosure level. The meaning of that association is to define a threshold above which the level can be disclosured.

An example of the usage of what have just been described is shown in Example 6.4.3.

Example 6.4.3. *Let the attack tree show in Figure 6.1 be the resource controlled by a party called Controller (**C**). A party, called Requester (**R**), wants to access that resource. **R** sends to **C** a request of the form: $Library(Type = \text{“}4\text{”})$ **C** replies with the disclosure policy related to $Library(Type = \text{“}4\text{”})$. Let it be $Library(Type = \text{“}4\text{”}) \leftarrow \phi(t_1, t_2, t_3, t_4)$ where the formula $\phi(t_1, t_2, t_3, t_4) \equiv t_1 \wedge t_2 \wedge t_3 \wedge t_4$. **R** doesn't own t_2 and t_3, or doesn't want to disclose them to **C**, it replies with the rebate formula $\rho(t_1, t_4) \equiv t_1 \wedge t_4$ Following Algorithm 5 **C** identify as R^2, according to Example 6.4.1, the appropriate level of disclosure for the terms offered by **R**. It replies with the new resource requested and the new associated disclosure policy $Library(Type = \text{“}4\text{”})[1,2] \leftarrow \phi(t_1, t_4)$ where $\phi(t_1, t_4) \equiv t_1 \wedge t_4$. If **C** accept to access the offered level of disclosure over the resource R, it replies with the disclosure policies associated to the involved terms. If it refuses it can try to offer even less terms or give up with the negotiation. Figure 6.4 shows the modification occurred within the negotiation tree according to the use of the rebate formula $\rho(t_1, t_4)$.*

```
/* $\mathcal{L}$ is the set of the level of disclosure, $w$ is the level of
   disclosure threshold function, $s$ is the term evaluation function, $o$
   is the last level of disclosure used, if not provided $0 = 0$        */
Data: $\mathcal{L} = \{R^0, \dots, R^v\}, w, s, o, \rho(\tau_1, \dots, \tau_k)$
Result: $l$ = level associated to $\rho(\tau_1, \dots, \tau_k)$
begin
    $val = \sum_{i=0}^{k} w(\tau_i)$;
    for ($i = o$; $R^i \in \mathcal{L}$; $i$++) do
        if $val \geq s(R^i)$ then
            $l = i$;
            break;
    return $l$;
```

Algorithm 5: Identifies the disclosure level appropriate for the terms offered by the counter-party

6.4.3 Enlarge the resource level

The second tool provided allows the user to re-negotiate the resource argument of a previous negotiation to try to access a level of disclosure R^m upon the resource R which expresses more information then the level of disclosure R^l. Such is obtained using the so called *extend formulae*. An extend is an expression of the form $\xi(< R^l, h(\mathfrak{t}) >, \tau_1, \ldots, \tau_k)$, where R is the initial resource previously negotiated and each τ_i is a term referring to the corresponding disclosure policy exchanged by the other peer. Let be $R \leftarrow \phi(t_1, \ldots, t_u)$ with $\phi(t_1, \ldots, t_u) \equiv t_1 \wedge \ldots \wedge t_u$ the disclosure policy for the resource R, with $u \leq k$. $\forall i$, $1 \leq i \leq k$, $\exists j$ such that $\tau_i = t_j$. Each τ_i is a term not included within a rebate formula used in a previous negotiation for the resource R. The meaning of such expression is that the peer sending the *extend formula* extends the set of terms offered for the resource R in such a way to be able to obtain access to a disclosure level m, $m > l$

Negotiation history According to [106], it is possible to suspend and resume an ongoing negotiation. Exploiting the savepoint technique presented in [105], it is possible for a peer to expand the resource obtained by means of a previous negotiation. In [13], it is described that at the end of a negotiation the peer providing the resource will release a *Trust Ticket* which is a ticket attesting the successful ending of the negotiation for the resource R with level of disclosure l, as mentioned before. Slightly altering the meaning of Trust Ticket, we associate with it the negotiation tree obtained by the negotiation of the required resource level R. Such negotiation tree has to been saved by the controller which, at the end of the negotiation, will release it. The new version of the definition of Trust Ticket (shown in Definition 6.4.6 contains the (level of) resource the Controller released during the negotiation just successfully ended and the hash value of the negotiation tree created.

Definition 6.4.6 (Trust Ticket)**.** *Formally, a trust ticket is a tuple of he form*

$$< R^l, h(\mathfrak{t}) >$$

where R^l is the level of the resource R argument of the just successfully terminated negotiation and $h(\mathfrak{t})$ [105] is the Merkle hash tree [69] of the associated negotiation tree.

Intuitively, it identifies the negotiation by means of the identification of the resource, its level of disclosure and the negotiation tree that facilitated the release of R^l.

Level expansion Let $< R^l, h(\mathfrak{t}) >$ be the Trust Ticket as described in [13] and in Definition 6.4.6, where $\mathfrak{t}$ is the negotiation tree for the negotiation occurred between the owner of the Trust Ticket and the emitter to allow the owner to access the level l of the resource R. $h(\mathfrak{t})$ is the hash value of the negotiation tree $\mathfrak{t}$. h is a hash function defined as follow:

$$h : \mathfrak{T} \rightarrow \mathbb{N} \tag{6.3}$$

Where $\mathfrak{T}$ is the set of possible negotiation trees. By means of extend formulae, a Requester resumes the successfully terminated negotiation for R^l trying to obtain access to a level of disclosure m, likely with $m \leq l$.

Such process is called *level expansion* and it exploits the tool functions defined for the *level negotiation*.

The difference between level negotiation and level expansion is that the first is used to reduce the amount of terms used in the negotiation in order to be able to access at least a lesser but acceptable subset of the resource according to the terms owned by the requester. With the level expansion we authorize a peer to access a greater level of the previously negotiated resource without the need to begin a completely new negotiation but exploiting the work done before. An example of usage of the extend formulae is shown in Example 6.4.4

Example 6.4.4. *With respect to Example 6.4.3, let* $< R^2, h(\mathfrak{t}) >$ *be the Trust Ticket released after the successfully terminated negotiation between the Controller (**C**) and the Requester (**R**), where* $R^2 = Library(Type = \text{“4”})[1,2]$ *Now, **R** wants to access more information contained in R and resume the negotiation with **C** sending it the expand formula* $\xi(< R^2, h(\mathfrak{t}) >, t_3)$. *Resuming the negotiation tree* $\mathfrak{t}$ *previously used, **C** discovers that it negotiated with **R** for the level of disclosure* R^2. *With the extend formula* $\xi(< R^2, h(\mathfrak{t}) >, t_3)$, ***R** offers the term* t_3 *which was not used in the past negotiation. **C** decides that* $w(t_1, t_2, t_3) > s(R^1)$, *which means that the terms* t_1, t_2 *and* t_3 *are enough to satisfy the disclosure level* $R^1 = Library(Type = \text{“4”})[1,2,3]$. *After that, **C** replies to **R** with the disclosure policy offered by **R**:*

$$Library(Type = \text{“4”})[1,2,3] \leftarrow \phi(t_1, t_2, t_3) \equiv t_1 \wedge t_2 \wedge t_3$$

Figure 6.5 shows the modification made to the negotiation tree $\mathfrak{t}$ *with the use of the extend formula* $\xi(< R^2, h(\mathfrak{t}) >, t_3)$.

```
<?xml version="1.0" encoding="utf-8"?>
<Library Type="4" Id="4"
   Guid="3a4b70ff-150a-4266-997d-1278abce8758"
   MajorVersion="0" MinorVersion="1">
  <Name>Default Library of Attack Patterns</Name>
  <Description />
  <Date>29-02-2008 12:41</Date>
  <AttackTreeList>
    <AttackTree Id="13" IsMacro="0">
      <Name>Apache B.O. Attack</Name>
      <Description />
      <Expertise />
      <Resource />
      <AttackEvaluation Plausibility="-1"
         Severity="-1" />
      <Composition>
        <LogicPort Id="159">
          <Name>AND</Name>
          <Goal>Remote Shell with Root rights</Goal>
          <Children>
            <VulnNode Id="160">
              <Name>Vulnerability</Name>
              <Description />
              <Vulnerabilities>
                <IdVuln>34</IdVuln>
              </Vulnerabilities>
            </VulnNode>
            <Assertion Id="161">
              <Name>AssertComp</Name>
              <Plausibility>1</Plausibility>
              <Description />
              <Components>
                <IdComponent>1</IdComponent>
              </Components>
            </Assertion>
            </Children>
        </LogicPort>
      </Composition>
    </AttackTree>
  </AttackTreeList>
</Library>
```

Figure 6.1: An example of XML document describing a simple attack tree

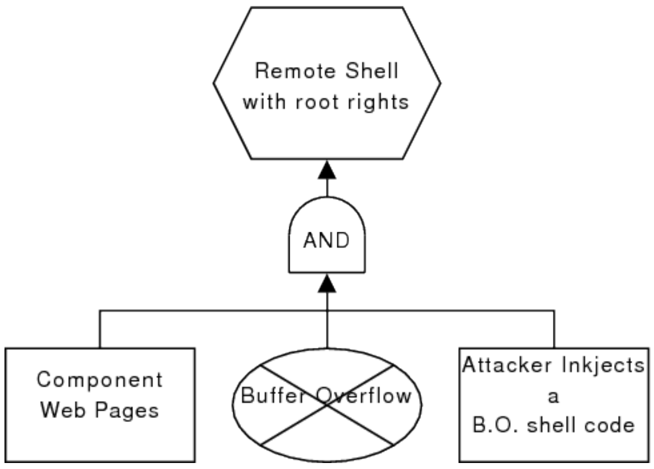

Remote Shell with root rights
AND
Component Web Pages
Buffer Overflow
Attacker Inkjects a B.O. shell code

Figure 6.2: An example of attack tree

```
<Component Id="77" FailureRate="0">
 <Name>Combustion chamber (CCH)</Name>
 <SecurityManager />
 <ClassC />
 <Description>
 It transforms the chemical energy of gas
 into kinetic energy of the exhaust gas
 that is then delivered to the turbine
  </Description>
 <Brand />
 <Model />
 <Version />
 <IdComponentType>5</IdComponentType>
 <Vulnerabilities />
 <Roles />
 <SecurityPolicies />
 <ServiceList>
  <LowLevelService Id="51" Threshold="100"
        StateActive="0">
   <Name>Combustion CCH_C</Name>
   <SecurityManager />
   <Description>
    Combustion of the air-gas mix
   </Description>
   <Type />
   <Roles />
   <SecurityPolicies />
   <DocumentList />
  </LowLevelService>
 </ServiceList>
 <DocumentList />
</Component>
```

Figure 6.3: An example of XML document describing a system component

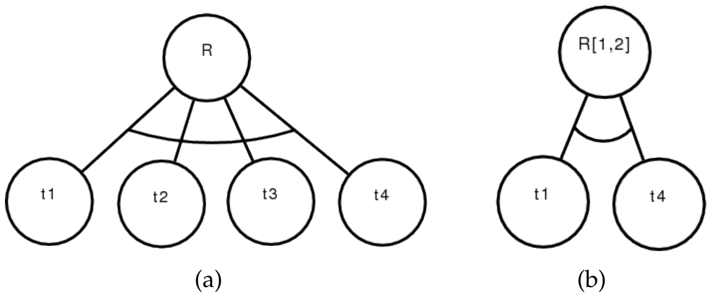


Figure 6.4: The negotiation tree before (a) and after the use of the rebate formula (b)

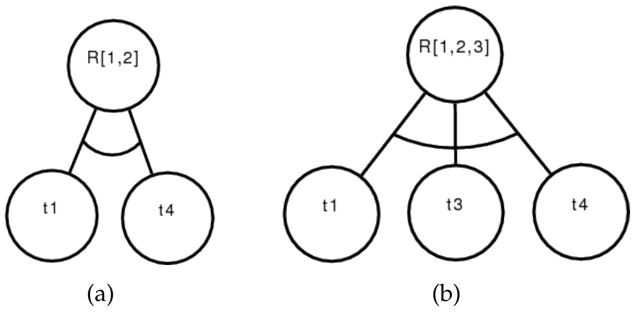


Figure 6.5: The negotiation tree of the saved negotiation (a) and the same expanded (b)

Chapter 7

TN for spectrum sharing in cognitive radion networks

7.1 Scenario introduction

The second application scenario in which we applied the techniques previously presented is the cognitive radio management.

In the current approach for fixed spectrum management, the radio frequency spectrum is divided in separate bands which are allocated to specific wireless services or licensed to providers of wireless services. Wireless services may include wireless communications, positioning or navigation systems like GPS, broadcast services like TV or other technologies like radar or RFID. The shortcoming of the fixed spectrum management approach as described in [109], is the risk of poor spectrum utilization because some spectrum bands may be underused most of the time while other bands may be overused or congested. Spectrum utilization has become a critical issues in recent years because of the needs to provide broadband wireless connectivity to an increasing number of users and to activate new wireless services. An alternative approach is called Dynamic Spectrum Access (DSA), where the allocation of spectrum bands can change in time or space. If a specific spectrum bands is not used, it can be dynamically reallocated to another user for a specific amount of time or in a specific geographical area.

In the fixed spectrum management approach, wireless equipments (e.g. communication terminals) are designed and implemented to use predefined spectrum bands. In the DSA approach, the radio terminals and the radio network infrastructures can transmit and receive in a wide range of spectrum bands and change dynamically their transmission parameters depending on the environment, operational context and needs of the users. Cognitive radio (CR) is an enabler for DSA as it provides the capabilities to implement a flexible use of the radio frequency (RF) spectrum.

In the DSA approach, communication systems based on cognitive radio nodes and terminals could effectively "share" the available spectrum resources and change dynamically the allocation of the spectrum bands for the various communication services. This is a radical change from the fixed spectrum management approach as cognitive radio nodes would not be limited to use specific spectrum bands but they could access all the available spectrum resources within the constraints defined by spectrum regulators. A major consequence of DSA and spectrum "sharing" is the need to define suitable access policies, which describe the rules for sharing spectrum resources. Spectrum sharing policies are particularly important for the application of DSA in the public safety domain where communication systems must satisfy severe requirements in terms of resilience, security and performance.

New applications like mobile video-surveillance, mobile biometric identification and remote emergency health have increased the need for broadband wireless communications in the public safety domain. Higher data throughput requires a wider allocation of spectrum to public safety, but this may not be possible in the current spectrum regulation framework. Innovative approaches like DSA has been investigated as a potential solution to provide broadband wireless connectivity in [11] but only if it satisfies specific requirements including reliances, security and the prioritization of "shared" spectrum resources.

As a motivating scenario related to public safety, consider the organizations participating to the resolution of an emergency crisis. They may include heterogeneous entities such as police, fire fighters, emergency medical services, volunteers organization, border security guards and military forces. Such entities usually have different priorities regarding resources' access and may interact in complex ways depending on the operational context. There is little or no coordination among different groups of actors and – typically – a sudden, unexpected appearance of a new group on the crisis scenario may impose an on-the-fly rearrangement of the assigned resources.

Clearly, this context requires a security mechanism for regulating access to sensitive resources (e.g. spectrum bands). The deployment of DSA may have a large number of operating dimensions including frequencies, waveforms, power levels, and so forth. There is the need to define an access control framework, which allows to benefit of the DSA features while ensuring the conformance to regulatory policies and rules of conduct among Public Safety organizations.

Mainstream approaches to access control do not seem to be suited for a complex environment like the one sketched above. In fact, resources' access to requesting entities is enforced by a centralized authority, given an a-priori fixed set of rules describing what are the resources and under what conditions they can be accessed. Requesting entities are usually identified through a standard login-password online mechanism. Other more sophisticated off-line authentication mechanisms require the presence of heavyweight, centralized and rather inflexible infrastructures, like PKIs.

7.2 A brief overview of spectrum management in Cognitive radio network

The design and deployment of Cognitive Radio and DSA has been investigated in a number of papers and research studies starting from the seminal work of John Mitola in [53]. A survey is presented in [4] by Akuldiz ed al., where the authors describe the various cognitive radio techniques and architectures to implement DSA and Cognitive radio networks. The paper presents the various classifications of spectrum sharing techniques including the one based on access behavior, where spectrum sharing can be cooperative or non-cooperative. In the cooperative approach, the cognitive radio nodes consider the effect of their own communication on other nodes and they cooperate to minimize the level of interference and to optimize spectrum utilization. In the non-cooperative or selfish approach only a node is considered. In this paper, we will consider only the cooperative approach.

The application of cognitive radio and DSA to the Public Safety domain is investigated in the SDR Forum Technical Report [96]. The document describes the benefits and the related challenges of the deployment of these new technologies. A major challenge is to ensure that DSA can provide the same level of security and reliability of conventional communication systems. The definition of requirements for the application of cognitive radio in emergency network is provided in [82], which also describes a protocol for the exchange of control message for the use of the radio resources. In [112], the authors presented a framework for the use of cognitive radio in the public safety domain. The paper presented a workflow for the dynamic allocation of the spectrum and a protocol for exchanging spectrum resources among the actors involved in the scenario. In [11], the authors describe the technical and psychological challenges for resources management and spectrum sharing across different public safety organizations. The paper highlights the importance of creating control mechanisms and a trust framework to overcome the challenges and improve the efficiency of spectrum utilization. The paper does not identify of describe a specific trust or policy framework but identifies the benefits of adopting such framework.

The cooperative approach for spectrum sharing requires the definition of a policy language to regulate the sharing of the resources among the cognitive radio nodes. A policy framework for DSA has been defined in the DARPA XG program. The *neXt Generation program* (XG) is a technology development project sponsored by the US DARPA's Strategic Technology Office, with the goals to develop both the enabling technologies and system concepts to dynamically redistribute allocated spectrum. XG uses a declarative policy engine that supports spectrum sharing while ensuring that radios will adhere to regulatory policies and it is able to adapt to changes in policies, applications, and radio technology. The policy engine is based on a declarative language called Cognitive Radio Language (CoRaL) for expressing spectrum sharing policies (see [39]). In CoRal, policy rules such as allow (permissive), disallow (restrictive) are logical axioms that express under which conditions these predicates hold. The policy rules may consider the radios capability, current state, location, time, and spectral environment for allowing a transmission. The design of a Policy Reasoner based on CoRaL language is presented in [38]. The paper describes the demonstration of the XG technology, CoRaL language and the Policy Reasoner in a testing scenario where CoRaL policies are used to change how XG radios access spectrum resources, based on the location of the radio, its operational mode and the sensed signal strengths. Reference [98] from SDR Forum is one of the first documents, which identifies and describes a modeling language to negotiate and control the network resources in the public safety domain. The modeling language is called MetaLanguage for Mobility (MLM) and it is used to describe the functions, resource and roles of the elements and actors participating in the operational scenarios related to the Public Safety domain. The reference presents a specific scenario for spectrum sharing. MLM is based on OWL ontology language and the use cases are described using UML. The language does not have security elements to define various levels of authority or trust among the actors as the sharing of network resources can be based on a pre-defined agreement among organizations.

The shortcoming of the previous papers is that the presented policy languages are not specifically designed for describing operational contexts where users have different priorities and capabilities, when deploying spectrum resources. Public Safety operational scenarios are characterized by many organizations with different levels of authority and priority in the access to the available resources in the scenario (i.e. energy, water or communications). Generally military organizations have the highest authority, then police and volunteers organizations. The priority depends on the operational scenario, as well. A suitable policy language should have– among other things – the capability of describing the different levels of priority in using the spectrum resources on the basis of the type of operational organization and the type of operational scenario.

One of the first papers to address the challenge of defining a DSA model in a context with multiple organizations with various levels of authority is [40]. The paper presents a multi-organizational policy management system for DSM based on the fine-grained control of delegation of authority between communities of users. The contribution is identified as an extension of the XG policy engine but with a clear focus on the management of different levels of authority. Reference [63] addresses the management of DSA in a holistic manner. The paper describes a meta-policy framework that includes the definition of the hierarchal structures of the organizations involved in DSA scenarios. However, the issue of defining and enforcing access policies to network resources – given the above mentioned hierarchies – is not addressed.

7.3 Introducing Trust-$\mathcal{X}$ in Cognitive radio networks

The previously described scenario requires a flexible access control mechanism. The Trust-$\mathcal{X}$ framework and in particular the $\mathcal{X}$-RNL language – described in Chapter 3 – address such requirements.

Example 7.3.1. *As an example, consider the following negotiation:*

1. *A fireman ($\mathcal{F}$) asks to access a credential containing the positions of the gas pipes (GasBluePrint) which belongs to the LondonGasSociety ($\mathcal{LGS}$).*
2. *$\mathcal{LGS}$ replies with the disclosure policy $GasBluePrint \leftarrow ID(Country = UK) \wedge FiremanID \wedge FireBrigateID$.*
3. *All the required credentials are available but the credentials $FiremanID$ and $BrigateID$ are considered sensitive credentials. Hence, they are protected by disclosure policies too. Therefore, $\mathcal{F}$ sends the disclosure policies $FiremanID \leftarrow ID(Country = UK)$ and $FireBrigateID \leftarrow ID(Country = UK)$.*
4. *$\mathcal{LGS}$ owns a credential attesting its identity and is freely available. It sends it to $\mathcal{F}$.*
5. *$\mathcal{F}$, upon the verification of the validity of the credential received, sends the required credentials ID, $FiremanID$ and $FireBrigateID$ to $\mathcal{LGS}$.*
6. *Finally, $\mathcal{F}$ is able to access the credential $GasBluePrint$ disclosed by $\mathcal{LGS}$.*

The messages exchanged are shown in Figure 7.1.

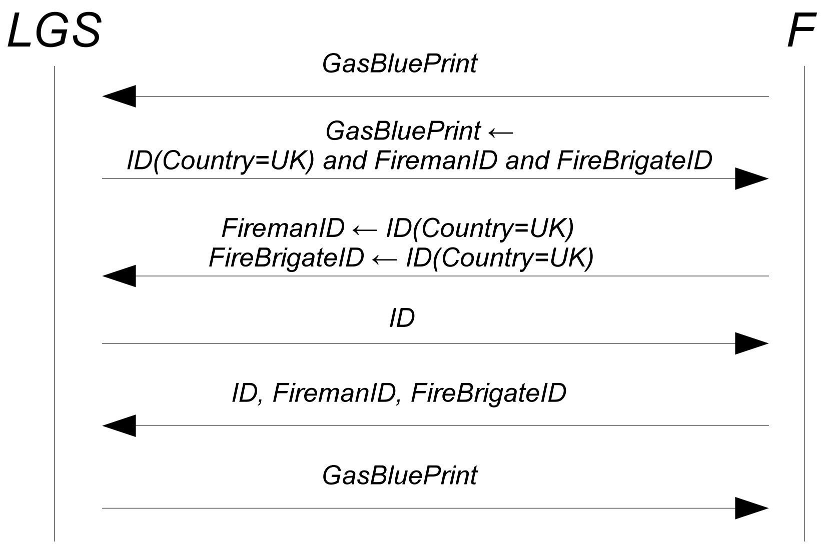

LGS
F
GasBluePrint
GasBluePrint ←
ID(Country=UK) and FiremanID and FireBrigateID
FiremanID ← ID(Country=UK)
FireBrigateID ← ID(Country=UK)
ID
ID, FiremanID, FireBrigateID
GasBluePrint

Figure 7.1: An example of trust negotiation

7.3.1 The Spectrum Management Language

The trust negotiation language presented do not suffices for expressing all the complex setup procedures required by parties communicating over a CR networks. Towards this end, we extend our negotiation language illustrating how to include a spectrum management language largely inspired by the CoRal language [39] into our framework.

Definition 7.3.1 (Condition term). *A* condition term CT *is an expression of the form* $Condition(PredList)$ *where Condition denotes a condition type, such as* Time, Location, DeviceCapability *and* NodeIdentity, *and* $PredList$ *is the same list of attribute conditions presented in Definition 3.4.1.*

Note that the possible attributes in the $PredList$ of a condition term depends on the type of condition represented.

Examples of condition terms are:

- *Location(Latitude="51.30 N", Longitude="0.30 W")* and
- *Time(localtime≥10:00, localTime≤17:00).*

A *frequency list* is an ordered list of frequencies, such as {*3847MHz, 3990MHz, 4375MHz*}.

A *spectrum management policy* is an expression of the form $freqList \leftarrow CT_1 \wedge CT_2 \ldots \wedge TC_v$ where $freqList$ is a frequency list and CT_i are condition terms.

Moreover, we classify the spectrum management policies in two categories: *permissive policies* and *restrictive policies*.

In each instant, the frequencies used by a terminal are determined as follows:

1. Identify all the allowed frequencies, which entails the identification of the permissive policies whose right side is true;
2. Identify all the prohibited frequencies, which entails the identification of the restrictive policies whose right side is true;
3. Finally, a terminal is allowed to transmit on the difference between allowed frequencies and prohibited frequencies.

Example 7.3.2. *For example, consider the following permissive policy:*

$$\{5132MHz, 231.2250MHz\} \leftarrow Location(City = London) \wedge Time(hour \geq 08)$$

and the following restrictive policy:

$$\{5132MHz\} \leftarrow Time(hour \geq 22)$$

Applying such policies at the 11:00PM, a terminal located in London will be allowed to transmit on the frequency of 231.2250MHz.

The spectrum management policy language described above is an example. It is possible to extend the language to achieve the same expressive power of [39, 38] but is behind the objective of the current work.

7.4 Motivating scenario

We now apply the negotiation language presented above in a very well known crisis scenario: the London Underground bombing of July 2005 and the subsequently deployed resolution efforts [45].

We have chosen this specific operational scenario because it illustrates the significant challenges in resolving an emergency crisis in a urban environment where a large number of users (i.e. public safety organizations and civil population) were present and communication resources were overused.

Urban environment scenarios are usually characterized by the need for fast reaction time by the first time responders and consequently heavy reliance on communications to cooperate the recovery efforts. In this specific case, the existing communication resources were particularly strained because of the high volume of traffic due to panic conditions and the degradation of some network infrastructures due to the bombing.

A major challenge was that the traffic demand exceeded the capacity of the network and access control mechanisms were used to deny access to some users, including first time responders, who did not have priority access.

The aftermaths of the three explosions on the trains of the London Underground posed particularly harsh conditions to the first responders. Bombs exploded on underground trains having varying distances to nearest stations. Power lines were cut off. Responders had to walk to the bombing scenes through tunnels. Once police and fire responders went into the tunnels, their radios lost connectivity to the above-ground infrastructure. The only means for responders to communicate back to their respective command centers and any above ground personnel was to walk to the nearest station and position themselves at the entrance to the Underground system. It took 15 minutes to walk from the scene to the entrance. Though responders had adequate authority to communicate on their own networks, individual radios were not capable of exploiting peer-to-peer capabilities to provide network extension to connect isolated nodes to the network [97].

The agencies that responded to the emergency included the Metropolitan Police, the British Transport Police, the London Fire Brigade and the London Ambulance. Clearly, the poor management of communication channels resulted in a lack of coordination – and hence a degraded service – among different groups and even within single groups.

7.5 Negotiating Resources in a Critical Environment

It is well understood that in crisis scenarios DSA would provide more effective communication means through its improved spectrum (and, more in general, network resources) utilization. We argue that in order to fully exploit the potential of DSA in critical scenarios is necessary to clearly define suitable access policies to sensitive resources and – consequently – provide mechanisms for their enforcement. Towards this end, we now show how effective resource disclosure policies related to the London Underground bombing scenario can be expressed using the Trust-$\mathcal{X}$ framework.

7.5.1 Roles identification and Role Hierarchy

First of all, in order to be able to achieve a coordinated and adaptive spectrum frequency management, we need to define different roles in which the users operate and a hierarchy among them.

With respect to the scenario described in the previous section, we define the following roles:

1. Government
2. Military, such as the UK Army;
3. Public safety, such as the metropolitan Police and the London Fire Brigade;
4. Paramedic, such as the London Ambulance.

In the analyzed scenario there are few roles, hence is possible to define a simple hierarchy among these roles. Such hierarchy defines that, for example, a cognitive radio used by an intelligence officer, a user belonging to the Military role, must have a priority higher than a doctor or a user belonging to the Paramedic group.

7.5.2 Cognitive Radio Authentication

Once defined a role hierarchy, we present how a radios arrived on the incident area can authenticate themselves with the existing ones to properly operate in the Incident Area Network (IAN). We assume that the different entities participating in the crisis response are joined into separate groups, at least one for each role defined above, which denote different deployment modes of the available communication resources, e.g. bandwidth.

In order to communicate in a given spectrum band, a newly arrived cognitive radio joins the appropriate group. The group joining process is carried on by the newly arrived radio and an IAN coordinator through a trust negotiation. If the trust negotiation ends successfully, the newly arrived radio receives a group membership credential. We remind that a credential is a digital document attesting some properties (stated as tuples of pairs (Attribute, Value)) about the owner. We also remind that the disclosure of each credential is controlled by a disclosure policy, as described in Chapter 3 and in Section 7.3.1.

The group membership credential contains different information. In particular, it contains the spectrum management policies, which greatly differ depending on the role of the owner.

The disclosure policy 7.1 shows the requirements associated with the credential containing the configuration parameters to access the IAN at "King's Cross-Russell Square".

$$\begin{aligned} &IAN(Location = \text{``}King'sCross - RussellSquare\text{''}) \leftarrow \\ &\quad Military(Country = \text{``}UK\text{''}) \vee \\ &\quad (ID(Country = \text{``}UK\text{''}) \wedge \\ &\quad PoliceID(Affiliation = \text{``}LondonPoliceDepartment\text{''})) \\ &\quad \vee\ FireDepartment() \vee \\ &\quad (Paramedic()\ \wedge\ DoctorSpecialization()) \end{aligned} \tag{7.1}$$

We will now present two possible scenarios which differentiate on the type of actor which wants to communicate on the network.

Let's assume the first cognitive radio belongs to a policeman replying to the initial requesting call. His radio will reply to the disclosure policy 7.1 requiring some other credential in order to disclose the credential attesting its membership in the London City Police Department.

$$\begin{aligned} &PoliceID() \leftarrow (ID(Country = \text{``}UK\text{''}) \wedge \\ &\quad GovernmentAuthorization(Country = \text{``}UK\text{''})) \vee \\ &\quad Military(Country = \text{``}UK\text{''}) \end{aligned} \tag{7.2}$$

The coordinator of the IAN has the appropriate government authorization and it has an ID released from the United Kingdom. The ID can be freely disclosed, therefore it has no disclosure policy associated. On the other side, the access to the government authorization credential is restricted to some other property related to UK nationality.

$$GovernmentAuthorization \leftarrow ID(Country = \text{``}UK\text{''}); \tag{7.3}$$

Hence, the negotiation terminates successfully, allowing the policeman to communicate in the IAN with the spectrum management policies of the Public Safety role.

Let us suppose that the next cognitive radio arriving on the incident area belongs to a paramedic. As in the previous example, his radio will try to obtain the credential required to configure itself appropriately in order to be able to communicate with the other devices in the network. The disclosure policy is still 7.1. This time, the cognitive radio will try to access as a paramedic, allowing the disclosure of the appropriate credentials against the satisfaction of the corresponding disclosure policies, namely 7.4 and 7.5.

$$\begin{aligned} &Paramedic() \leftarrow \\ &\quad PublicHealthServiceAuthorization(Country = \text{``}UK\text{''}) \\ &\quad \wedge\ GovernmentAuthorization(Country = \text{``}UK\text{''}) \end{aligned} \tag{7.4}$$

$$DoctorSpecialization() \leftarrow ID(Country = \text{“}UK\text{”}) \tag{7.5}$$

Again, the paramedic successfully joins the IAN, receiving the group membership credential with the appropriate spectrum management policies for the role Paramedic.

7.5.3 Adaptive Spectrum Frequencies Management

Radio frequency spectrum management can be based on centralized or distributed architecture. In the centralized architecture, a central authority has the task to allocate the spectrum resources to the nodes of the cognitive radio networks. In a distributed architecture, the various nodes must negotiate the allocation of the spectrum frequencies on the basis of defined policies. A centralized architecture is usually preferred in public safety, but it may not be possible in operational scenarios where the communication infrastructure is degraded or destroyed or a central authority is missing. This is often the case in the initial phases of the resolution of an emergency crisis. For these reasons, we are going to describe the case of a distributed architecture where a central authority is missing. In what follows, we will present, through an example, two protocols for the management of the spectrum band, with respect to the role hierarchy introduced in Section 7.5.1 and enforced by means of the authentication procedure described in Section 7.5.2. We will assume the absence of a central authority.

Higher Role Enters and Leaves

The first example we present is about the arrival of a cognitive radio belonging to a high-level role, for example a military radio.

The military entity requires radio communication with high quality of service and resilience of the communications. As a consequence, the military radio must negotiate, through a well defined protocol, the necessary spectrum resources with other nodes in the network. Suppose the roles are ordered and that each role R_i is assigned to a number r_i. $r_1 > r_2$ means that r_1 has a higher priority then r_2 and with r_n the maximum value. The proposed protocol is the following:

When a user has to renegotiate the spectrum band allocated, it is forced to leave or accept a lower quality of spectrum band. The decision about how the bandwidth cut-off from each user in a given role r_j is demanded to the policies defined in the radio, by means of the group membership credentials. A reasonable requirement is that the objective is to retrieve enough spectrum band to let the new higher role radio to properly operate within the network. The modifications of the spectrum band used by each radio are performed creating additional restrictive rules.

A related example is about how to handle the spectrum band associated with a radio leaving the IAN. Different strategies are possible and we propose to reassign the bandwidth among the existing groups by means of the following procedure: A leaving radio notifies its role to the other radios. If no other radio is operating in

```
Data: $u_i$, the user who joint the group; $r_i$, the role assigned to $u_i$
begin
    if not enough bandwidth for $r_i$ requirements then
        for $j \in [r_n, r_{n-1}, \ldots, r_{i+1}]$ do
            for $u_k \in r_j$ do
                notify $u_k$ to re-negotiate the level;
                if enough bandwidth for $r_i$ requirements then
                    return
```

Algorithm 6: Protocol for finding enough spectrum band for a given user u_i of role r_i

the same role, no answer is received. In such a case, the leaving radio notifies the other radios of the IAN which initiate a negotiation to gain more spectrum band, as stated in their initial role configuration, with respect to the requirements of higher roles. The procedure followed by the remaining radio is presented by Algorithm 7.

```
Data: $u_i$, the leaving radio the group; $r_i$, the role assigned to $u_i$, $F_i$, the
      spectrum frequencies currently assigned to the role $r_i$
begin
    for $j \in [r_{i+1}, r_{i+2}, \ldots, r_{n-1}, r_n]$ do
        $F_j$ := spectrum frequencies currently assigned to role $r_j$;
        $carF_j$ := number of frequencies in $F_j$;
        $defF_j$ := default spectrum frequencies required by role $r_j$;
        $carDef_j$ := number of frequencies in $defF_j$;
        if $carF_j < carDef_j$ then
            extract from $F_i$ $n$ frequencies, where $n$ is the minimum between
            $(carF_j - carDef_j)$ and the number of frequencies in $F_i$;
            assign the $n$ frequencies chosen to $F_j$;
            remove the $n$ frequencies chosen from $F_i$;
            if $F_i$ is empty then
                return
```

Algorithm 7: Protocol for distribute the spectrum band released by leaving of the radio u_i, last representative of role r_i

7.6 Experimental Results

We performed some experiment to evaluate the approach proposed in the current scenario. Namely, we extended and adapted the Trust-$\mathcal{X}$ prototype, presented in Appendix B, to provides the features described before and to better fit the scenario.

To run our experiments we used a network of two computers with the following configurations:

- Linux, kernel 2.6.30, CPU 2.20GHz
- Macbook, OS 10.6, CPU 2.53GHz

In order to have a more realistic feedback from our experiments, we run them twice on two different lightweight DBMSs, namely SQLite and MySQL. Such DBMSs are deployed for the storage of credentials and disclosure policies.

First of all, we evaluated the time required by a new device to authenticate itself in the IAN with respect to the number of credentials that have to be exchanged.

Figure 7.2 shows how Trust-$\mathcal{X}$ performances are linear to the number of credentials. More precisely, its performance depends on the structure of the disclosure policies exchanged. The simpler negotiation, which involved the exchange of two credentials, represented by a negotiation of the form $A \leftarrow B$, it required in average 226 milliseconds, with a lower bound of 188 milliseconds. On the other hand, to negotiate and exchange 50 credentials Trust-$\mathcal{X}$ required 3859 milliseconds.

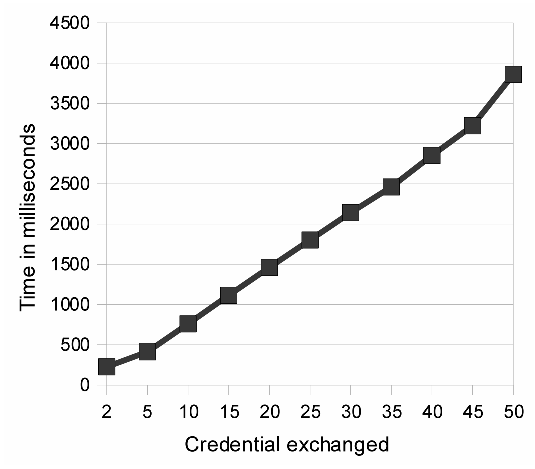


Figure 7.2: Time required to authenticate with respect to the number of credentials involved

According to the presented results, the negotiation illustrated in Section 7.5.2 required in average 265 milliseconds.

In another group of simulations, we evaluated the performance of the negotiation protocol in different environments characterized by various sizes of the population of CR nodes and various levels of mobility. With the term mobility, we mean the rate of topological changes of the CR network, for example, the number of CR nodes leaving or entering the scenario in a specific time. This is an important parameter on which evaluate the performance of a CR network. Public Safety operational scenarios may be characterized by an high degree of mobility as new public safety organizations appear or disappear from the context, radio links are degraded by natural or man-made obstacles or because one or more CR nodes suffer from technical failure of power exhaustion.

Because operational requirements impose specific timing constraints on the allocation of spectrum resources, the negotiation protocol should not introduce large delays in presence of an elevated mobility of the CR network. Because of this reason, we evaluated the negotiation protocol again different populations of CR nodes, with sizes ranging from 100 to 500 nodes. Figure 7.3 shows that the time required is linear with the cardinality of the CR nodes in the network. The overall time used by the negotiation protocol is still limited to few seconds even for networks of large size (500 nodes). Such values are comparable to the timing constraints defined by public safety operational requirements as in [89].

In the simulation we also introduced the delay of the communication of the information itself. The lines in Figure 7.3 show different levels of mobility. Because the divergence of the lines is small in comparison to the overall time, another graph was created to highlight the time difference for various levels of mobility from the best case of a complete static CR network. In Figure 7.4 the line on the X axis represents the static case, while the other curves represents increasing levels of mobility. From the graph, we can see that even for high levels of mobility (50 CR nodes per second) the difference from the static case is only of few hundreds milliseconds. This means that the negotiation protocol described in this paper is minimally influenced by the level of mobility of the CR network. For large sizes of the CR network, all lines are converging to the same value, as the percentage of the CR nodes moving in or out the network is small when compared to the overall size of the CR network. We can conclude that the rate of mobility of the nodes does not heavily influence the performances of the negotiations.

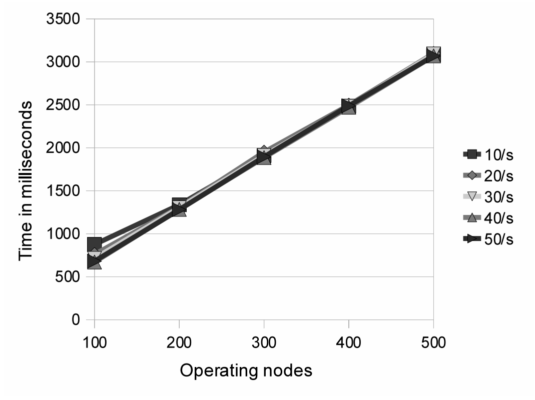


Figure 7.3: Stability of the approach with respect of interferences

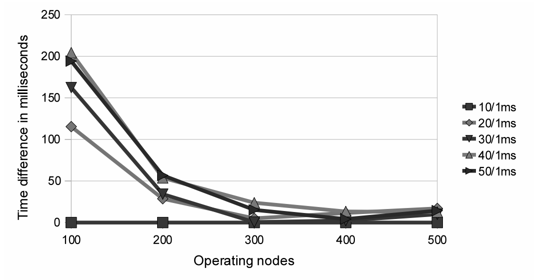


Figure 7.4: Stability of the approach with respect to mobility

Chapter 8

TN in On-line Social Network

8.1 A brief introduction to OSN access control

In our last application scenario, we applied the trust negotiation techniques previously presented extending an on-line social network access control mechanisms.

Similarly to other networks, an OSN SN can be represented as a labeled graph, where each node denotes a user in the network, whereas edges represent the existing relationships between users, and their trust levels. Edge direction denotes which node specified the relationship and the node for which the relationship has been specified, whereas the label associated with each edge denotes the type of the relationship. Figure 8.1 shows an example of OSN graph.

The number and type of supported relationships depend on the specific OSN and its purposes; our only assumption is that there exists at least one relationship type. We also assume that, if RT denotes the set of supported relationship types, given two nodes $A, B \in SN$, there may exist at most $|RT|$ edges from A to B (from B to A, respectively), all labeled with distinct relationship types. We can now formally define OSNs as follows.

Definition 8.1.1 (OSN). *[28] An OSN SN is a tuple* $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$*, where* RT_{SN} *is the set of supported relationship types,* V_{SN} *and* $E_{SN} \subseteq V_{SN} \times V_{SN} \times RT_{SN}$ *are, respectively, the nodes and edges of a directed labeled graph* $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$*, whereas* $\phi_{SN} : E_{SN} \to [0, 1]$ *is a function assigning to each edge* E_{SN} *a trust level t, which is a rational number in the range* $[0, 1]$*.*

An edge $e = vv' \in E_{SN}$ expresses that node v has established a relationship of a given type $rt_e \in RT_{SN}$ with node v'.

In this chapter, we assume that OSN resources are protected according to the model presented in [28]. According to this model, access control requirements of OSN users are expressed in terms of *access rules* specified by resource owners. Access rules denote authorized members in terms of the type, maximum depth and minimum trust level of the relationships they must have with other network nodes. Such constraints are expressed as a set of *access conditions* $(v, rt, d_{max}, t_{min})$, where

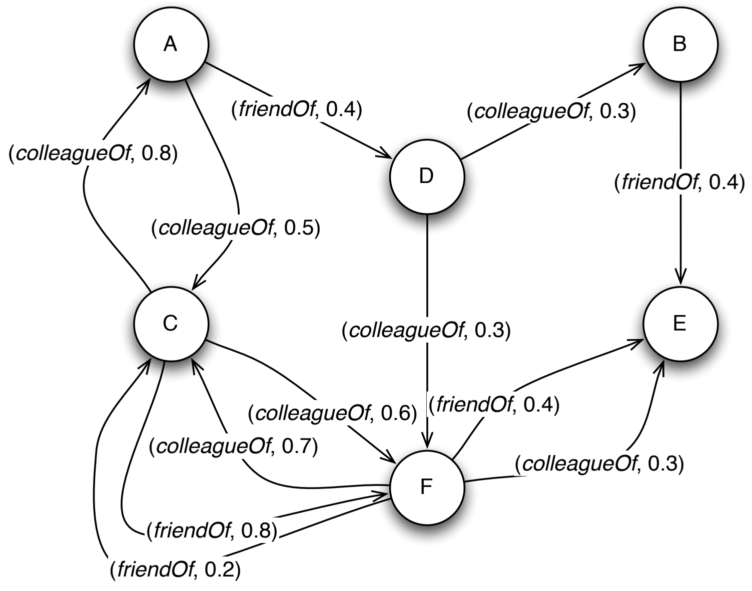


Figure 8.1: An example of OSN labeled graph

v is the node with which the requesting node must have a relationship of type rt, whereas d_{max} and t_{min} are, respectively, the maximum depth and the minimum trust level that the relationship must have. If $v = *$ and/or $rt = *$, v corresponds to any node in the OSN and/or rt corresponds to any supported relationship type. Whereas if $d_{max} = *$ and/or $t_{min} = *$, there is no constraint concerning the depth and/or trust level, respectively.

Example 8.1.1. *Referring to the OSN in Figure 8.1, suppose that Alice (A) holds a resource r that she wants to share only with her colleagues or the colleagues of their colleagues no matter of their trust level. She can encode such requirement by the following access condition* $(A, colleagueOf, 2, *)$*. Moreover, if a resource is protected by the following access rule* $\{(A, friendOf, 1, 1)\}$*, it means that it can be accessed only by A more trusted direct friends.*

In [28], access control is client-based, according to which the requester must provide the resource owner with a proof of being authorized to access the requested resource. As access rules constraint relationships, the proof has to show the existence of a path in the network satisfying the constraints on depth and trust level imposed by the rule. In order to generate valid proofs, it is assumed that a "relationship certificate" is associated with each relationship, containing information on the relationship (i.e., users involved, trust, depth, type), which is signed by both the involved users. A relationship certificate can be seen as a proof that between the involved users there exists a direct relationship of a certain type and with a certain trust level. Proofs of indirect relationships can therefore be generated through a set of certificates confirming the existence of a path of a specified type between them. Certificates are managed by a trusted authority *CS* which is in charge of path retrieval and delivering to a resource requester.

More details can be found in [28], here we just briefly recall the access control protocol, which is graphically depicted in Figure 8.2[1], taken from [28], since it is required in the following of the paper. The explanation of the protocol is provided in Figure 8.3.

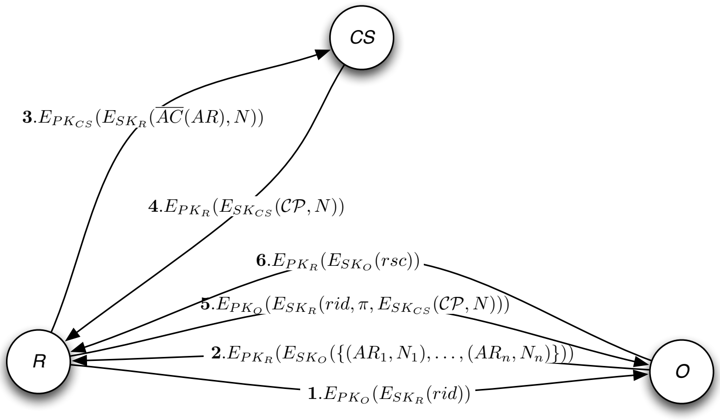

CS
$\mathbf{3}.E_{PK_{CS}}(E_{SK_R}(\overline{AC}(AR),N))$
$\mathbf{4}.E_{PK_R}(E_{SK_{CS}}(\mathcal{CP},N))$
$\mathbf{6}.E_{PK_R}(E_{SK_O}(rsc))$
$\mathbf{5}.E_{PK_O}(E_{SK_R}(rid,\pi,E_{SK_{CS}}(\mathcal{CP},N)))$
R
$\mathbf{2}.E_{PK_R}(E_{SK_O}(\{(AR_1,N_1),\ldots,(AR_n,N_n)\}))$
O
$\mathbf{1}.E_{PK_O}(E_{SK_R}(rid))$

Figure 8.2: Access control protocol. *R* is the node requesting a resource with identifier *rid*, *O* is the node owning the resource, whereas *CS* is the certificate server.

8.2 Extension of the access control rule definition language

In order to introduce trust negotiations in the framework presented in [28], we need to first extend the language to specify access rules adapting – when required – the definitions of Chapter 3. More precisely, we aim at extending the access rule language in order to allow the inclusion of resources – or credentials – as conditions in access rules.

Borrowing the definitions of resource conditions from Chapter 3, we are able to extend the definition of access rules given in [28] (cfr. Section 8.1) to support both resource and access conditions.

Definition 8.2.1 (Access rule). *An access rule* `AR` *is a tuple* (rid, AC) *where rid is the identifier of resource rsc, whereas AC is a set of access and resource conditions. Such set AC expresses the requirements a node must satisfy in order to be allowed to access resource rsc.*

[1]In the figure, $E_k(d)$ denotes the encryption of d with key k.

Resources associated with an access rule *ar* with $AC(ar) = \emptyset$ are always accessible. Such resources are fundamental for the successful execution of trust negotiations.

Definition 8.2.2 (Deliverable resource)**.** *We define a resource rsc identified by an identifier rid as* deliverable resource *– DELIV for short – if and only if, for each access rule ar with* $rid(ar) = rid$, $AC(ar) = \emptyset$.

With the introduction of resource conditions, we need to modify the protocol described in Figure 8.3. More precisely, if the access rule `AR` exchanged in Step 2 contains a resource condition *rc*, the current access control procedure must temporarily pause in order to perform the disclosure of the resource $rn(rc)$. This is achieved by extending the protocol in such a way that it takes advantages of the features described in Chapter 3. The procedure is described in Figure 8.4.

Example 8.2.1. *Suppose that Bob (B) asks to Alice (A) resource rsc, with identifier rid. A defined for rsc the following access rule:*

$$(rid, \{(rsc', \{(att_1, =, 5), (att_2, <, 3)\}), (rsc'', \emptyset)\})$$

stating that, for the disclosure of rsc it is required that the requester provides a copy of the resource rsc'*, having attribute* $attr_1$ *equal to 5 and attribute* $attr_2$ *less then 3, and a copy of the resource* rsc''.

Also suppose that B owns resource rsc'*, with identifier* rid' *and that the corresponding access rule is:*

$$(rid', \{(B, friendOf, 4, 2)\})$$

stating that, for the disclosure of rsc' *B requires the existence of a path between him and the requester, labeled with the relationship* friendOf *with maximum depth 4 and having trust value greater then or equal to 2. Moreover, the access rule defined by B for resource* rsc'' *is:*

$$(rid'', \{(rsc''', \emptyset)\})$$

For simplicity, suppose that resource rsc''' *is deliverable.*

Therefore, if A is able to proof the path required by the above access rule, then B will release rsc'*. After having verified the validity of resource* rsc'*, finally, A will release the resource rsc to B.*

The negotiation tree built during the trust negotiation is presented in Figure 8.6.

8.3 Dynamic relationships and trust level adjustment

In a realistic setting, the outcome of a negotiation between two nodes in an OSN may influence the trust level occurring between them. This may occur in a rather natural way: namely, in the case the negotiation is successful, the trust level increases, while, in the opposite case, the trust level is bound to decrease. Our aim in this section is to show how to extend the model described in Section 8.1 in such a way that negotiation outcomes influence trust levels of the participants in such negotiations. This is achieved by adding some private relationships to the OSN graph, whose trust level is adjusted on the basis of the negotiations performed so far. Such relationships can then be exploited to speed up subsequent access requests.

8.3.1 Trust computation

Since the trust relationship between two nodes is asymmetric, the trust level of the first node towards the second and the trust level of the second node towards the first are adjusted in an independent way, by the corresponding nodes. The value of the updated trust level depends on – apart from the positive outcome of a negotiation, of course – the relevance of the resources involved in the negotiation. How a node measures the relevance of its own resources is a far from trivial matter that would deserve a more in-depth investigation than the one presented here. In the following, we limit the presentation on what is strictly needed in the context of the present work.

Definition 8.3.1 (Relevance of a resource)**.** *Given a resource r of a node n protected by an access rule ar, the* relevance $rel_{ar}(r)$ *of resource r given access rule ar is a numeric value computed by aggregating the relevances of the access conditions contained in the access rule ar. The aggregation function used for this purpose can be chosen from $\{sum, max, min\}$.*

With respect to a resource protected by a set of access rules, our approach is quite similar: suppose a resource r is protected by a set AR of access rules and that with each access rule $ar_i \in AR$ there is associated a corresponding relevance value $rel_{ar_i}(r)$. Thus, the relevance $rel_{AR}(r)$ of resource r with respect to the access rules set AR is given by aggregating the relevances $rel_{ar_i}(r)$ using as aggregating function a function in $\{sum, avg, max\}$, analogously to what has been done for computing resource's relevance with respect to a single access rule.

In the case that a resource r is not protected by any access rule, the relevance of r is decided by the resource owner by simply assigning it a (possibly normalized) numeric value.

Taking advantage of the concept of relevance of a resource, it is possible to modify the relationships of a user, and the trust levels associated to such relationships, with respect to the resources granted and accessed. Note that such relationships are local to the user, which means they are unknown to the CS.

If a negotiation for a resource r between two users A and B successfully ends, then each user update her/his trust level relative to the other user. Such update depends both on the relevance of the negotiated resource and the previously existing trust level.

Namely, we propose to update the trust level according to the following equation:

$$\phi(e) = \phi'(e) + Relevance(r) \cdot (1 - \phi'(e))$$

where ϕ is the function described in Definition 8.1.1, e is a the edge representing the relationship whose trust level has to be updated. We denote the value of the function previous to the update with ϕ'. Similarly, if the negotiation fails to successfully end then the trust level is updated to reflect such failure lowering the trust level associated with the relationship. Algorithm 8 presents the procedure to update the trust level of the relationships of the OSN interacting users. Note that, according to our OSN model (cfr. Definition 8.1.1) the trust level of a relationship between two users may depend on the type of the relation (for instance, I can trust a user more as my friend than as my colleague). Therefore, once two nodes end a negotiation for the first time, two *dynamic edges* are created between the two, of type *disclosedTo* and *receivedFrom*, respectively to which the dynamic computed trust level is assigned.

The variations of the trust levels (a variation for each user) depend – apart from the negotiation outcome – on the current trust levels of the negotiating users, in such a way that the higher the trust levels, the lesser the variations. In other words, if the users trust each other with an high degree, it will take a negotiation involving highly relevant resources to significantly modify the corresponding trust levels.

```
Data: $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, the social network graph; $rt_O, rt_R \in RT_{SN}$, the
      relationship type for dynamic edges, respectively for the resource
      owner and for the requester
Input: $R$ requester id, $O$ owner id, $out \in \{-1, 1\}$ result of the negotiation, $r$
       the resource requested by $R$

begin
    if the edge $e = (O, R, rt)$ does not exist in $E_{SN}$ then
        Create $e_O = (O, R, rt)$;
        $E_{SN} = E_{SN} \cup \{e_O\}$;
    Update $\phi_{SN}$ such that
    $\phi_{SN}(e_O) = \phi'_{SN}(e_O) + (out \cdot \texttt{Relevance}(r) \cdot (1 - \phi'_{SN}(e_O)))$;
    if the edge $e_R = (R, O, rt)$ does not exist in $E_{SN}$ then
        Create $e_R = (R, O, rt)$;
        $E_{SN} = E_{SN} \cup \{e_R\}$;
    Update $\phi_{SN}$ such that
    $\phi_{SN}(e_R) = \phi'_{SN}(e_R) + (out \cdot \texttt{Relevance}(r) \cdot (1 - \phi'_{SN}(e_R)))$;
```

Algorithm 8: Updating of the social network graph's edges and the associated trust levels

We show our approach through the following example.

Example 8.3.1. *Consider the scenario described in Example 8.2.1 and let us suppose Alice (A) and Bob (B) never interacted before and that the access control procedure ends successfully. Let us also suppose that the resource r has relevance* 0.5 *for A and* 0.3 *for B.*

After the successful ending of the access control procedure two new edges are added to the OSN graph. The first one, from A to B, labeled with the relationship type disclosedTo *and the second one, from B to A, labeled with the relationship type* receivedFrom. *The trust level associated with* $e = (A, B, \text{disclosedTo})$ *is computed as follows:*

$$\phi(e) = \phi'(e) + (Rel(r) \cdot (1 - \phi'(e)) = 0 + (0.5 \cdot 1) = 0.5$$

On other hand, the trust level associated with $e' = (B, A, \text{receivedFrom})$ *is computed as follows:*

$$\phi(e') = \phi'(e') + (Rel(r) \cdot (1 - \phi'(e')) = 0 + (0.3 \cdot 1) = 0.3$$

Consider, instead, a subsequent scenario in which B requests a resource r' *to A. Let us suppose that such resource has relevance* 0.3 *for both users and that the access control protocol fails. In such a case, the trust level associated with the edges e and* e' *previously introduced is modified as follows:*

$$\phi(e) = \phi'(e) - (Rel(r') \cdot (1 - \phi'(e)) = 0.5 - (0.3 \cdot 0.5) = 0.35$$

$$\phi(e') = \phi'(e') + (Rel(r') \cdot (1 - \phi'(e')) = 0.3 - (0.3 \cdot 0.7) = 0.09$$

8.3.2 Practical usage of dynamic relationships

The considered access control model does not limit the number of access rules that may be associated with a given resource, and this may result in an increase of the time required to process an access request. In this section, we show how it is possible to take advantage of the relationships introduced as a result of a negotiation to speed up subsequent access requests. This is achieved by creating alternative access rules, which exploit the relationship *disclosedTo* as a shortcut in the access control procedure.

Let us explain how this works by means of an example.

Example 8.3.2. *Suppose that Bob (B) wants to access the resource rsc which belongs to Alice (A). Also suppose that the access rule defined by Alice is:*

$$(rid, \{(A, \text{friendOf}, 3, 0.6), (rid', \{(a = 5)\})\})$$

where rid is the resource identifier of rsc, and rid' *is the identifier of resource* rsc'. *Note that the resource condition* $(rid', \{(a = 5)\})$ *indicates that, to be satisfied, the resource rsc must have an attribute name a whose value is* 5.

Such access rule states that, if there exists a path between Alice and Bob, labeled by the friendOf *relationship, with maximum depth* 3 *and with a trust level greater than or equal to* 0.6 *and if Bob releases to Alice resource* rsc', *then he can access rsc.*

Let us suppose that Bob accesses resource rsc, thus satisfying the access rule. Because of this, a relationship of type disclosedTo *between Alice and Bob with trust level* 0.7 *is created. Also suppose Alice defined the following access rule for rsc:*

$$(rid, \{(A, \text{disclosedTo}, 1, 0.5)\})$$

After some time, Bob requires again to access resource rsc. After the request from Bob, Alice verifies the existence of the relationship of type disclosedTo *with Bob, having a trust level greater than* 0.5*. Thus, Alice grants access to the resource rsc to Bob without contacting CS and without negotiating for rsc'.*

Clearly an important issue when dealing with the introduced dynamic relationships concerns their lifespan. To avoid misuse of the access rules exploiting such relationships, we associate with each dynamic relationship a time limit after which the relationship is removed from the OSN's graph. Such time limit is user-defined, but it directly depends on the trust level of the relationship, in such a way that the *expiration* of the relationship is obtained as

$$exp_e = tl \cdot \phi_{SN}(e)$$

where e is an edge with $rt(e)$ equal to *disclosedTo* or *receivedFrom*, ϕ_{SN} is the function which associates the trust level with each edge, and tl is the time limit defined by the user $v(e)$ (see Section 8.1). Note that such time limit is renewed after each interaction between the users which modify the trust level associated with the edge.

Example 8.3.3. *Consider, again, the scenario presented in Example 8.3.2. Supposing the time limit tl defined by Alice is 10 days, the relationship identified by the edge* $e = (A, B, disclosedTo)$ *expires after* $10 \cdot \phi_{SN}(e) = 7$ *days.*

Another way to use dynamic relationships is to improve the probability that a user can access a required resource. Consider a user B who requires to access a resource *rsc*. If B has requested the same resource to another user A, in a near past, then she/he would try to request the same resource from the same user A to improve the chances to get it. Hence, the relationship *receivedFrom* acts as a reminder of past interactions with the users of the OSN. Moreover, considering that a resource can, and probably will, be owned by more than a single user in the OSN, requesting such resource from a trusted user would further slightly improve the chances of a successful interaction.

A natural consideration which came out from the presented usage of the relationship types *deliveredTo* and *receivedFrom* is that relationships of such types are, somehow, too general. Indeed, a relationship which refers directly to the negotiated resource would be more significative. However, choosing to keep a fine-grained history of all the transactions between users of the OSN would result in an explosion of private relationships; consider, for example, the number of distinct resource disclosures which took place between two Facebook's users. To address such issue, we propose that the users specify in their preferences a set of relationship types which keep track of the access to specific *key resources*, such as rare or high value ones, or those more frequently accessed. In such a way, it is possible to customize the number of relationships that are inserted in the OSN graph as an outcome of trust negotiations.

8.4 Complexity analysis

We now discuss the complexity of the proposed access control protocol.

As explained in [28], Step 4 (see Figure 8.3) represents the most expensive task of the protocol. More precisely, during such step, given an access rule ar, CS discovers the shortest certificate paths referring to the set of access conditions $AC(ar)$ received by the requester node. This is achieved by exploring the OSN's graph. Such operation requires either $O(V_{SN} + E_{SN})$ or $\Theta(V_{SN} + E_{SN})$ time complexity, depending on the used search technique, for each $ac \in AC(ar)$. However, it is possible to reduce the size of the graph which has to be explored taking advantage of the constraints on the relationship type and depth specified in the access conditions. Hence, in the general case, the time complexity required to evaluate an access condition ac is $O(\sum_{rt \in RT(ac)}(V_{SN_{rt}} + E_{SN_{rt}}))$, where $RT(ac)$ is the set of relationship types specified in the access condition ac and $V_{SN_{rt}}$, $E_{SN_{rt}}$ are, respectively, the sets of nodes and edges of the OSN's graph with relationship type rt. Finally, since an access rule consists of one or more access conditions, evaluating an access rule ar requires $O(\sum_{ac \in AC(ar)} \sum_{rt \in RT(ac)}(V_{SN_{rt}} + E_{SN_{rt}}))$. We refer to [28] for a more detailed discussion about the results here reported.

To analyze the time complexity of the Trust-$\mathcal{X}$ framework we need to analyze the interactions between the negotiating users in each phase composing the negotiation.

During the introductory phase a fixed number of messages are exchanged, depending on which features of the Trust-$\mathcal{X}$ framework are used, therefore the time complexity is constant.

The subsequent phase, the policy evaluation phase, is the most time consuming one, because it is the one in which the policies which compose the negotiation tree are exchanged. The number of messages exchanged is therefore linear with respect to the height of the tree while the size of the message exchanged in each turn i is linear to the number of nodes of the tree at deep i. Hence, given an OSN $(V_{SN}, E_{SN}, RT_{SN}, \phi_{SN})$, the execution of the policy evaluation phase between two nodes $A, B \in V_{SN}$ requires $O(|R_A| + |R_B|)$ messages, where $|R_u|$ is the number of resources owned by node $u \in V_{SN}$.

The credential exchange phase is where the required resources are actually exchanged. As mentioned in Chapter 3.6, the resources which have to be exchanged in order to successfully end the negotiation are the nodes of the selected valid view. Such valid view is a subtree of the negotiation tree constructed during the policy evaluation phase. Hence, the height of the valid view is at most the height of the negotiation tree. Thus, we can state that the credential exchange phase requires at most $O(|R_A| + |R_B|)$ messages.

Globally, a trust negotiation executed between two nodes A and B of an OSN requires at most $2 \cdot O(|R_A| + |R_B|)$.

Considering the composition of the two frameworks, it is possible to state that the overall time complexity to evaluate an access rule *ar* is given by the maximum between the time complexity to evaluate the same access rule purged of resource conditions, and the maximum time complexity to negotiate, in parallel, the previously mentioned resource conditions. Such computational parallelism is ascribable to the fact that the negotiation for each resource condition is independent from the others. Similarly, the proof computed by CS is independent from the negotiations, therefore they can be simultaneously computed.

8.5 Experimental Results

The evaluation of the proposed approach has been preliminary done performing several experiments using a Trust-$\mathcal{X}$ prototype described in Appendix B. The integration with the access control mechanism described in [28] is currently under development. To run our experiments we used the the same experiment environment described in Chapter 7.6. In a social network scenario, the crucial point is the capability of the system to scale. Therefore, we performed a series of tests in order to evaluate the scalability of our prototype with respect to the number of parallel negotiations. To be able to compare the results, we performed a crescent number of parallel negotiations. Each trust negotiation involves the same resources and, therefore, the same access rules. The results are presented in Figure 8.7.

1. R submits to O an access request for resource *rsc*, with identifier *rid*.
2. If the resource is public, access is granted. Otherwise, O returns to R the set of access rules $AR = \{\mathtt{AR}_1, \ldots, \mathtt{AR}_n\}$ regulating the access to *rsc*. With each access rule $\mathtt{AR}_i \in AR$, $i \in [1, n]$, a distinct nonce value $\mathtt{N}_i$ is associated as a session identifier.
3. R chooses from `AR` an access rule *ar* and sends CS the nonce value `N` associated with *ar* and the corresponding condition set $AC(\mathtt{AR})$. More precisely, since the certificate server CS has only to discover the shortest certificate paths referring to the relationships denoted by $AC(\mathtt{AR})$, whereas the requestor is in charge of trust computation, for each $ac \in AC(\mathtt{AR})$, R sends CS a modified version of the corresponding set $AC(\mathtt{AR})$ of access conditions (denoted $AC(\mathtt{AR})$), where the t_{min} component of each $ac \in AC(\mathtt{AR})$ is set to `NULL`.
4. CS returns R the set $\mathcal{CP}$ of shortest certificate paths, if any, related to the relationship constraints expressed by the access conditions in $AC(\mathtt{AR})$, along with the nonce `N` associated with `AR`; otherwise, CS returns a failure message. In the latter case, R goes back to step 3 and chooses another access rule, until CS returns the set $\mathcal{CP}$, if any, or all the access rules have been processed.
5. Based on the certificate paths in $\mathcal{CP}$, R tries to generate a proof π of the existence of a path satisfying *ar*.

   If a proof is not obtained, R goes back to Step 3 and chooses another access rule; otherwise, he/she sends O a message, which contains the resource identifier, the proof π, and the certificate paths obtained from CS. $\mathcal{CP}$ and `N` are kept encrypted with the private key of CS in order to grant their authenticity.
6. O sends R the requested resource in case the proof π is valid and the nonce value `N` corresponds to the correct session identifier. Before granting access to the resource, O can locally check whether the set of assertions used in the proof are actually derived from the received certificate paths in $\mathcal{CP}$, by performing the same steps done by the requestor for proof generation.

Figure 8.3: Description of the access control protocol depicted in Figure 8.2

3.0 R chooses from `AR` an access rule ar.

3.1 For each resource condition $rc \in ar$, R starts a trust negotiation procedure according to the protocol described in Figure 8.5.

3.2 If one of the trust negotiations does not terminate successfully then R returns to Step 3.0.

3.3 R sends CS the nonce value `N` associated with ar and the corresponding condition set $AC(ar)$.

Figure 8.4: Modification to the access control protocol in Figure 2 to support trust negotiations

Let O be the resource owner, R be the resource requester and rc be the resource condition requested by R.

1. O initiates a negotiation tree rooted with $rn(rc)$.
2. O sends R all the access rules ar, where $rn(ar) = rn(rc)$ and add all the access rules to the negotiation tree as leaves of the root. More precisely, each rule is added as an AND-node
3. If there exists a subtree of the negotiation tree consisting only of AND-nodes[a] and with leaves only labeled as $DELIV$ then go to Step 6 else, for each access rule ar received, R sends O the access rule corresponding to the resources in ar. Moreover, R adds the rules to a local copy of the negotiation tree.
4. If there exists a subtree of the negotiation tree consisting only of AND-nodes and with leaves only labeled as $DELIV$, then go to Step 6 else, for each access rule ar received, O sends to R the access rule corresponding to the resources in ar and it updates the negotiation tree adding the rules sent.
5. Go to Step 3
6. The party which found a subtree of the negotiation tree with leaves labeled as $DELIV$ communicates the counterpart to switch to the credential exchange phase using the identified subtree as valid view.
7. The parties iteratively send to the counterpart the resources, one level at a time, beginning from the max level, and according to the owner of the resources.

[a] A node n is defined AND-node if it is part of a multi-edge (see Definition 3.6.1 in Chapter 3.6).

Figure 8.5: Trust negotiation protocol

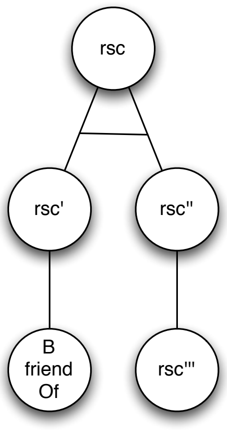


Figure 8.6: Negotiation tree for Example 4.1

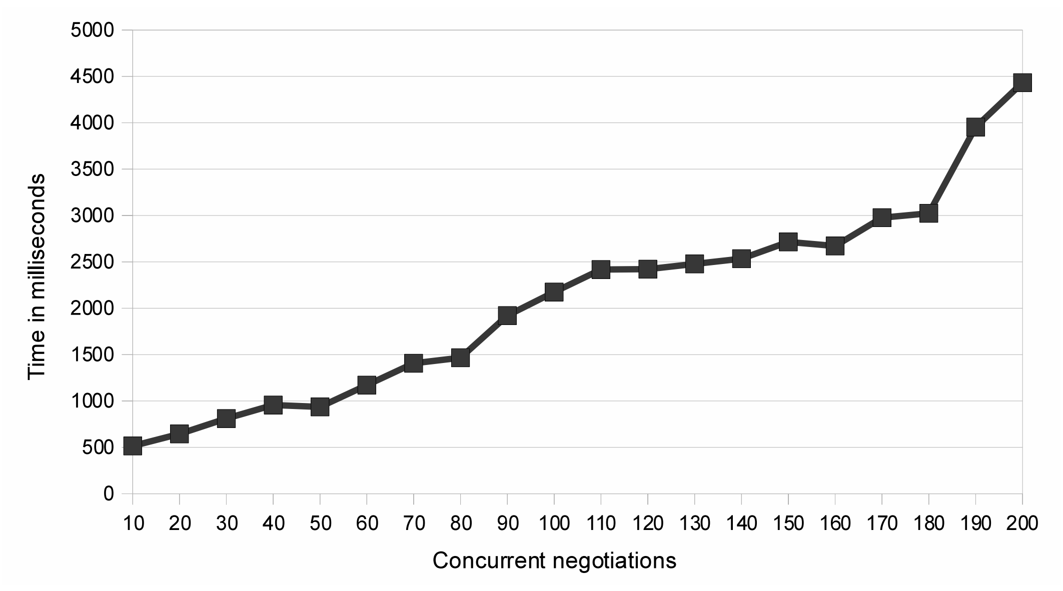


Figure 8.7: Scalability of the prototype with respect to the number of simultaneous trust negotiations

Chapter 9

Conclusions and future work

In this thesis we formalized an highly expressive resource negotiation language, able to support the specification of a large variety of conditions applying to single peers or groups of peers. Such language have been shown to be very flexible and to be easily adapted to different application scenarios.

We have also proposed a novel peer group joining protocol based on negotiations. Furthermore, we have presented an event-based resource availability checking protocol.

We implemented the resource negotiation language and the corresponding protocols in the JXTA P2P platform, thus providing a resource negotiation-based Membership Service. We have therefore extended the current monitoring mechanisms supported by JXTA with event handling capabilities, that enable the execution of specific actions when a specific event is detected.

In order to support the detection of such events we have introduced special peers, and proposed a simple approach to elect and manage them. Clearly, the existence of such peers in charge of controlling the validity of the strong formulae can raise some security problems. Peers may abuse of their power to validate formulae which are false, or not provide the functionalities they committed to. Such type of security threats can be controlled by means of techniques well known in the distributed system community and we are currently developing a fully distributed membership service taking advantage of technologies such as DHTs.

Further, we notice that the current approach still suffers of some potential problems that are typical of trust negotiations: malicious peers may purposely run negotiations to gather other peers' credentials, declaring to qualify for policies when they actually do not or share fake credentials/policies. These issues are partly addressed by the new stringent controls and verification protocols introduced with the Group Manager. Techniques such as those proposed in [50, 64] could be leveraged to further reduce the risks of misuse, and are object of our future work.

In this thesis we have also presented a multi-session dependable approach to trust negotiations. The proposed framework supports voluntary and unpredicted interruptions, enabling the negotiating parties to complete the negotiation despite temporary unavailability of resources. In designing the protocols, we have carefully considered all possible issues related to validity, temporary loss of data, and extended unavailability of one of the two negotiators.

To this extent, we further introduced protocols for mobile negotiations. Using the enhanced version of Trust-$\mathcal{X}$, a peer is able to suspend an ongoing negotiation and resume it with another (authenticated) peer. Negotiation portions and intermediate states can be safely and privately be passed among peers, to guarantee the stability needed to continue suspended negotiations.

The proposed procedures and protocols for suspension and recovery of multi-session, mobile trust negotiations do not impose a large overhead in terms of exchanged messages among peers and computations at the peers. The overall computational overhead is linear in the size of the exchanged policies and credentials, while the communication overhead, expressed as the number of exchanged messages, is (a fairly low) constant. Such linear dependency holds for messages' sizes as well. We remark that the most critical of such quantities is the number of exchanged messages because the network is often a bottleneck (especially in the case of mobile clients). Nevertheless, our solution requires a fixed number of messages, whose sizes linearly depends from the size of the negotiation tree built until the suspension.

The proposed framework is effective in managing complex, real-world negotiations involving long chains of mutual requests, possibly among more than two parties. We remark that the protocols presented in this work can be applied to any trust negotiation system that adopts a multi phase negotiation protocol. It is part of our future work to investigate how to migrate the membership service protocol any general trust negotiation infrastructure.

Using the techniques proposed in this thesis to the critical infrastructure environment shown us as the $\mathcal{X}$-RNL language here described satisfies the requirements presented in Chapter 6.2.1 for critical infrastructure information sharing systems. In fact, such negotiation language enables to express *exchange policies* and *constraints*, introducing formal mechanisms allowing to assign different level of trust to the actors of the sharing network, on the basis of the required information. Moreover it allows to model the concept of partial information sharing, i.e. the possibility to release only portions of a target information, on the basis of the identity and of the properties of the requesting actor.

By adopting such language it is possible, to create a distributed network of peers layered and clustered according to different kinds of characteristics. They can freely and securely exchange information about critical infrastructure preserving at the same time the strong constraints related to the confidentiality of such information. Figure 9.1 shows for example the case described in Chapter 6.2.1: the peers are organized into country clusters; they can negotiate and exchange directly some information with members of the same or different cluster, and indirectly by leaving the negotiation tasks to the government peers.

The present work leaves obviously opened some questions which are typical and not definitively solved in the information sharing world:

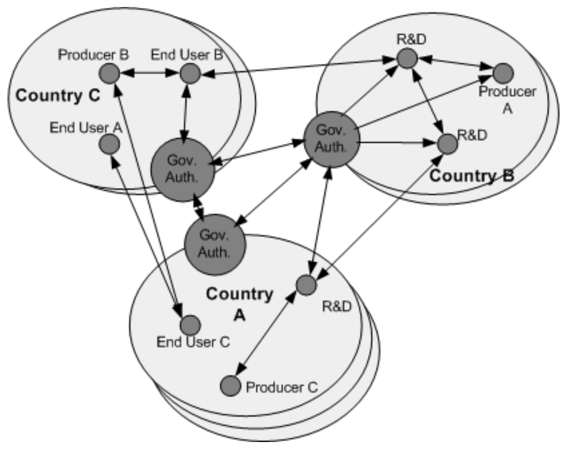


Figure 9.1: High level Information Exchange Network schema

(i) The owner of a released information loses any control over the released copy, i.e. she/he cannot force the actor which receives the information to apply the same distribution criteria on such information while requested to distribute it by a third actor.

(ii) The system is, at the moment, prone to data-mining analysis attacks. In fact by using such techniques, could be possible for a malicious actor, to use information obtained legally in order to guess information for which he has not any access right.

The first problem could be solved for example by implementing some reputation mechanism, while for the second, some *statistical disclosure control* and *privacy preserving data-mining* mechanisms can be identified and applied. We plan for the future, to address such issues. Moreover, we plan to implement a fully working peer-to-peer prototype of the presented architecture, in order to conduct an extensive on filed working test.

Moreover, we presented an approach for managing accesses in cognitive radio networks, when deployed in scenarios having conflicting requirements like a) security needs and b) high flexibility in managing dynamic reconfigurations. The proposed solution builds on the concept of trust negotiation, a well-known and accepted approach in the access control research area. We have defined a negotiation language for managing access control in a cognitive radio network and we applied it to a real-world critical scenario. Finally, we reported promising experimental results, showing the effectiveness of our approach.

With respect to our last application scenario, we have presented an extension of the framework introduced in [28], aimed at the integration of trust negotiations with an access control mechanism for resources in OSNs. This has been achieved by properly extending the language for expressing access control policies and integrating the relevant features of the Trust-$\mathcal{X}$ framework. Moreover, a feedback mechanism that takes into account the outcome of a trust negotiation between two nodes to dynamically set their trust level has been presented. Finally, several experiments have been carried out in order to show the feasibility of our approach. Further extensions we plan to work on are related to the automatic setting of the lifetime of dynamic relationships, the automatic identification of key resources and to the development of methods to compute the relevance of a resource.

Appendix A

Cryptographic protocols

In the following chapter we will summarize some well-known cryptographic protocols used all along the thesis.

A.1 Pedersen commitment protocol

A commitment protocol is a method that allows a party, referred to as the prover, to commit to a value to another party, referred to as the verifier, while keeping it hidden and preserving the prover's ability to reveal the committed value later. First introduced in [83], the Pedersen Commitment scheme is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. We describe how it works as follows.

Setup A trusted third party T chooses a finite cyclic group G of large prime order p so that the computational Diffie-Hellman problem is hard in G. In what follows we assume G is multiplicative. T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer α such that $h = g^\alpha$. Note that T may or may not know the number α. T publishes (G, p, g, h) as the system's parameters.

Commit The domain of committed values is the finite field F_p of p elements, which can be implemented as the set of integers $F_p = \{0, 1, \ldots, p?1\}$. For a party U to commit a value $m \in F_p$, U randomly chooses $r \in F_p$ and computes the commitment as

$$c = g^m h^r \in G \tag{A.1}$$

Open U shows the values m and r to open a commitment c. The verifier checks whether $c = g^m h^r$.

A.2 Shamir's Secret Sharing Scheme

Secret sharing refers to methods for distributing a secret amongst a group of participants, each of which is allocated a share of the secret. An example of secret sharing, which we adopt in Chapter 5 and Section B.12.1 the (k, n) threshold scheme by Shamir [95]. Such a scheme splits a secret S into n partial secrets so that k, with $k < n$, partial secrets are required to reconstruct S. The scheme works as follows.

- $(k$-1) random coefficients $\{a_1, \ldots, a_{k-1}\}$ are chosen.
- A polynomial $f(x) = a_o + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$, with $a_0 = S$, is generated.
- Based on $f(x)$, n shares are constructed. Each share is of the form $(i, f(i))$ where i is the input to the polynomial and $f(i)$ the output. Given any subset k of these pairs, the coefficients of the polynomial can be evaluated using interpolation. The secret, that is, $a_0 = S$ can thus be determined.

A.3 Damgård-Fujisaki commitment scheme and auxiliary protocols

Here we describe another set of commitment protocols we used in our approach: the protocols by Damgård and Fujisaki [35]. The basic steps of the protocols are the following:

Setup On input s (the security parameter) the outputs are:

a) an RSA modulus $n = pq$, such that $p = 2p' + 1$ and $q = 2q' + 1$, where p, p', q, q' are primes;

b) a random $h \in QR_n$, where QR_n denotes the set of quadratic residues modulo n;

c) a random $g \in \langle h \rangle$, where $\langle h \rangle$ denotes the group generated by h. Thus the public parameters generated by the setup phase are n, g, h.

Commit To commit an $m \in \mathbb{Z}$, the prover randomly chooses $r \in \mathbb{Z}$ and computes

$$\mathsf{commit}(m, r) = g^m h^r \mod n = c \tag{A.2}$$

Open To open a commitment c, the prover reveals m and r to the verifier which checks whether

$$c = g^m h^r \mod n \tag{A.3}$$

A.3.1 Proofs of knowledge

In many applications, such as in the trust negotiation protocol described in Chapter 5, revealing the committed value is not always required. Instead, it is sufficient that the prover proves to the verifier that it knows the committed value. Several such protocols, referred to as *proofs of knowledge* have been proposed. In our approach, we specifically need the following two proofs of knowledge:

1. *a committed value is equal to a given value*: given public parameters n, g, h of the Damgård-Fujisaki commitment scheme, a commitment c and an integer m', the prover proves the knowledge of m and r such that $c = g^m h^r$ and $m = m'$. Following the usual notation, we denote the protocol with:

$$PK\{(m,r) : c = g^m h^r \wedge m = m'\} \tag{A.4}$$

   We detail a well-known realization of such protocol as from [35]:

   i) The prover randomly chooses integers $y \in [0, \dots, TC2^s), t \in [0, \dots, 2^{B+2s})$, where T is an arbitrarily large integer constant, C is an integer superpolinomially greater than s but smaller than the order of $\langle h \rangle$ and B is an integer is an integer polynomially greater than s. The prover sends $d = g^y h^t$ and y to the verifier.
   ii) The verifier chooses a random integer $e \in [0, \dots, C)$ and sends it to the prover.
   iii) The prover sends to the verifier $v = t + er$. The verifier computes $u = y + em'$ checks whether $g^u h^v = dc^e$

2. *a committed value lies in a given interval*: given public parameters n, g, h of the Damgård-Fujisaki commitment scheme, a commitment c and integers m' and m'', the prover proves the knowledge of m and r such that $c = g^m h^r$ and $m' \leq m \leq m''$. Following the usual notation, we denote the protocol with:

$$PK\{(m,r) : c = g^m h^r \wedge m' \leq m \leq m''\} \tag{A.5}$$

   A well-known example of such a protocol is found in [22].

A.4 Fiat-Shamir heuristics

In a non-interactive version of the above proofs of knowledge, the prover, instead of receiving a random challenge from the verifier (step ii) in the above protocol, generates by itself such random value by taking the output of a cryptographic hash function H [43]. The non-interactive version of the above protocol is:

i) The prover randomly chooses integers $y \in [0, \dots, TC2^k)$, $s \in [0, \dots, 2^{B+2k})$, where T is an arbitrarily large integer constant, C is an integer superpolinomially greater than k but smaller than the order of $\langle h \rangle$ and B is an integer is an integer polynomially greater than k. The prover sends $d = g^y h^s$, $H(d)$ (where H is a cryptographic hash function), $v = s + H(d)r$ and y to the verifier.

iii) The verifier computes $u = y + H(d)m'$ checks whether $g^u h^v = dc^e$.

If we assume the (widely accepted) random oracle hypothesis, the modified protocol is a zero-knowledge proof of knowledge, that is, the prover proves to the verified the knowledge of the secret without leaking any information to the verifier [22].

We adopt the commitment scheme by Damgård-Fujisaki because it fits trust negotiation requirements. First, it imposes fewer constraints on the input than other schemes: it accepts as valid input an arbitrary integer (and not only a group member, as in the case of the Pedersen commitment scheme [83]); second it has efficient knowledge proofs that a committed value is equal to a given value and that a committed value lies in a given interval. This last property is used in our approach to test whether a committed value is less than a given value.

Appendix B

Implementation

Aim of this appendix chapter is to provide an overview of the architectural design of the Trust-$\mathcal{X}$ framework described in the previous Chapters.

B.1 Prototype evolution

The original Trust-$\mathcal{X}$ prototype has been implemented by the University of Milano. It as been initially presented in [13]. During the recent years it has been extended in order to support trust negotiation recovery [105], it has been converted into a web-service[101] and eventually extended to support resource negotiations between peers' groups [104]. In order to support such features and to ease the integration of future extensions, the code of Trust-$\mathcal{X}$ prototype has been thoroughly re-factorized and in large parts completely rewritten. This work documents the current architectural layout of the Trust-$\mathcal{X}$ prototype, as resulted from the refactoring and rewriting steps.

B.2 Architecture Overview

The Trust-$\mathcal{X}$ prototype has been implemented using Java 1.6 [77]. It supports the trust negotiation language presented in [105] and the following evolutions described in the previous chapters and in [102, 106, 104, 25, 24, 8].

The source code has been organized in several packages according to the functionalities provided. Such packages are the following:

- `communicationLayer`, it provides a defined set of methods used by the upper layers to communicate with the other peers;
- `database`, it provides a standard interface for the database used in the different installations;
- `parser`, it provides a parser for the XML documents exchanged during the trust negotiation;

- `trustx`, it provides the classes required for the execution of a Trust-$\mathcal{X}$ negotiation. It also contains the packages of the different phases and the packages for some other features:
  - `gui`, it provides a simple graphical user interface to configure and execute a *client* peer of a Trust-$\mathcal{X}$ trust negotiation;
  - `introductoryPhase`, it provides the interface and the required classes to execute the introductory phase;
  - `policyEvaluation`, it contains interfaces and classes required to perform the policy evaluation phase;
  - `credentialExchange`, it provides the features required by the credential exchange phase;
  - `treemanager`, it contains the classes for the management of the negotiation tree;
  - `groupmanager`, it contains the classes required to collaboratively verify the availability of the peers of the group and of the resources they stated to provided to the group.
  - `commitmend`, such package contains the interfaces and some implementations of the commitments required by the long running trust negotiations.

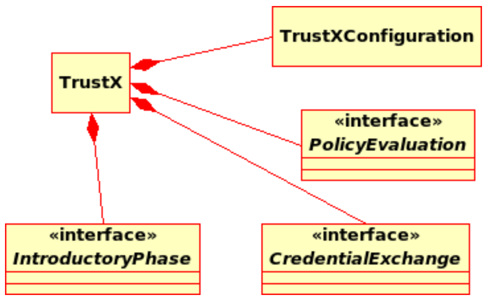


Figure B.1: The high level class diagram of the Trust-$\mathcal{X}$ framework

The `parser` package contains the classes required for parsing, validating and creating the documents used in the trust negotiation and is therefore. It validates both $\mathcal{X}$-TNL and $\mathcal{X}$-RNL languages by means of a set of XML schemas.

To provide an easier way to create a trust negotiation, we implemented a Graphical User Interface (GUI) which provides a simple interface for the configuration and the the execution of a *client* peer for the Trust-$\mathcal{X}$. Figure B.2, B.3 and B.4 show respectively, the main window, how to configure the database connection and the communication layer which will be used.

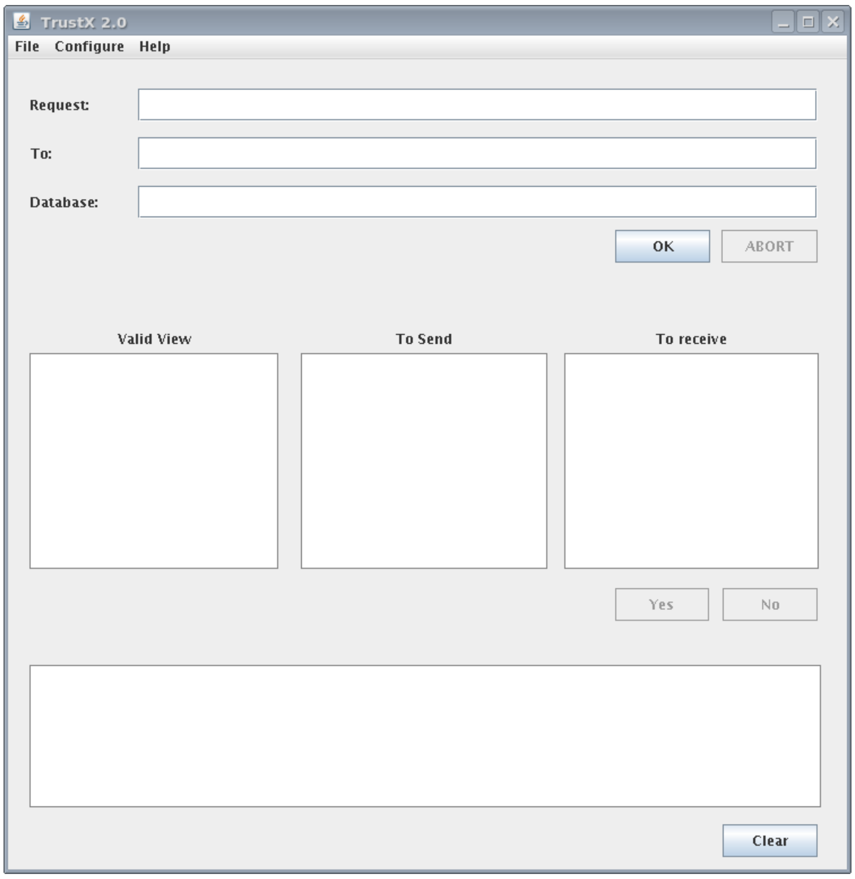


Figure B.2: The main window

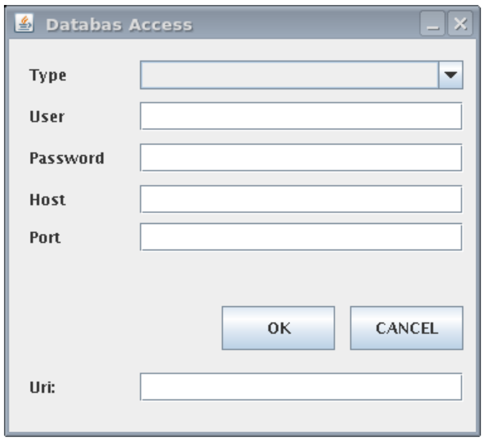


Figure B.3: The dialog to configure the connection with the database

B.2.1 Example of execution

An example on how to execute the Trust-$\mathcal{X}$ framework is provided by two targets within the ant config file (`build.xml`), contained in the source distribution. Such targets are *testServer* and *testClient*.

The results of the execution of such targets is show in Figure B.5.

B.3 Communication Layer

The communication layer is contained in the `communicationLayer` package. Such package contains the interface `CommunicationLayer` which every real communication layer has to provide.

Such interface is handled by the `CommunicationLayerManager`, which is in charge for returning the actual implementation of the configuration obtained as parameter of the static method `getCommunicationLayer()`.

The interface `CommunicationLayer` provides a core set of methods, necessary to allow the interactions of the parties involved in the trust negotiation. Such interface, shown in Figure B.6, provides the following methods:

So far, there are three classes implementing the `CommunicationLayer` interface:

1. `CommunicationLayerOverJXTA`, a communication channel exploiting the JXTA network,

2. `CommunicationLayerOverIPv4`, a communication channel which uses TCP/IP socket, and

3. `CommunicationLayerOverIPv4SSL`, a communication layer which uses a socket connection protected with the SSL protocol.

B.4 Database

Besides the communication layer, an abstract class regarding the database used by parties has been written. This has been achieved by means of the interface `Database`. Such interface provides the methods required to retrieve the credentials and the policies used in a Trust-$\mathcal{X}$ negotiation.

In order to support a new DBMS, it is required to extend the `Configuration` class to handle the connection URI required by the DBMS.

By means of the method `getDatabaseInstance()`, the class `Configuration` will create an object of a class, which depends on the DBMS, implementing the `Database` interface.

The `Database` interface, shown in Figure B.8, provides the method required to search and retrieve the resources present in the database and to retrieve the associated policies.

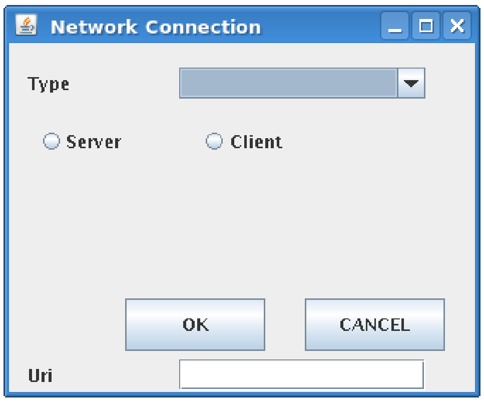
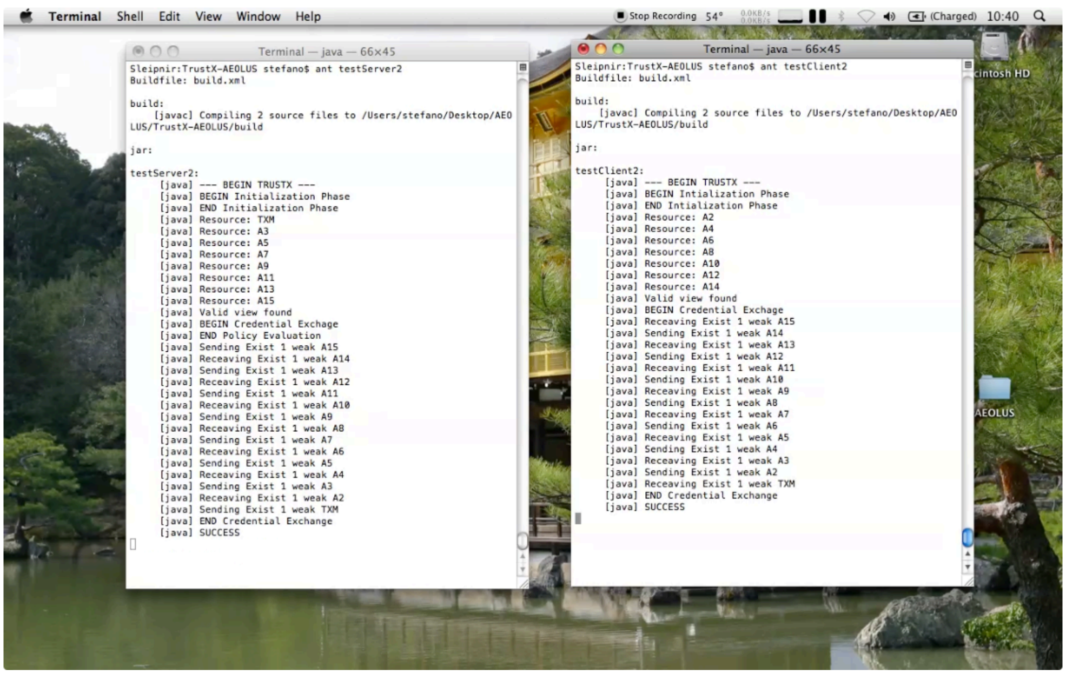


Figure B.4: The dialog to configure the communication layer


Terminal Shell Edit View Window Help
Stop Recording 54° (Charged) 10:40
Terminal — java — 66×45
Macintosh HD
AEOLUS

```
Sleipnir:TrustX-AEOLUS stefano$ ant testServer2
Buildfile: build.xml

build:
    [javac] Compiling 2 source files to /Users/stefano/Desktop/AEO
LUS/TrustX-AEOLUS/build

jar:

testServer2:
     [java] --- BEGIN TRUSTX ---
     [java] BEGIN Initialization Phase
     [java] END Initialization Phase
     [java] Resource: TXM
     [java] Resource: A3
     [java] Resource: A5
     [java] Resource: A7
     [java] Resource: A9
     [java] Resource: A11
     [java] Resource: A13
     [java] Resource: A15
     [java] Valid view found
     [java] BEGIN Credential Exchage
     [java] END Policy Evaluation
     [java] Sending Exist 1 weak A15
     [java] Receaving Exist 1 weak A14
     [java] Sending Exist 1 weak A13
     [java] Receaving Exist 1 weak A12
     [java] Sending Exist 1 weak A11
     [java] Receaving Exist 1 weak A10
     [java] Sending Exist 1 weak A9
     [java] Receaving Exist 1 weak A8
     [java] Sending Exist 1 weak A7
     [java] Receaving Exist 1 weak A6
     [java] Sending Exist 1 weak A5
     [java] Receaving Exist 1 weak A4
     [java] Sending Exist 1 weak A3
     [java] Receaving Exist 1 weak A2
     [java] Sending Exist 1 weak TXM
     [java] END Credential Exchange
     [java] SUCCESS
```

```
Sleipnir:TrustX-AEOLUS stefano$ ant testClient2
Buildfile: build.xml

build:
    [javac] Compiling 2 source files to /Users/stefano/Desktop/AEO
LUS/TrustX-AEOLUS/build

jar:

testClient2:
     [java] --- BEGIN TRUSTX ---
     [java] BEGIN Intialization Phase
     [java] END Intialization Phase
     [java] Resource: A2
     [java] Resource: A4
     [java] Resource: A6
     [java] Resource: A8
     [java] Resource: A10
     [java] Resource: A12
     [java] Resource: A14
     [java] Valid view found
     [java] BEGIN Credential Exchage
     [java] Receaving Exist 1 weak A15
     [java] Sending Exist 1 weak A14
     [java] Receaving Exist 1 weak A13
     [java] Sending Exist 1 weak A12
     [java] Receaving Exist 1 weak A11
     [java] Sending Exist 1 weak A10
     [java] Receaving Exist 1 weak A9
     [java] Sending Exist 1 weak A8
     [java] Receaving Exist 1 weak A7
     [java] Sending Exist 1 weak A6
     [java] Receaving Exist 1 weak A5
     [java] Sending Exist 1 weak A4
     [java] Receaving Exist 1 weak A3
     [java] Sending Exist 1 weak A2
     [java] Receaving Exist 1 weak TXM
     [java] END Credential Exchange
     [java] SUCCESS
```

Figure B.5: Example of execution of the Trust-$\mathcal{X}$ framework

```
public interface CommunicationLayer {
  public Configuration getConfiguration();
  public void open() throws UnknownHostException,
                     IOException;
  public void close() throws IOException;
  public Message readMessage() throws IOException;
  public void writeMessage(Message msg)
                throws IOException;

}
```

Figure B.6: The interface `CommunicationLayer`

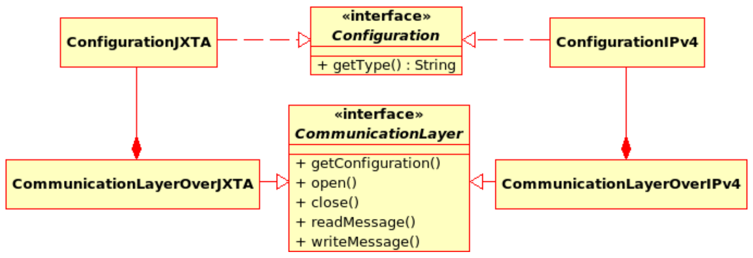


Figure B.7: The class diagram of the package `communicationLayer`

```
public interface Database{
    public void setConfiguration(Configuration conf)
           throws InvalidConfigurationException;
    public Configuration getConfiguration();
    public void connect() throws NullPointerException,
                   IOException;
    public int searchResource(String str)
           throws ResourceNotFoundException,
                   IOException, CertificateException;
    public OurDocument getResource(int cid)
            throws ResourceNotFoundException,
                   IOException, CertificateException;
    public OurDocument searchAndGetResource(String str)
            throws ResourceNotFoundException,
                    IOException, CertificateException;
    public OurDocument readPolicy(int cid)
            throws IOException, CertificateException;
    public OurDocument searchAndReadPolicy(String resource)
            throws ResourceNotFoundException,
                    IOException, CertificateException;
}
```

Figure B.8: The interface `Database`

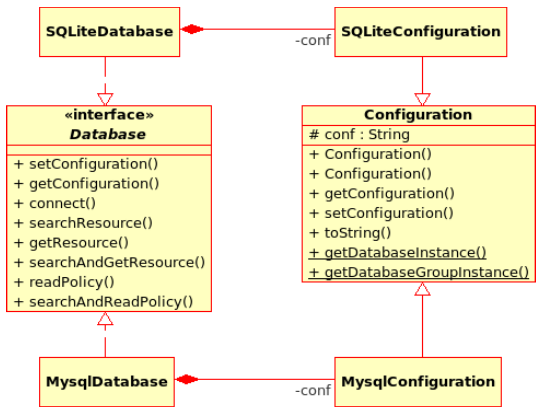


Figure B.9: The class diagram for the `database` package

If the negotiation is performed not by two peers but by a peer and a group, the Group Manager, described in Section B.10.1, needs to access a modified version of the `Database` interface. Such interface is called `DatabaseGroup`, shown in Figure B.10, and is obtained by calling the method `getDatabaseGroupInstance()` of the `Configuration` class. The methods provided by the `DatabaseGroup` interface in addiction to the ones provide by the `Database` interface have been introduced to perform searches of resources and policies among the ones belonging to the group.

The supported DBMS supported up to now are the following:

- Oracle[79],
- MySQL[78],
- PostgreSQL[84] and
- SQLite[99].

B.5 Introductory Phase

The introductory phase is the first phase of the trust negotiation provided by Trust-$\mathcal{X}$. Such phase is in charge of performing the tasks required to synchronize the peers involved in the negotiation and, according to the implementation used in the current deployment, perform optional tasks such as find an agreement upon the resource to be negotiated, recover a previously suspended (both voluntarily and accidentally) negotiation and so on. The interface `IntroductoryPhase` provides a common set of methods to be used by the peer involved in the negotiation but each class implementing the `IntroductoryPhase` interface may provide more advanced methods.

The classes implementing such interface are the following:

```
public interface DatabaseGroup {
    public void setConfiguration(Configuration conf)
                    throws InvalidConfigurationException;
    public Configuration getConfiguration();
    public void connect() throws InvalidConfigurationException,
                        IOException;
    public OurDocument searchAndReadPolicy(String resource)
                    throws ResourceNotFoundException,
                        IOException,
                        CertificateException;
    public OurDocument readPolicy(int cid)
                    throws IOException,
                           CertificateException;
    public OurDocument searchAndReadGroupPolicy(String resource,
           int quantification)
                    throws ResourceNotFoundException,
                           IOException,
                           CertificateException;
    public int addPeer(String uri) throws IOException;
    public int[] searchResources(String str,int quantified)
                    throws ResourceNotFoundException,
                           IOException, CertificateException;
    public int searchResource(String str) throws
                           ResourceNotFoundException,
                           IOException, CertificateException;
    public OurDocument getResource(int res)
                   throws CertificateException, IOException;
    public int getOwnerResource(int res)
                   throws IOException, CertificateException;
    public int addResource(String uri, OurDocument res,
           OurDocument policy, boolean necessary) throws IOException;
    public void addStrongRequest(int peerId, int resId)
                   throws IOException;
    public void addStrongRequest(int peerId, int resId, int time)
                   throws IOException;
    public boolean testStrongRequestValidity(int peerId, int resId)
                throws IOException;
    public DatabasePostconditions removePeer(String uri)
                   throws IOException;
    public DatabasePostconditions removeResource(
           String uri, String resource) throws IOException;
    public int getPeersNumber() throws IOException;
}
```

Figure B.10: The interface `DatabaseGroup`

- `DummyIntroductoryPhase`, a simple implementation created for debug purposes.
- `RebateIntroductoryPhase`, such class implements the rebate formulae during the introductory phase, as described in [24, 25] and in Section 6.1.
- `RecoveryIntroductoryPhase`, this class implements the resume / recovery phases described in [106] and in Chapter 5. It will be better described in Section B.12.
- `StandardIntroductoryPhase`, this class implements all the above features. This implementation is the one which should be used in a real deployment of the Trust-$\mathcal{X}$ framework.

B.6 Policy Evaluation Phase

The policy evaluation phase occurs when each peer involved in the negotiation sends disclosure policies to the other peer upon the request of a resource. It terminates when the peers find a deliv subtree within the current negotiation tree (see Section B.8 and [13]).

The methods required by such phase are provided by means of the interface `PolicyEvaluation`, as show in Figure B.11.

```
public interface PolicyEvaluation {
    public void evaluateTree(Tree tree) throws IOException;
    public boolean searchAnotherValidView() throws IOException;
    public List<Tree> getLastValidViews();
    public boolean proposeValidViewC(boolean b) throws IOException;
    public boolean proposeValidViewS(boolean b) throws IOException;
}
```

Figure B.11: The interface `PolicyEvaluation`

A negotiation can be performed between two single peers and between a peer and a group of peers. This fact required the creation of two distinct classes implementing the interface `PolicyEvaluation`.

A trust negotiation is normally a symmetric process, which means that both involved parties perform the same operations in order to execute the trust negotiation. The introduction of the possibility to perform a negotiation between a peer and a group of peers removes this symmetry. Therefore, the method required for a P2P negotiation has been implemented in the class `PolicyEvaluationImpl`. On the other side, for the SP2G negotiation three classes have been implemented. the first is the class `PolicyEvaluationGroup`, an abstract class providing the methods used by both the Group Manager and the extern peer. Both the class executed by the GroupManager, called `PolicyEvaluationGroupManager`, and the class for the entering peer, called `PolicyEvaluationGroupPeer`, extend the abstract class `PolicyEvaluationGroup`.

B.7 Credential Exchange Phase

In the credential exchange phase the peers involved in the trust negotiation actually exchange the credentials and resources named in the valid view (see Section B.9).

The method required for the execution of the credential exchange phase is provided by the interface `CredentialExchange`.

```
public interface CredentialExchange {
    public boolean exchange(Tree validView)
            throws IOException;
}
```

Figure B.12: The interface `CredentialExchange`

Similarly to the policy evaluation phase, the operations really performed by a credential exchange phase lightly differ with respect of the type of negotiation performed. Again, it has been implemented a class for the negotiation P2P, called `CredentialExchangeImpl`, and three classes for the negotiation SP2G.

The classes for the negotiation SP2G follow the same structure presented for the policy evaluation phase. The abstract class `CredentialExchangeGroup` implements the common methods and two classes extends such class in order to perform the operation specifically required by each parties.
Therefore, the `trustx.credentialExchange` package contains two classes:

- `CredentialExchangeGroupManager` and
- `CredentialExchangeGroupPeer`.

B.8 Negotiation Tree

The classes and interfaces contained in the `trustx.tree` package are the core of the trust negotiation performed by the Trust-$\mathcal{X}$ framework. They implement the negotiation tree, which is the data structure containing credentials and (representation of) resources, and it is dynamically built by peers during the exchange of their disclosure policies.

A Trust-$\mathcal{X}$ negotiation tree is implemented by the class `Tree`, which extends the class `TreeManager`. The class `TreeManager` defines the interface used by the classes implementing the different negotiation phases and provides a set of methods common to each implementation of the negotiation tree. The class `Tree` provides an efficient implementation of the data structures required by the negotiation tree.

The negotiation tree evolves according to the policies exchanged by the parties. The terms which compose in the policies are represented within the negotiation tree by the class `Node`. Such class models the characteristic of a possible term as described in [42, 106, 104]. In case of extensions at the policy language, unless the modifications involve the structure of the policy, the class `Node` is the one which must be modified in order to implement the required features.

Moreover, the class `TreeManager` provides the methods required to save the negotiation tree to a stable storage and to later load the same negotiation tree from the stable storage.

B.9 Valid View

A valid view, as described in [13] and in Chapter 3, is a subtree of the negotiation tree in which every nodes are marked as *deliverable*[1]. The class `TreeManager` is able to identify if within the negotiation tree there is at least a subtree with such property. With the method `getValidView()` a user is able to obtain all the subtrees of the negotiation tree which are definable as valid view. With such list, a user may choose one of the available valid view and, instantiating an object of the class `ValidViewManager` can operate on the subtree as it were a *trust sequence*[2].

B.10 Groups

As anticipated, the Trust-$\mathcal{X}$ framework has been extended in order to be able to handle trust negotiation between a peer and a group of peers (see Chapter 4). To obtain such type of negotiation and the advanced features introduced in the language $\mathcal{X}$-RNL, such as weak and strong formulae and quantifiers, some way to handle the state of the group are needed. The package `trustx.groupmanager` provides the interfaces and the classes required to handle the state of the group and the consistency of the strong formulae.

B.10.1 Group Manager

The manager of the group is the peer that created the group. The class `GroupManager` provides the methods required by the group manager in order to have an updated view of the group and to handle the consistency of the formulae marked as strong. We recall that a strong formula is required to be valid along the whole time period a peer is member of a group. The object of the class `GroupManager` must be instantiated by the group manager when it creates the group protected with the Trust-$\mathcal{X}$ framework and must be kept for the whole duration of the existence of the group.

[1]We recall that a node is marked as deliverable if its disclosure policy is satisfied, i.e. if and only if it has no disclosure policy associated or there exist a set of terms in its disclosure policy marked as deliverable such that the whole disclosure policy holds, see Chapter 3.

[2]A trust sequence is the serialization of the nodes contained in a valid view, refer to Chapter 3.6.3

B.10.2 Group Agent

The group agent is a thread that will remain in each peer of the group to communicate with the group manager in order to collaboratively determinate the state of the group. More precisely, at the end of the trust negotiation required in order to join the group, if the negotiation ends successfully, an object of the class `ConnectionToGroup` will be instantiated. Through the method provided by such class the agent will be able to communicate with the group manger in order to send it the modification of the state of the group the peer detects. This include even the modification caused by the peer itself such as when a peer revoke the availability of a resource provided or when it decides to leave the group.

The class `ConnectionToGroup` provides another important feature. Such feature is the *hearthbeat*. The heartbeat is a ping sent by the group manager to each peer within the group. This ping is required in order to verify the liveness of each peer because each peer may crash or became unavailable. If such unavailability is protracted for more then an user define period, the group manager will mark the peer as definitively unavailable and will operate to remove the peer and its resources from the group pool.

B.11 TrustXMembershipService

To experiment the proposed solution, the JXTA framework has been extended by means of the creation of a new MembershipService which exploits the Trust-$\mathcal{X}$ framework.

To perform the two side authentication required by Trust-$\mathcal{X}$, we also created an authentication service, implemented by means of the class `TrustXAuthService`. Such authentication service provides the server side of the Trust-$\mathcal{X}$ negotiation and is therefore associated with the group it is mandated to protect. To achieve that, such class implements the interface `Service`, which is required to create core JXTA services, and the interface `Runnable` in order to be able to execute the authentication service as a separate thread. Moreover, it provides the services required by the GroupManager described in Section B.10.1.

```
public interface MembershipService {
        public Authenticator  apply(AuthenticationCredential ac) throws
                PeerGroupExeception, ProtocolNotSupportedException;
        public Credential join(Authenticator auth) throws
                PeerGroupExeception;
        public resign() throws PeerGroupExeception;
}
```

Figure B.13: The interface `MembershipService`

The class `TrustXMembershipService` implements the interface `MembershipService`. Such interface defines the methods required by a MembershipService compliant to the JXTA architecture. Therefore, the implementing class provides the `apply` method. Such method, upon the examination that the `AuthenticationCredential` received as parameter requires the execution of a join process which exploits Trust-$\mathcal{X}$, returns an object implementing the interface `Authenticator`. In the specific case, the object is an instance of the class `TrustXAuthenticator`, described in what follows.

Another method required by the interface `MembershipService` is the method `join`, which if the joining process ends successfully, returns an object of the class `TrustXCredential`, which contains the GroupAgent described in Section B.10.2. Finally, the method `resign` is used to leave the group.

The class `TrustXAuthenticator` implements the interface `Auhenticator`. An instance of such class is obtained by the invocation of the method `apply` of the class `TrustXMembershipService`. The `TrustXAuthenticator` is the class which implements the client side of a Trust-$\mathcal{X}$ negotiation. In order to perform such task, upon its creation, the instance must be instructed with the correct information such as the location of the database and the communication layer to use for the negotiation. Moreover, it implements all the methods required by the implemented interface `Authenticator`, which follow the behavior defined by the JXTA architecture.

```
public class TrustXAuthenticator {
        public void setTrustXDatabase(String dbUrl) throws
                NullPointerException, InvalidConfigurationException;
        public void useJXTAConnection() throws PeerGroupException;
        public void useIPV4Connection(String conn);
        ...
}
```

Figure B.14: The methods of the class `TrustXAuthenticator`

All the classes described above can be found in the package
`net.jxta.impl.membership.trustx`

B.11.1 Integration in the JXTAShell

The MembershipService described in the previous part of the current section, has been made available to the user by means of commands of the JXTA shell. The following commands have been introduced:

- `trustx.newpgrp`, such command operates similarly to the original command `newprgp`. It creates a new peergroup using the TrustXMembershipService described above as MembershipService.
- `trustx.join`, such command let a peer join a peergroup created with the `trustx.newpgrp` command.
- `trustx.authsvc`, this command start and stop the TrustXAuthService for the current peergroup.

- trustx.login, which execute the authentication of the current peer in the joined peergroup. It has as side effect the execution of the GroupAgent within the current peer.

B.12 Multisessions

The features described in Chapter 5 have also been implemented in the Trust-$\mathcal{X}$ prototype.

As introduced in Section B.5, a specific implementation of the IntroductoryPhase interface has been developed to provide the prototype the ability to identify a suspended negotiation which requires to be recovered. Moreover, such implementation of the IntroductoryPhase interface, called RecoveryIntroductoryPhase, allows the negotiating parties to perform the steps of the resume phase for both client and server.

Beside the introductory phase, the classes PolicyEvaluationPhaseImpl and CredentialExchangePhaseImpl have been extended in two new classes:

- PolicyEvaluationWithMultisession and
- CredentialExchangeWithMultisession

Such classes provide support to the suspension protocols described in Chapter 5.3.

B.12.1 Secret Sharing

Both the class PolicyEvaluationPhaseWithMultisession and the class CredentialExchangeWithMultisession take advantage of the content of the package trustx.splitting and in particular of the interface Splitter which provides the method required to create and reconstruct the serialization of the state of the negotiation.

```
public interface Splitter {
        public SplitParameters generateParameters(long k, long n);
        public List<Share> split(Tree t, SplitParameters parameters);
        public Tree merge(List<Share> shares)
                throws TooFewSharesException;
}
```

Figure B.15: The interface Splitter

As a proof of concept a class implementing such interface is provided. Namely, the class ShamirSplitter provides the SSS (Shamir Secret Sharing) protocol, described in Appendix A.2.

B.12.2 Commitment schemas

The `RecoveryIntroductoryPhase` takes advantage of the content of the package `trustx.commitment` to restore the state of the negotiation. The classes contained in such package are especially when the multi-session protocol is configured to save negotiation snapshot single-side.

The package provides the `Commitment` interface, shown in Figure B.16, in which are defined two methods to both commit and decommit the negotiation tree.

```
public interface Commitment {
        public Tree commit(Tree t);
        public Tree decommit(Tree commitedTree);
}
```

Figure B.16: The interface `Commitment`

Again as a proof of concept, the package `trustx.introductory.commitment` provides two implementations of the `Commitment` interface. The first implementing class is called`PedersenCommitment` and it provides the commitment schema defined in [83]. The second class, called `CommitmentWithEllipticCurves`, takes advantages of the jPBC library [36] to implement the commitment scheme defined in [46].

Bibliography

[1] *Proceedings of the Network and Distributed System Security Symposium, NDSS 2001, San Diego, California, USA* (2001), The Internet Society.

[2] ADAMS, C., AND LLOYD, S. *Understanding PKI: Concepts, Standards, and Deployment Considerations*, 2nd ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[3] AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[4] AKYILDIZ, I. F., LEE, W.-Y., VURAN, M. C., AND MOHANTY, S. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *COMPUTER NETWORKS JOURNAL (ELSEVIER 50* (2006), 2127–2159.

[5] ALBERTS, C., AND DOROFEE, A. *Managing Information Security Risks: The OCTAVE (SM) Approach*. Addison-Wesley Professional, July 2002.

[6] ASHPOLE, B., EHRIG, M., EUZENAT, J., AND STUCKENSCHMIDT, H., Eds. *Integrating Ontologies '05, Proceedings of the K-CAP 2005 Workshop on Integrating Ontologies, Banff, Canada, October 2, 2005* (2005), vol. 156 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[7] BACKES, M., CAMENISCH, J., AND SOMMER, D. Anonymous yet accountable access control. In *WPES* (2005), V. Atluri, S. D. C. di Vimercati, and R. Dingledine, Eds., ACM, pp. 40–46.

[8] BALDINI, G., BRAGHIN, S., NAI FOVINO, I., AND TROMBETTA, A. Adaptive and distributed access control in cognitive radio networks. In *AINA* (2010), IEEE Computer Society, pp. 988–995.

[9] BARLOW, T., HESS, A., AND SEAMONS, K. E. Trust negotiation in electronic markets. In *Proceedings of the Eighth Research Symposium on Emerging Electronic Markets (RSEEM 01)* (September 2001).

[10] BECKER, M. Y., FOURNET, C., AND GORDON, A. D. Design and semantics of a decentralized authorization language. In *CSF* (2007), IEEE Computer Society, pp. 3–15.

[11] BERNTHAL, B., AND JESUALE, N. Smart radios and collaborative public safety communications. In *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks* (October 2008).

[12] BERTINO, E., FERRARI, E., AND SQUICCIARINI, A. C. Privacy-Preserving Trust Negotiation. *Proceedings of 4th Privacy Enhancing Technologies Workshop, Toronto, CA* (May 2004).

[13] BERTINO, E., FERRARI, E., AND SQUICCIARINI, A. C. Trust-$\mathcal{X}$: A Peer-to-Peer Framework for Trust Establishment. *IEEE Trans. Knowl. Data Eng. 16*, 7 (2004), 827–842.

[14] BERTINO, E., AND TAKAHASHI, K., Eds. *Proceedings of the 4th Workshop on Digital Identity Management, Alexandria, VA, USA, October 31, 2008* (2008), ACM.

[15] BLAZE, M., FEIGENBAUM, J., AND KEROMYTIS, A. D. The role of trust management in distributed systems security. In *Secure Internet Programming* (1999), J. Vitek and C. D. Jensen, Eds., vol. 1603 of *Lecture Notes in Computer Science*, Springer, pp. 185–210.

[16] BLAZE, M., FEIGENBAUM, J., AND LACY, J. Decentralized trust management. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 1996), IEEE Computer Society, p. 164.

[17] BLAZE, M., IOANNIDIS, J., AND KEROMYTIS, A. D. Trust management for ipsec. In *NDSS* [1].

[18] BOHRING, H., AND AUER, S. Mapping xml to owl ontologies. In *Leipziger Informatik-Tage* (2005), K. P. Jantke, K.-P. Fähnrich, and W. S. Wittig, Eds., vol. 72 of *LNI*, GI, pp. 147–156.

[19] BONATTI, P. A., COI, J. L. D., OLMEDILLA, D., AND SAURO, L. Policy-driven negotiations and explanations: Exploiting logic-programming for trust management, privacy & security. In *ICLP* (2008), M. G. de la Banda and E. Pontelli, Eds., vol. 5366 of *Lecture Notes in Computer Science*, Springer, pp. 779–784.

[20] BONATTI, P. A., COI, J. L. D., OLMEDILLA, D., AND SAURO, L. Rule-based policy representations and reasoning. In *REWERSE*, F. Bry and J. Maluszynski, Eds., vol. 5500 of *Lecture Notes in Computer Science*. Springer, 2009, pp. 201–232.

[21] BONATTI, P. A., COI, J. L. D., OLMEDILLA, D., AND SAURO, L. A rule-based trust negotiation system. *IEEE Trans. Knowl. Data Eng. 22*, 11 (2010), 1507–1520.

[22] BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *EUROCRYPT* (2000), pp. 431–444.

[23] BOWERS, K. D., BAUER, L., GARG, D., PFENNING, F., AND REITER, M. K. Consumable credentials in linear-logic-based access-control systems. In *NDSS* (2007), The Internet Society.

[24] BRAGHIN, S., NAI FOVINO, I., AND TROMBETTA, A. Advanced trust negotiation in critical infrastructures. In *International Conference on Infrastructure Systems* (November 2008).

[25] BRAGHIN, S., NAI FOVINO, I., AND TROMBETTA, A. Advanced trust negotiations in critical infrastructures. *International Journal of Critical Infrastructures 6*, 3 (2010), 225–245.

[26] BRANDS, S. A. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[27] CAMENISCH, J., AND HERREWEGHEN, E. V. Design and implementation of the *demix* anonymous credential system. In *ACM Conference on Computer and Communications Security* (2002), V. Atluri, Ed., ACM, pp. 21–30.

[28] CARMINATI, B., FERRARI, E., AND PEREGO, A. Enforcing access control in web-based social networks. *ACM Trans. Inf. Syst. Secur. 13*, 1 (2009).

[29] CERI, S., GOTTLOB, G., AND TANCA, L. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng. 1*, 1 (1989), 146–166.

[30] CHAUM, D. Showing credentials without identification transfeering signatures between unconditionally unlinkable pseudonyms. In *AUSCRYPT* (1990), J. Seberry and J. Pieprzyk, Eds., vol. 453 of *Lecture Notes in Computer Science*, Springer, pp. 246–264.

[31] CHEN, R., AND YEAGER, B. Poblano - a distributed trust model for peer-to-peer networks. Available on-line at: `http://www.JXTA.org/docs/trust.pdf`.

[32] COI, J. L. D., OLMEDILLA, D., BONATTI, P. A., AND SAURO, L. Protune: A framework for semantic web policies. In *International Semantic Web Conference (Posters & Demos)* (2008), C. Bizer and A. Joshi, Eds., vol. 401 of *CEUR Workshop Proceedings*, CEUR-WS.org.

[33] COULOURIS, G. F., DILLIMORE, J., AND KINDBERG, T. *Distributed Systems: Concepts and Design*, fourth ed. Addison-Wesley, 2005.

[34] COX, L. P., AND NOBLE, B. D. Samsara: honor among thieves in peer-to-peer storage. In *SOSP* (2003), M. L. Scott and L. L. Peterson, Eds., ACM, pp. 120–132.

[35] DAMGÅRD, I., AND FUJISAKI, E. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT* (2002), Y. Zheng, Ed., vol. 2501 of *Lecture Notes in Computer Science*, Springer, pp. 125–142.

[36] DE CARO, A. jPBC: The Java Pairing Based Cryptography Library. Available on-line at: `http://gas.dia.unisa.it/projects/jpbc/`.

[37] DEN BRABER, F., DIMITRAKOS, T., GRAN, B. A., LUND, M., STØLEN, K., AND AAGEDAL, J. *The CORAS methodology: model-based risk assessment using UML and UP*. IGI Publishing, Hershey, PA, USA, 2003, pp. 332–357.

[38] DENKER, G., ELENIUS, D., SENANAYAKE, R., STEHR, M., AND WILKINS, D. A policy engine for spectrum sharing. In *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (April 2007), pp. 55 – 65.

[39] DENKER, G., ELENIUS, D., SENANAYAKE, R., STEHR, M.-O., AND WILKINS, D. A Policy Engine For Spectrum Sharing. In Webb et al. [113].

[40] FEENEY, K., LEWIS, D., ARGYROUDIS, P., NOLAN, K., AND O'SULLIVAN, D. Grouping Abstraction and Authority Control in Policy-based Spectrum Management. In *Proceedings of 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (IEEE DySPAN'07)* (Dublin, Ireland, April 2007), IEEE, pp. 363–371.

[41] FERGUSON, N., AND SCHNEIER, B. *Practical Cryptography*. John Wiley & Sons, 2003.

[42] FERRARI, E., SQUICCIARINI, A. C., AND BERTINO, E. $\mathcal{X}$-tnl: An xml language for trust negotiations. *4th IEEE Workshop on Policies for Distributed Systems and Networks, Como, Italy* (June 2003).

[43] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO* (1986), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, Springer, pp. 186–194.

[44] GARG, D., BAUER, L., BOWERS, K. D., PFENNING, F., AND REITER, M. K. A linear logic of authorization and knowledge. In *ESORICS* (2006), D. Gollmann, J. Meier, and A. Sabelfeld, Eds., vol. 4189 of *Lecture Notes in Computer Science*, Springer, pp. 297–312.

[45] GREATER LONDON AUTHORITY. Report of the 7 July Review Committee, June 2006.

[46] GROTH, J. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, 2009. `http://eprint.iacr.org/`.

[47] HAN, S. C., AND XIA, Y. Optimal leader election scheme for peer-to-peer applications. In *ICN* (2007), IEEE Computer Society, p. 29.

[48] HERZBERG, A., MASS, Y., MICHAELI, J., RAVID, Y., AND NAOR, D. Access control meets public key infrastructure, or: Assigning roles to strangers. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2000), IEEE Computer Society, p. 2.

[49] HESS, A., JACOBSON, J., MILLS, H., WAMSLEY, R., SEAMONS, K. E., AND SMITH, B. Advanced client/server authentication in tls. In *NDSS* (2002), The Internet Society.

[50] HOLT, J. E., BRADSHAW, R. W., SEAMONS, K. E., AND ORMAN, H. K. Hidden credentials. In *WPES* (2003), S. Jajodia, P. Samarati, and P. F. Syverson, Eds., ACM, pp. 1–8.

[51] HOUSELY, R., FORD, W., POLK, T., AND SOLO, D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF Request for Comments RFC-2459, January 1999.

[52] HU, W., JIAN, N., QU, Y., AND WANG, Y. Gmo: A graph matching for ontologies. In Ashpole et al. [6].

[53] III, J. M., AND JR, G. Q. M. Cognitive radio: making software radios more personal. *IEEE Personal Communication 6*, 4 (1999), 13 – 18.

[54] JIAN, N., HU, W., CHENG, G., AND QU, Y. FalconAO: Aligning Ontologies with Falcon. In Ashpole et al. [6].

[55] JOHNSTONE, S., SAGE, P., AND MILLIGAN, P. ixchange - a self-organising super peer network model. In *ISCC* (2005), IEEE Computer Society, pp. 164–169.

[56] KATZ, J., AND LINDELL, Y. *Introduction to Modern Cryptography*. Cryptography and Network Security. Chapman & Hall/CRC, 2007.

[57] KAWULOK, L., ZIELINSKI, K., AND JAESCHKE, M. Trusted group membership service for jxta. In *International Conference on Computational Science* (2004), M. Bubak, G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds., vol. 3038 of *Lecture Notes in Computer Science*, Springer, pp. 218–225.

[58] KRAWCZYK, H. Secret sharing made short. In *CRYPTO* (1993), D. R. Stinson, Ed., vol. 773 of *Lecture Notes in Computer Science*, Springer, pp. 136–146.

[59] LEE, A. J. *Towards Practical And Secure Decentralized Attribute-Based Authorization Systems*. PhD thesis, University of Illinois at Urbana-Champaign, 2008.

[60] LEE, A. J., AND WINSLETT, M. Open problems for usable and secure open systems.

[61] LEE, A. J., WINSLETT, M., BASNEY, J., AND WELCH, V. The traust authorization service. *ACM Trans. Inf. Syst. Secur. 11*, 1 (2008).

[62] LEE, A. J., WINSLETT, M., AND PERANO, K. J. Trustbuilder2: A reconfigurable framework for trust negotiation. In *Proceedings of the Third IFIP WG 11.11 International Conference on Trust Management (IFIPTM 2009)* (June 2009).

[63] LEWIS, D., FEENEY, K., AND O'SULLIVAN, D. Integrating the Policy Dialectic into Dynamic Spectrum Management. In Webb et al. [113].

[64] LI, J., AND LI, N. Oacerts: Oblivious attribute certificates. *IEEE Trans. Dependable Sec. Comput. 3*, 4 (2006), 340–352.

[65] LI, N., AND MITCHELL, J. C. Datalog with constraints: A foundation for trust management languages. In *PADL* (2003), pp. 58–73.

[66] LYNCH, N. A. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[67] MASERA, M., AND NAI FOVINO, I. Modelling information assets for security risk assessment in industrial settings. In *EICAR* (2006).

[68] MCDERMOTT, J. P. Attack net penetration testing. In *NSPW '00: Proceedings of the 2000 workshop on New security paradigms* (New York, NY, USA, 2000), ACM, pp. 15–21.

[69] MERKLE, R. C. A digital signature based on a conventional encryption function. In *CRYPTO* (1987), C. Pomerance, Ed., vol. 293 of *Lecture Notes in Computer Science*, Springer, pp. 369–378.

[70] NAI FOVINO, I., AND MASERA, M. Emergent disservices in interdependent systems and system-of-systems. In *SMC* (October 2006), IEEE, Ed.

[71] NAI FOVINO, I., AND MASERA, M. A service oriented approach to the assessment of infrastructure security. In *First Annual IFIP Working Group 11.10 International Federation for Information Processing* (October 2006).

[72] NAI FOVINO, I., AND MASERA, M. Through the description of attacks: a multidimensional view. In *25th International Conference on Computer Safety, Reliability and Security* (2006).

[73] NEJDL, W., OLMEDILLA, D., AND WINSLETT, M. Peertrust: Automated trust negotiation for peers on the semantic web. In *Secure Data Management* (2004), W. Jonker and M. Petkovic, Eds., vol. 3178 of *Lecture Notes in Computer Science*, Springer, pp. 118–132.

[74] NEUMAN, C., YU, T., HARTMAN, S., AND RAEBURN, K. The Kerberos network authentication service (V5). IETF Request for Comments RFC-4120, July 2005.

[75] OAKS, S., GONG, L., AND TRAVERSAT, B. *JXTA in a Nutshell*. O'Reilly, 2002.

[76] ONTOLOGY MATCHING COMMUNITY. Ontology Matching Repository. Available on-line at: `http://www.ontologymatching.org/`.

[77] ORACLE. Java 1.6 Specification. Available on-line at: `http://download.oracle.com/javase/6/docs/`.

[78] ORACLE. MySQL database version 5.1. Available on-line at: `http://www.mysql.com`.

[79] ORACLE. Oracle database version 10gR2. Available on-line at `http://www.oracle.com`.

[80] ORAM, A. *Peer-To-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[81] PACI, F., BAUER, D., BERTINO, E., BLOUGH, D. M., AND SQUICCIARINI, A. C. Minimal credential disclosure in trust negotiations. In Bertino and Takahashi [14], pp. 89–96.

[82] PAWEŁCZAK, P., PRASAD, R. V., XIA, L., AND NIEMEGEERS, I. G. M. M. Cognitive radio emergency networks - requirements and design. In *Proceedings of First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (2005), IEEE Computer Society, pp. 601–606.

[83] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO* (1991), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, Springer, pp. 129–140.

[84] POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL Database. Available on-line at `http://www.postgresql.org`.

[85] QU, Y., HU, W., AND CHENG, G. Constructing virtual documents for ontology matching. In *WWW* (2006), L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, Eds., ACM, pp. 23–31.

[86] RAMAKRISHNA, V., EUSTICE, K., AND REIHER, P. L. Negotiating agreements using policies in ubiquitous computing scenarios. In *SOCA* (2007), IEEE Computer Society, pp. 180–190.

[87] RUOHOMAA, S., AND KUTVONEN, L. Trust management survey. In *iTrust* (2005), pp. 77–92.

[88] SACHA, J., DOWLING, J., CUNNINGHAM, R., AND MEIER, R. Using aggregation for adaptive super-peer discovery on the gradient topology. In *SelfMan* (2006), A. Keller and J.-P. Martin-Flatin, Eds., vol. 3996 of *Lecture Notes in Computer Science*, Springer, pp. 73–86.

[89] SAFECOM. Public safety Statements of Requirements for communications and interoperability volumes I and II. Tech. rep., U.S. Department of Homeland Security's Office for Interoperability and Compatibility, 2006.

[90] SCHNEIER, B. Modeling security threats. *Dr. Dobb's Journal* (2001).

[91] SEAMONS, K. E., WINSLETT, M., AND YU, T. Limiting the disclosure of access control policies during automated trust negotiation. In *NDSS* [1].

[92] SEAMONS, K. E., WINSLETT, M., YU, T., SMITH, B., CHILD, E., JACOBSON, J., MILLS, H., AND YU, L. Requirements for policy languages for trust negotiation. In *POLICY* (2002), pp. 68–79.

[93] SEAMONS, K. E., WINSLETT, M., YU, T., YU, L., AND JARVIS, R. Protecting privacy during on-line trust negotiation. In *Privacy Enhancing Technologies* (2002), R. Dingledine and P. F. Syverson, Eds., vol. 2482 of *Lecture Notes in Computer Science*, Springer, pp. 129–143.

[94] SECURITY FOCUS. Bugtraq Home Page. Available on-line at `http://securityfocus.com`.

[95] SHAMIR, A. How to share a secret. *Commun. ACM 22*, 11 (1979), 612–613.

[96] SOFTWARE DEFINED RADIO FORUM. Software defined radio technology for public safety formerly approved document. Tech. Rep. SDRF-06-P-0001-V1.0.0, Software Defined Radio Forum, 2006. Formerly Approved Document.

[97] SOFTWARE DEFINED RADIO FORUM. Use Cases for Cognitive Applications in Public Safety Communications Systems Volume 1: Review of the 7 July Bombing of the London Underground. Tech. Rep. SDRF-07-P-0019-V1.0.0, Software Defined Radio Forum, November 2007.

[98] SOFTWARE DEFINED RADIO FORUM. Use Cases for MLM Language in Modern Wireless Networks. Tech. Rep. SDRF-08-P-0009-V1.0.0, Software Defined Radio Forum, January 2009.

[99] SQLITE COMMUNITY. SQLite Database version 3. Available on-line at: `http://www.sqlite.org`.

[100] SQUICCIARINI, A. C. *Trust and Privacy-Preserving Methodologies in Distributed Systems*. PhD thesis, Università degli Studi di Milano, November 2005.

[101] SQUICCIARINI, A. C., BERTINO, E., FERRARI, E., PACI, F., AND THURAISINGHAM, B. M. PP-Trust-$\mathcal{X}$: A system for privacy preserving trust negotiations. *ACM Trans. Inf. Syst. Secur. 10*, 3 (2007).

[102] SQUICCIARINI, A. C., BERTINO, E., TROMBETTA, A., AND BRAGHIN, S. A Flexible and Secure Rule-Based Approach to Trust Negotiation. *Submitted to IEEE Transactions on Dependable and Secure Computing*.

[103] SQUICCIARINI, A. C., PACI, F., AND BERTINO, E. Trust establishment in the formation of virtual organizations. In *ICDE Workshops* (2008), IEEE Computer Society, pp. 454–461.

[104] SQUICCIARINI, A. C., PACI, F., BERTINO, E., TROMBETTA, A., AND BRAGHIN, S. Group-based negotiations in p2p systems. *IEEE Trans. Parallel Distrib. Syst. 21*, 10 (2010), 1473–1486.

[105] SQUICCIARINI, A. C., TROMBETTA, A., AND BERTINO, E. Supporting robust and secure interactions in open domains through recovery of trust negotiations. In *ICDCS* (2007), IEEE Computer Society, p. 57.

[106] SQUICCIARINI, A. C., TROMBETTA, A., BERTINO, E., AND BRAGHIN, S. Identity-based long running negotiations. In Bertino and Takahashi [14], pp. 97–106.

[107] STALLINGS, W. *Cryptography and Network Security*, second ed. Prentice Hall, 1998. ISBN 0-13-869017-0.

[108] STEFAFN, J., AND SCHUMACHER, M. Collaborative attack modeling. In *Symposium on Applied Computing* (2002), pp. 253–259.

[109] STINE, J. A., AND PORTIGAL, D. L. Spectrum 101. an introduction to. Tech. Rep. MTR 04W0000048, MITRE, 2004.

[110] SUN MICROSYSTEMS. JXTA Documentation. Available on-line at: `https://jxta-docs.dev.java.net/`.

[111] WANG, T., TSAI, K., AND LEE, Y. Crown: An efficient and stable distributed resource lookup protocol. In *Embedded and Ubiquitous Computing*, L. T. Yang, M. Guo, G. R. Gao, and N. K. Jha, Eds., vol. 3207 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 211–222. 10.1007/978-3-540-30121-9-103.

[112] WANG, W., GAO, W., BAI, X., PENG, T., CHUAI, G., AND WANG, W. A framework of wireless emergency communications based on relaying and cognitive radio. In *In Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)* (2007), IEEE Computer Society, pp. 1–5.

[113] WEBB, W., JONDRAL, F., MARSHALL, P., CAVE, M., LEHR, W., AND BUDDHIKOT, M., Eds. *2nd International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (April 2007), IEEE Computer Society.

[114] WEEKS, S. Understanding trust management systems. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2001), IEEE Computer Society, p. 94.

[115] WINSBOROUGH, W. H., AND JACOBS, J. Automated trust negotiation in attribute-based access control. In *DISCEX (2)* (2003), IEEE Computer Society, pp. 252–.

[116] WINSBOROUGH, W. H., AND LI, N. Towards practical automated trust negotiation. In *POLICY* (2002), IEEE Computer Society, pp. 92–103.

[117] WINSBOROUGH, W. H., AND LI, N. Safety in automated trust negotiation. *ACM Trans. Inf. Syst. Secur. 9*, 3 (2006), 352–390.

[118] WORLD WIDE WEB CONSORTIUM. Extensible Markup Language (xml) 1.0, 1998. Available on-line at: `http://www.w3.org/TR/REC-xml`.

[119] WORLD WIDE WEB CONSORTIUM. OWL Web Ontology Language, 2004. Available on-line at: `http://www.w3.org/TR/owl-ref/`.

[120] WORLD WIDE WEB CONSORTIUM. XML Schema Definition Language (xsd) 1.1 Part 2: Datatypes. Available on-line at: `http://www.w3.org/TR/xmlschema11-2/`.

[121] WORLD WIDE WEB CONSORTIUM. XSL Transformations, version 1.0. Available on-line at: `http://www.w3.org/TR/xslt/`.

[122] WORLD WIDE WEB CONSORTIUM. Gleaning resource descriptions from dialects of languages (grddl). Available on-line at `http://www.w3.org/TR/grddl/`, September 2007.

[123] WORLD WIDE WEB CONSORTIUM. GRDDL Use Cases: Scenarios of extracting RDF data from XML documents. Available on-line at `http://www.w3.org/TR/grddl-scenarios/`, April 2007.

[124] XIONG, L., AND LIU, L. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng. 16*, 7 (2004), 843–857.

[125] YEAGER, W., AND WILLIAMS, J. Secure peer-to-peer networking: The jxta example. *IT Professional 4* (2002), 53–57.

[126] YU, T. *Automated Trust Establishment in Open Systems*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.

[127] YU, T., AND WINSLETT, M. A unified scheme for resource protection in automated trust negotiation. In *IEEE Symposium on Security and Privacy* (2003), IEEE Computer Society, pp. 110–122.

[128] ZERR, S., OLMEDILLA, D., COI, J. L. D., NEJDL, W., BONATTI, P. A., AND SAURO, L. Policy based protection and personalized generation of web content. In *LA-WEB/CLIHC* (2009), E. Chávez, E. Furtado, and A. L. Morán, Eds., IEEE Computer Society, pp. 112–119.