IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA ITB10573

Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

Chapter VI

A Comprehensive XML Based Approach to Trust Negotiations

Elisa Bertino, Purdue University, USA

Elena Ferrari, Università degli Studi dell'Insubria, Italy

Anna Cinzia Squicciarini, Università degli Studi di Milano, Italy

Abstract

Trust negotiation is a promising approach for establishing trust in open systems like the Internet, where sensitive interactions may often occur between entities at first contact, with no prior knowledge of each other. In this chapter we present Trust-X, a comprehensive XML-based XML framework for trust negotiations, specifically conceived for a peer-to-peer environment. We also discuss the applicability of trust negotiation principles to mobile commerce. We introduce a variety of possible approaches to extend and improve Trust-X in order to fully support mobile commerce transactions and payments. In the chapter, besides presenting the Trust-X system, we present the basic principles of trust negotiation.

Introduction

Computer systems have traditionally had centrally managed security domains. Every entity that can access such systems has one or more identities in that domain. The underlying assumption is that entities in the system already know each other. Therefore, the system relies on party identities to grant or deny authorizations.

As we move towards a globally Internetworked infrastructure, like the Internet, interactions involving strangers are dramatically increasing. In particular, transactions between companies and their cooperating partners or customers are becoming of everyday use. Furthermore, advances in technology enable users to perform commerce transactions through the use of mobile systems, adding new requirements to the traditional scenario. Nowadays, companies of all sizes are able to conduct business without worrying about the territorial market limitations of the past. In such a complex scenario, traditional assumptions for establishing and enforcing access control regulations no longer hold. The entities need not only to authenticate each other, but also to trust each other in order to exchange sensitive information and resources. Interactions are further complicated by the fact that usually the interacting entities belong to different security domains, or can change domains during a transaction if they are mobile users, and/or do not have any pre-existing relationships.

Traditional attempts to establish trust in open systems either minimize security measures or assume that parties are not strangers and can present a local identity to obtain services. According to such paradigm each subject is uniquely identified by an ID (e.g., login name, IP address) that is the means for proving the subject's trustworthiness. However, identity-based methods for establishing trust are not feasible in an environment like the Web. In such an environment, properties other than identity are crucial in determining parties' trustworthiness.

A promising approach in this respect is represented by trust negotiation (TN) (Seamons & Winslett, 2001), according to which trust is established through a mutual exchange of *digital credentials*. Disclosure of credentials, in turn, must be protected by the use of policies specifying which credentials must be received before the requested credential can be disclosed.

A trust negotiation system, thus, relies on digital credentials held by the negotiating parties, with the goal of establishing mutual trust before completing the transaction. This approach allows parties having no pre-existing relationships to confidently perform sensitive interactions.

One of the most interesting applications for trust negotiation systems is represented by e-commerce applications. An e-commerce application typically carries out commercial transactions on the Web, such as buying and selling products, or various other activities, such as supply chain management. Trust negotiation systems represent a powerful means to conduct business transactions, very often characterized by the fact that the interacting entities are unknown to each other and need to establish a sufficient level of trust to complete the transaction. Mobile commerce, in particular, is an important branch of e-commerce requiring additional trust establishment capabilities. In a nutshell, mobile commerce provides consumers with secure, faster and personalized services and is

becoming one of the most important wireless applications. Mobile commerce is a vast area of activity comprised of transactions with monetary value conducted via a mobile device. More and more people prefer m-commerce services and truly enjoy these prompt services.

Although the problem of trust negotiations performed using typical desktop computers has been thoroughly explored, the issue of negotiations involving mobile devices is still an unexplored research area. This is a promising and challenging research area, as trust negotiation systems have a number of features that might be exploited to develop efficient and powerful mobile negotiation systems for conducting business transactions.

This chapter is devoted to present the basic principles of trust negotiation and its basic building blocks. Then, as an example of trust negotiation system, we present Trust-*X*, a framework we have developed providing a comprehensive solution to trust negotiation management. Trust-*X* provides both an XML based language for expressing policies and credentials and a methodology and related algorithms for carrying on negotiations. We end the chapter by discussing the applicability of trust negotiation principles to mobile commerce. We introduce a variety of possible approaches to extend and improve Trust-*X* in order to fully support mobile commerce transactions and payments. In presenting such approaches we refer to a set of open issues we have identified and that have to be taken into account while redesigning the system.

More precisely, this chapter is organized as follows. Next section summarizes basic concepts underlying trust negotiation. The following two sections are devoted to Trust-*X*. Then, we survey related work and compares Trust-*X* with some of the most relevant proposals in the negotiation area. Finally, we discuss the use of Trust-*X* in mobile commerce applications and identify open research issues, and conclude the chapter by outlining future research directions. The chapter also contains an appendix, reporting formal proofs.

Trust Negotiation: Basic Concepts

A trust negotiation consists of a bilateral disclosure of digital credentials, representing statements certified by given entities, which can be used for verifying properties of their holders. Typically, a trust negotiation involves two entities, namely a *client*, that is, the entity asking for a certain resource, and a *server*, that is, the entity owning (or more generally, managing access to) the requested resource. The notion of *resource* comprises both sensitive information and services, whereas the notion of entity includes users, processes, roles, and servers. Resource disclosure is protected by a set of *policies*.

A trust negotiation is basically peer-to-peer: both negotiation entities may possess sensitive resources to be protected and thus must be equipped with a compliant negotiation system. Trust is incrementally built by iteratively disclosing *digital credentials* in order to verify properties of the negotiating parties. Credentials are typically collected by each party in appropriate repositories, also named profiles. Disclosure

policies govern access to protected resources by specifying credential combinations that must be submitted in order to obtain authorizations. The overall interaction process between parties is usually carried out through software components such as browsers, user agents, and wrappers (Subrahmanian et al., n.d.).

The fundamental elements of trust negotiations are thus digital credentials and policies, discussed in the remainder of this section.

Digital Credentials

Digital credentials are assertions describing one or more properties about a given subject, referred to as the "owner," certified by trusted credential authorities (CAs). Entities are thus identified and described through a set of *digital credentials*.

Like paper credentials that subjects carry in their wallets, digital credentials state properties about their owners. Typically, a digital credential contains a set of attributes, specified using name/value pairs, that are signed by the issuer's private key and can be verified using the issuer's public key (Stallings, 1999). To achieve unforgeability and verifiability, digital credentials are usually implemented using the X.509 V3 standard for public key certificates. However, since the X.509 certificates were not conceived for online negotiations they do not efficiently support either the notion of attribute or protect privacy. As a result, other formats have been recently proposed that can better support entities' property description (Bertino & Ferrrari, 2003). An example of these proposals will be presented later on in the chapter. Finally, in order to increase privacy guarantee and non-forgeability, alternative certificate formats (Brands, 2000; Persiano & Visconti, 2000) have also been developed. These approaches rely on the possibility of selectively disclosing attributes within credentials, so that only the required subset is disclosed to the counterpart. In particular, to achieve this goal Persiano et al. in 2002 introduced the SPSL protocol extending the transport layer security protocol, whereas Brands introduces a number of techniques for designing private credentials and protocols for issuing and disclosing private credentials.

Disclosure Policies

To automate trust negotiation, each party must specify disclosure policies to regulate access to sensitive resources such as services, data, credentials or even policies themselves. Policies are usually expressed as constraints against the credentials of the interacting parties and their attributes. Further, depending on their contents, credentials themselves may be sensitive and thus their disclosure may be regulated by ad-hoc disclosure policies. For example, a credential may contain non-public attributes about an individual, such as a credit card number.

Disclosure policies, in turn, may be regarded as sensitive information because they are often related to the business and governance processes of organizations. For example, the Web site for a secret joint venture of two companies might be protected by policies that limit access to particular employees of those companies. Outsiders who access the

policies may infer information about the type of engagement between the companies, and take advantage of such information. Therefore, recent researchers consider disclosure policies as sensitive as other resources and provide mechanisms for their protection (Winslett & Seamons, 2002).

Besides general purpose policy languages (Damianou, Dulay, Lupu & Sloman, 2001), a number of policy languages especially conceived for encoding security information used in trust negotiations have been proposed (Bonatti & Samarati, 2000; Herzberg & Mihaeli, 2000; Winsborough & Li, 2002). An example of trust negotiation policy language is given in the next sections, where we present *X*-TNL, the XML based language supported by Trust-*X*.

Trust-*X*: A Comprehensive Framework for Trust Negotiations

Trust-*X* is a system providing a comprehensive approach to all aspects of a negotiation. Building blocks of Trust-*X* are an XML-based language, named *X*-TNL, for specifying Trust-*X* certificates and policies, and an architecture and related algorithms for managing negotiations. In Bertino and Ferrari (2003) we have presented the language for specifying certificates. In this chapter we mainly focus on disclosure policies and present the Trust-*X* architecture for negotiation management. A Trust-*X* negotiation consists of a set of phases that are sequentially executed. Trust-*X* negotiations maximize the protection of the involved resources; indeed, certificates and services are disclosed only after a complete counterpart policy evaluation, that is, and only when the parties have found a sequence of certificate disclosures that makes it possible to release the requested resource. Additionally, both parties can drive the steps of the negotiation by selecting the adopted strategy from a variety of alternative strategies, and thus can better trade-off protection and efficiency.

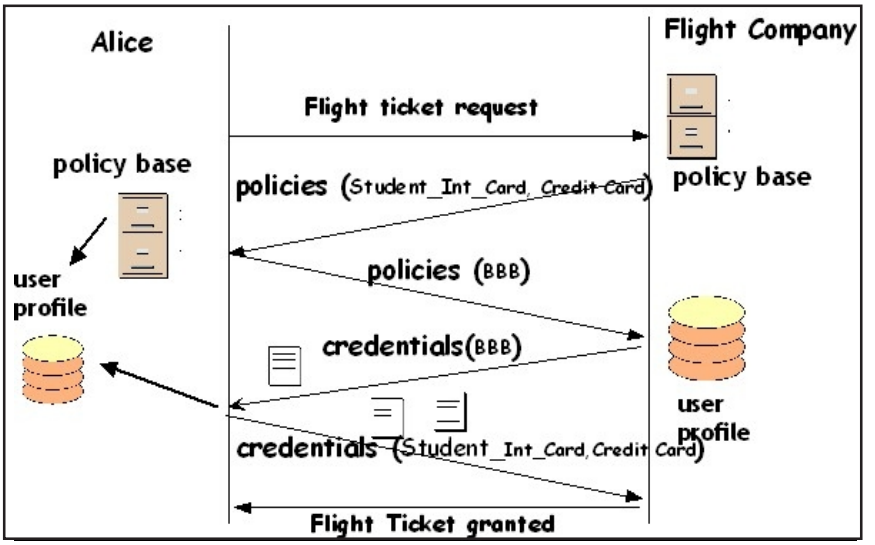
An Example of Trust Negotiation

We now introduce a basic example of a trust negotiation, which we will use throughout this section as a running example to illustrate the basic functions of Trust-*X* and its main features.

KTH is a flight company selling electronic flight tickets to travel around Europe. We assume that KTH and all the entities interacting with KTH are characterized by a profile of certificates, describing properties of the owners. Additionally, each party has specified a set of disclosure policies to protect credentials and services.

Alice is a student wishing to purchase a flight ticket to KTH airline. As Alice browses the site, she fills out the booking form on the Web, checking a form box to indicate that

Figure 1. Sketch of a trust negotiation process (In the example, negotiating participants perform a TN in which the client obtains a service after exchanging policies and credentials with the server.)



she wishes to take advantage of a special offer. Upon receipt of the reservation request, KTH asks for a valid credit card or her account credential, issued by KTH as a frequent flyer, and a current international student card. Alice has both the account with KTH and a valid credit card. Suppose she does not want to use her account as frequent flyer. She is willing to disclose her student card to anyone under certain privacy guarantees, but she will only show her credit card to members of the Better Business Bureau. Alice may specify such security requirements by means of disclosure policies, exchanged following the protocol sketched in Figure 1. The same can do KTH to express the requirements needed to sell a flight ticket. In the following, we will show how this negotiation can be carried on in the framework of Trust-*X*.

X-TNL Trust Negotiation Language

In this section, we present *X*-TNL (Bertino & Ferrari, 2003), the XML-based language we have developed for specifying Trust-*X* certificates and policies. The language provides a flexible way of qualifying the parties involved in the negotiation, which relies on a distinction between *credentials* and *declarations*. Additionally, it provides an expressive XML encoding of *disclosure policies*, where a disclosure policy regulates the disclosure of a resource by imposing conditions on both credentials and negotiations.

The language we provide has been especially conceived for handling multiple and heterogeneous credentials and it is flexible enough to express a wide range of protection requirements.

We first present the credential language, that is, the language encoding credentials and declarations. Then, we present *X*-TNL *disclosure policies*, that is, policies regulating the disclosure of resources by imposing conditions on the certificates the requesting party should possess.

Credential Language

Constructs of *X*-TNL include the notion of *certificate*, which is the means to convey information about the profile of the parties involved in the negotiation. A certificate can be either a *credential* or a *declaration*. A credential is a set of properties of a party certified by a certification authority, whereas declarations contain information that may help the negotiation process, such as for instance specific preferences of one of the parties, but that do not need to be certified.

As far as credentials are concerned, *X*-TNL simplifies the task of credential specification because it provides a set of templates, called *credential types*, for the specification of credentials with similar structure. In *X*-TNL, a credential type is modelled as a DTD and a credential as a valid document with respect to the corresponding credential type. Each credential is digitally signed by the issuer credential authority, according to the standard defined by W3C for XML signatures. A credential is an instance of a credential type, and specifies the list of property values characterizing a given subject. A Trust-*X* credential is thus a valid XML document conforming to the DTD modelling of the corresponding credential type. Figure 2 shows an example of credential, containing the basic information about a Frequent_Traveller. Note that, as each Trust-*X* credential, the Frequent_Traveller credential has a set of default attributes, namely, SENS, CREDID and CIssuer. The CREDID and CIssuer attributes specify the credential identifier and the identity of the issuer of the credential, respectively. By contrast, the SENS denotes the degree of sensitivity of the information contained in the credential. This attribute takes values from a set ν of sensitivity levels, defined according to the considered domain. Throughout the chapter, we assume $\nu = \{\text{HIGH}, \text{NORMAL}, \text{LOW}\}$.

By contrast, declarations are sets of data without any certification; therefore they are stated by their owner. Declarations can be considered as self-certificates, collecting personal information about the owner. This kind of certificate thus provides auxiliary information that can help the negotiation process. For instance, a declaration named customer_info may describe the habits of a given subject for what concern travels.

In *X*-TNL, we simply define a declaration as a valid XML document. Like credentials, also declarations are structured into declaration types, that is, DTDs to which the corresponding declarations conform. Figure 3 shows the Trust-*X* representation of the customer_info declaration. The declaration describes Alice's personal information about her travels with KTH airlines. This declaration can be used to communicate Alice's personal preferences during negotiation with a KTH Airline.

Figure 2. An example of Trust-X credential

```
<FREQUENT_TRAVELLER  CREDID='12AB', SENS= 'NORMAL'>
<ISSUER HREF='HTTP://WWW.CORRIER.COM'
TITLE=CORRIER\_EMPLOYEES_REPOSITORY/>
      <NAME>
         <FNAME> ALICE  </FNAME>
         <LNAME>  WHITE </LNAME>
     </NAME>
     <ADDRESS> GRANGE WOOD  69  DUBLIN     </ADDRESS>
   <CITIZENSHIP> IRISH </CITIZENSHIP>
     <CARD\_NUMBER CODE=34ABN/>
       <E_MAIL> O.WHITE@YAHOO.COM </E_MAIL>
        <DEPARTMENT>  AGENCY 45 </DEPARTMENT>
  <POSITION> DRIVER </POSITION>
 </FREQUENT_TRAVELLER>
```

Figure 3. An example of Trust-X declaration

```
<customer_info>
<name>
<Fname> Alice </Fname>
 <lname >   White</lname>
 </name>   <flight_class> business traveller </flight_class>
 <meal_preferences >
< vegetarian>
</meal_preferences >
<preferred_travelling_time> 9AM-8PM </preferred_travelling_time>
 <favorite_route >   Dublin-Rome </favorite_route>
  <collected_miles> 1220   </collected_miles>
</customer_info>
```

Data Sets and *X*-Profiles

All certificates associated with a party are collected into its *X*-Profile. To better structure credentials and declarations into an *X*-Profile, each *X*-Profile is organized into *data sets*. Each data set collects a class of credentials and declarations referring to a particular aspect of the life of their owner. For instance, Demographic Data, Education, and Working Experience are examples of possible data sets.[1] For example, Alice's certificates concerning working experiences can be collected in the Working Experience data set. In this group of digital documents we can find Alice's work license number, a digital copy of her last job contract and some uncertified information about her precedent job experiences. Organizing certificates into data sets facilitates their retrieval during negotiation because

all certificates collected in the same data set are logically related. Data sets can then be used to refer to a set of homogeneous declarations or credentials as a whole, and this can facilitate their evaluation and exchange during negotiation.

Disclosure Policies Language

Trust-*X* disclosure policies are specified by each party involved in a negotiation, and state the conditions under which a resource can be released during a negotiation. Conditions are expressed as constraints against the certificates possessed by the involved parties and on the certificate attributes. Each party adopts its own policies to regulate release of local information and access to services. Like certificates, disclosure policies are encoded using *X*-TNL (Bertino, Ferrari & Squicciarini, 2003). Additionally, Trust-*X* policies can also be formalized as logical rules. In the following, we present such logical representation, since it makes easier explaining the compliance checker mechanisms and runtime system algorithms.

Before introducing the notion of disclosure policy, we need to introduce some preliminary concepts. We first introduce the notion of *R-Term*. An R-Term univocally denotes a resource offered by a party. A resource can be either a certificate, or a service. By service we mean either an application that the requesting party can execute, for instance for purchasing goods, or an access to protected data, such as for instance medical data. Formally, an R-Term can be considered as a structured object identified by a name and some properties, and is modelled as an expression of the form *resource_name(attribute_list)*, where *resource_name* is the name of the resource, and *attribute_list* is a possible empty list of attribute names characterizing the resource. If the resource is a certificate-type the list of properties consists of the attribute and tag names contained in its XML encoding. Resource properties are used to express constraints on the resource release when specifying disclosure policies. We use the dot notation to refer to a specific attribute of a resource, that is, we use *R.a* to denote attribute *a* of a resource *R*. Expressions of the form *R.a* are called *resource expressions*.

Example 3.1

Examples of R-Terms for our running example are:

1. Flight_Ticket(customerCode, from, to, departure, return, class):

   it denotes an online flight ticket buying service. The service is characterized by a set of attributes required to customize the purchasing, such as the requester code, (if any), the route (attributes from and to), the travelling days (departure date and return date), and the flight category.
2. Frequent_Traveller():

   it denotes the credential type Frequent_Traveller.

We now review the notion of certificate conditions and terms. These concepts have been already presented in our previous work (Bertino, Ferrrari & Squicciarini, 2004) to formally define the Trust-*X* policy language. Informally, certificate conditions and terms can be regarded as the building blocks used to compose disclosure policies. More precisely, a certificate condition **C** is an expression of the form **a op expr,** where: **a** denotes an element tag or an attribute name in a certain credential type; **op** is a comparison operator, such as $<, >, =, \neq, \geq$; **expr** can be either a constant or a resource expression, compatible with the type of **a**. Terms, in turn, are expressions of the form **P(C)** or **P( )** where: **P** is a Trust-*X* certificate type; and **C** is a list of certificate conditions $C_1...C_n$ against **P**. The form **P( )** denotes a term without conditions.

Example 3.2

The following are examples of terms:

- T_1=CreditCard(Release_year > 1998);
- T_2=Frequent_Traveller(code=Flght_Ticket.customerCode).
- T_3=Id_Card().

T_1 is a term denoting a credit card, containing a certificate condition against the Release_year attribute. Similarly, T_2 is a term for the Frequent_Traveller credential, specifying a credential condition against attribute *code*. Finally, the last term denotes a term without any conditions, that is, *Id_Card*.

In the remainder of the chapter, we say that a certificate *X*-Cert *satisfies* a term *P(C)*, if *X*-Cert is of type *P* and satisfies all the conditions specified in *C*. Additionally, given a term, we use the notation P(*T*) to denote the certificate type in *T*, and C(*T*) to denote the certificate conditions in *P*.

We are now ready to formally define disclosure policies.

Definition 3.1.

Disclosure Policies: A disclosure policy is an expression of one of the following forms:

1. **R**$\leftarrow T_1, T_2, \ldots, T_n$, $n \geq 1$, where $T_1, T_2, \ldots, T_n$ are terms and **R** is the Resource_name component of an R-Term.
2. **R**$\leftarrow$**DELIV,** where **R** is the Resource_name component of an R-Term. These policies are called *delivery policies*.

A disclosure policy specifies which kind of certificates a party should possess in order to obtain access to a resource owned by the other party. Delivery policies are specified for resources that do not contain sensitive information, and can be released whenever requested.

Example 3.3

Consider the negotiation sketched in. As already mentioned, a special fare is offered either to frequent travellers or to students. Suppose that the server already knows frequent customers possessing the Frequent_Traveller credential and has a digital copy of their credit cards. By contrast, flight tickets are available on payment for unknown customers, who have to submit a digital copy of their student international card issued by a state member of EU to obtain a special fare, and a valid credit card. These requirements can be formalized by the following disclosure policies:

- Flight_Ticket←Frequent_Traveller(code=Flght_Ticket.customerCode);
- Flight_Ticket←Student_Int_Card(age<25,issuer=EU),Credit_Card (ExpirationDate>Flight_Ticket.ReturnDate).

In the remainder of this chapter we say that a disclosure policy R←$T_1, T_2, \ldots, T_n$ specified by one of the parties involved in the negotiation is satisfied if the right side elements of the policy are all satisfied by the counterpart *X*-Profile.

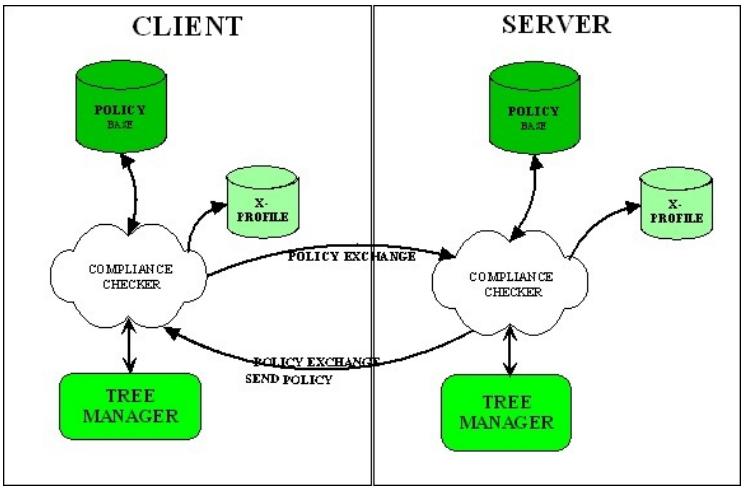
Trust-*X* policies are thus defined for protecting both services and certificates. Indeed, the left side element of a disclosure policy can denote either a service identifier or a certificate type. Different expressions having the same element R on the left side denote alternative policies equally valid to obtain R. Terms on the right side of a policy specify conditions for the release of R. Each resource R can be disclosed only if one of the corresponding policies is satisfied. In addition, the disclosure policy language may be adopted to define prerequisite information. Such policies denote conditions that must be satisfied for a resource request to be taken into consideration, and are therefore used at the beginning of the negotiation process, as explained in the next sections.

Trust-*X* Architecture

As shown in Figure 4, Trust-*X* is composed by several components. Further, Trust-*X* architecture is symmetric and peer-to-peer; therefore the terms *client* and *server* are simply used as a convenient way for distinguishing parties during negotiations. The goals of the system components are essentially the following: supporting policy exchange, testing whether a policy is satisfied, and supporting certificate exchange. Each of those functions is executed by a specific module of Trust-*X*. Facet modules may also be added to make the negotiation easier and faster, but we omit them to focus on the most relevant components. The system is composed of a *policy base*, storing disclosure policies, the *X*-Profile associated with the party, a *tree manager*, storing the state of the negotiation, and a *compliance checker*, to test policy satisfaction and determine request replies.

Each negotiation participant has a Trust-*X* profile of certificates, conforming to the *X*-TNL syntax summarized in the previous section. Unlike traditional approaches, during a

Figure 4. Trust-X architecture



negotiation mutual trust might be established between the client and the server: the client has to show its certificates to obtain the resource, and the server, whose honesty is not always assured, submits certificates to the client in order to prove its trustworthiness before receiving sensitive certificates. Disclosure of information and certificates is regulated by disclosure policies, which are exchanged to inform the other party of the trust requirements that need to be satisfied to advance the state of the negotiation. We elaborate on the trust negotiation process in the following section.

Trust-*X* Negotiations

In this section we focus on the approach used in Trust-*X* for policy disclosures during negotiation. Trust-*X* adopts a cautious strategy, distinguishing between policy exchange and certificates and/or resource disclosure. This distinction results in an efficient and effective protection of all the resources involved during negotiations. Certificates and services are disclosed only after a complete counterpart policies evaluation; that is, only when the parties have found a sequence of certificate disclosure that makes possible the release of the requested resource, according to the disclosure policies of both parties. In the following, we assume that both parties are Trust-*X* compliant. However, it is also possible to carry on negotiations even between parties that do not adopt the same negotiation language, simply by adding a translation mechanism. A Trust-*X* negotiation is organized according to the following phases:

- *Introduction*. It is the starting phase of a negotiation. A client contacts a server requiring a resource *R*. The server may react by asking *prerequisite information*, if necessary. Prerequisite information is general conditions that must be satisfied

to start processing the resource request, independently from the result of the request. Moreover, these conditions are usually independent also from the requested resource. For instance, a server providing services only to registered clients, before evaluating the requirements for the requested service, can ask the counterpart for the login name. If the client is not registered there is no reason to further proceed. The introductory phase may also be used to collect information about the client preferences or needs. For instance, in the Flight ticket scenario, the server may ask the customer to submit the customer_info declaration, if any, in order to better satisfy client requirements.[2] If the client does not assume honesty of the server it can, in turn, ask some prerequisite information to the server. Such a phase is therefore composed of a small number of simple messages exchanged between the two parties.

- *Policy evaluation*. During this phase, both client and server communicate disclosure policies adopted for the involved resources. The goal is to determine a sequence of client and server certificates that when disclosed allow the release of the requested resource, in accordance to the disclosure policies of both parties. This phase is carried out as an interplay between the client and the server. During each interaction one of the two parties sends a set of disclosure policies to the other. The receiver party verifies whether its *X*-Profile satisfies the conditions stated by the policies, and whether its local policy base contains policies regulating the disclosure of the certificates requested by the policies sent by the other party. If the *X*-Profile of the receiver party satisfies the conditions stated by at least one of the received policies, the receiver can adopt one of two alternative strategies. It can choose to maximize the protection of its local resources, by replying to only one policy at a time, thus hiding the real availability of the other requested resources. Alternatively, it can reply to all the policies in order to maximize the number of potential solutions for negotiation and thus speed up the overall process. Otherwise, if the *X*-Profile of the receiver party does not satisfy the conditions stated by the received policies, the receiver informs the counterpart that it does not possess the requested certificates. The counterpart then sends an alternative policy, if any, or halts the process if no other policies can be found. The interplay goes on until one or more potential solutions are determined; that is, whenever both client and server determine one or more set of policies that can be satisfied for all the resources and certificates involved. The policy evaluation phase is mostly executed by the *compliance checker,* whose goal is the evaluation of remote policies with respect to local policies and certificates (certificates can be locally available in the *X*-Profile or can be retrieved through certificate chains), and the selection of the strategy for carrying out the remainder of the negotiation. To simplify the process a tree structure is used, explained in detail in the next section, which is managed and updated by the *tree manager*. Note that no certificates are disclosed during the policy evaluation phase. The satisfaction of the policies is only checked to communicate to the other party the possibility of going on with the process and how the process can actually be executed.
- *Certificate exchange*. This phase begins when the policy evaluation phase determines one or more *trust sequences*[3] to successfully complete the negotiation. A trust sequence determines a list of sets of certificates where the disclosure of

each set of certificates in the list represents a condition for a trust release of the certificates following it in the list. Several trust sequences can be determined for the same negotiation and several criteria can then be used by both the client and the server to select one of the possible trust sequences. Examples of these criteria include the number of involved certificates, the sensitivity of their content, the expected length of the negotiation, or the number of certificate chains that need to be traversed. Once the parties have agreed on a sequence, the certificate exchange phase begins. Each party discloses its certificates, following the order defined in the trust sequence, eventually retrieving those that are not immediately available through certificate chains. Functions required to carry out certificate disclosure are: verification of certificate contents, check for revocation, check validity dates, and authentication of ownership (for credentials). The process ends with the disclosure of the requested resource or, if any unforeseen event happens, with an interruption. If the failure is caused by dishonest behavior of one of the parties, for instance a party discloses a revoked certificate, the negotiation fails. Otherwise, if it is due to events not related with parties' trustworthiness, for instance interruption of connection, the negotiation is restarted, repeating certificates exchange. If it is not possible to complete the certificate exchange for the interrupted sequence, one of the alternative trust sequences determined at the beginning of this phase is chosen.

Note that there is a significant difference between the first and the other two phases of a Trust-*X* negotiation. The introductory phase is executed following a static protocol, since it is simply a fixed exchange of information that is necessary for starting any negotiation involving the considered parties. By contrast, the second and the beginning part of the third phase are dynamic and may evolve in several ways.

Next two sections are thus devoted to the policy evaluation phase and the certificate exchange phase, since they are the most complex and interesting phases of the negotiation process.

Policy Evaluation Phase

In this section, we focus on the key phase of a Trust-*X* negotiation, that is, the policy evaluation phase. This phase consists of a bilateral and ordered policy exchange. The compliance checker module of each party, upon receiving a disclosure policy, determines if it can be satisfied by querying the local *X*-Profile. Then, it checks in its policy base the protection requirements associated with the certificates satisfying the policy, if any. The progress of a negotiation is recorded into a specific data structure, called *negotiation tree*, managed by the *tree manager*, which is described in the next section.

Negotiation Tree

A negotiation tree specifies a set of negotiation paths, where each path denotes a possible trust sequence. The path also keeps track of which certificates may contribute to the success of the negotiation, and of the correct order of certificate exchange.

Upon the end of the introductory phase, each party maintains a copy of a negotiation tree, rooted at the requested resource *R*. The policy evaluation phase ends when at least one trust sequence (corresponding to a path in the tree) is found or there is no compatibility between the policies of the parties. If no trust sequence can be determined the phase ends with a failure message. Otherwise, the subsequent phase is executed.

In defining a negotiation tree we make use of a function, called *Eval*, that receives as input a term *T* and an *X*-Profile, and returns TRUE if the *X*-Profile contains a certificate satisfying *T*, and FALSE otherwise.

Definition 4.1.

Negotiation Tree: Let *S* be a server and *C* be a client. Let PB_s and PB_c be the policy bases associated with *S* and *C*, respectively. Let $X\text{-}Prof_S$ and $X\text{-}Prof_C$ be the *X*-Profiles associated with *S* and *C* respectively. Let *R*, be the resource requested by *C* to *S*. A negotiation tree $NT=<N, R, E>$ for *R*, *S*, and *C* is a finite tree satisfying the following properties:

- *N* (the set of nodes) is a set of triples:

  $$n=<T,\ state, party>$$

  where:

  - *T* is a term;
  - *state* denotes the current state of the node;
  - *party* E ∈ {*C*, *S*} denotes whether the node belongs to *C* or *S*;

- $R=<R,\ state,\ S>$ is the root of the tree;
- *E* (the set of edges), where each $e \in E$ has one of the following forms:4
  - simple edge *SE*: $e=(n_1, n_2),\ n_1, n_2 \in N$ belongs to *SE* if both the following conditions hold:
    - $[\ Eval(T(n_1), X\text{-}Prof_C)= TRUE \wedge T(n_1) \leftarrow T(n_2) \in PB_c$ or $Eval(T(n_1), X\text{-}Prof_S)= TRUE \wedge T(n_1) \leftarrow T(n_2) \in PB_S]$ or $[\ T(n_1)= R \wedge R \leftarrow T(n_2) \in PB_S\ ]$;
    - $[(Eval(T(n_1), X\text{-}Prof_C)= TRUE) \wedge (Eval(T(n_2), X\text{-}Prof_S)= TRUE)] \vee (Eval(T(n_1), X\text{-}Prof_S)= TRUE) \wedge (Eval(T(n_2), X\text{-}Prof_C)= TRUE)]$;
  - multi edge *ME*: $e=\{(n, n_1),\ldots, (n, n_k)\}\ n, n_1,.., n_k \in N$ belongs to *ME* if both the following conditions hold:

- [$Eval(T(n), X\text{-}Prof_C = TRUE \wedge T(n) \leftarrow T(n_1),..., T(n_k) \in PB_c$ or $Eval(T(n), X\text{-}Prof_S = TRUE \wedge T(n) \leftarrow T(n_1),..., T(n_k) \in PB_S]$ or [$T(n) = R \wedge R \leftarrow T(n_1), ..., T(n_k) \in PB_S$];
- $[(Eval(T(n), X\text{-}Prof_C = TRUE) \wedge (Eval(T(n_1), X\text{-}Prof_S = TRUE) \wedge ... \wedge (Eval(T(n_K), X\text{-}Prof_S = TRUE)] \vee [(Eval(T(n), X\text{-}Prof_C = TRUE) \wedge (Eval(T(n_1), X\text{-}Prof_C = TRUE) \wedge ... \wedge (Eval(T(n_K), X\text{-}Prof_C = TRUE)]$

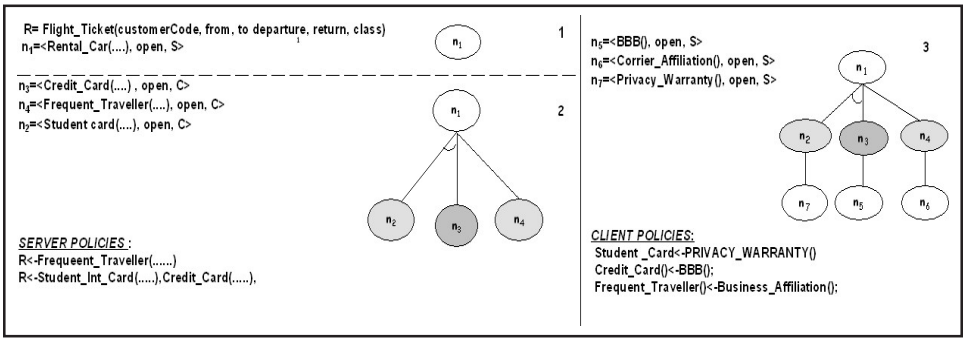
- The *state* of a node *n* can assume one of the following two values:
  - **DELIV**, if one of the following conditions holds:
    - $[T(n) \leftarrow \text{DELIV} \in PB_C \wedge Eval(T(n), X\text{-}Prof_C = TRUE] \vee T(n) \leftarrow \text{DELIV} \in PB_S \wedge Eval(T(n), X\text{-}Prof_S = TRUE]$
    - if $\exists\, e = (n, n_2) \in SE$ such that [$state(n_2)$=DELIV] or [$\exists\, e = \{(n, n_1),..., (n, n_k)\} \in ME$ and $\forall$ i $\in$ [1,k], $state(n_i)$=DELIV];
  - **OPEN**, if one of the following conditions holds:
    - if $e = (n, n_2) \in SE$ and $state(n_2)$=OPEN;
    - if $\{e = (n, n_1),..., (n, n_k)\} \in ME$ and $\exists$ i $\in$ [1,k] such that $state(n_i)$=OPEN;
    - $\forall\, e = (n, n_1) \in SE$ if $party(n) = C$ then $party(n_1) = S$ and $\forall\, e = \{(n, n_1),..., (n, n_k)\} \in ME$ if $party(n) = C$ then $party(n_1) = S,..., party(n_K) = S$, and vice versa.

A negotiation tree is thus a particular tree that evolves during the policy evaluation phase through addition of disclosure policies by one of the parties. Graphically, nodes are represented as labelled circles. Figure 5 shows three steps of the construction of a negotiation tree. The example shows a negotiation tree for our Flight Ticket scenario.

A negotiation tree may contain two different kinds of edges: multi and simple, which are the result of the different kinds of policy that can be expressed in our language. Multi edges are the result of policies having right side elements with more than one term; therefore, these edges model terms in conjunction. By contrast, simple edges are used to model policies having only one term on the left side component of the associated rule.

A simple edge is modelled as a directed line, whereas a multi edge is represented as many edges as the terms in the corresponding policy, linked by an arch. For example with respect to Figure 5, at step 2, the evaluation of policy: $R \leftarrow Credit\_Card(...), Student\_Int\_Card(...)$[5] results in a multi edge connecting node n_1 with nodes n_2 and n_3. The *state* associated with a node denotes the possibility of finding a trust sequence containing the corresponding term. Intuitively, if the state of a node *n* is DELIV, this means that there exists a trust sequence containing *T(n)*. In the example above, the request for the server certification from the Better Business Bureau, named *BBB*, does not need further requirements, and thus the corresponding node is tagged DELIV. When

Figure 5. Negotiation tree building



a new node is appended to the tree during the policy evaluation phase, its state is initially set to OPEN, meaning that the tree may evolve, through addition of children to that node. Then, it changes its state to DELIV when at least one of its children has a DELIV state, or there exists a delivery policy associated with its term (meaning that there are no additional protection requirements specified for such term). In case of multi edges it is required that the state of all the linked children be DELIV. With reference to Figure 5, the root node n_1 is labelled OPEN until the evaluation of disclosure policies *Privacy_Warranty←DELIV* (whose corresponding term is in node n_7) and *BBB←DELIV* (whose corresponding term is in node n_5) is completed, as shown in Figure 6.

OPEN nodes may have different evolutions. For instance, suppose that a Web server requires a certificate to a subject proving that the country that issued the subject birth certificate is a legal country. Likely, that kind of certificate does not belong to the birth certificate owner but to the issuer country, and, consequently, it is not immediately available. However, the subject may gather it using a credential chain. The disclosure is not certain, so the corresponding node is set to open. Alternatively, a node may be open also to maximize protection of policies and resources: a party may choose to reply only for one node at a time, without submitting policies for the other resources involved. Consequently, the tree evolves naturally giving priority to those solutions that can be locally solved.

Multi Path and Trust Sequences

The negotiation tree signals a potential trust sequence when it contains a *multi path* composed of all delivery nodes. We use the term *multi path* to outline the fact that the path may include multi edges, and consequently more brothers may be part of a path.1 Valid multi paths are formally defined as follows.

Definition 4.2.

Valid Multi path: Let $NT=< N, R, E>$ be a negotiation tree of height h. A *valid multi path* *mp* on *NT* is a partially ordered list of set of nodes $[S_1,...,S_k]$, where $S_i \in 2^N$, $i \in [1,k]$, $k \geq h$ such that:

- $S_1 = R$;
- The state of each node in S_i, $i \in [1,k]$ is *DELIV*;
- The nodes in each set S_i, $i \in [1,k]$, all refer either to *C* or *S*; that is, $\forall n \in S_i$, $i \in [1,k]$, *party(n) = C* (or *party(n) = S*);
- All the non leaf nodes in the sets belonging to the list are linked either by a simple edge or by a multi edge to one of the nodes in the sets following them in the list. Formally, $\forall S_i, S_{i+1} \in mp$:
  - if $|S_i|=|S_{i+1}|=1$, and the unique node in S_i is a non leaf node, then there must be in *E* a simple edge connecting the unique node in S_i to the unique node in S_{i+1};
  - if $|S_i|=1$, $|S_{i+1}| > 1$, and the unique node in S_i is a non leaf node, then there must exist in *E* a multi edge connecting the unique node in S_i to all the nodes in S_{i+1};
  - if $|S_i|=m$, $m > 1$, then $\forall n_j \in S_i$, $j \in [1,m]$ such that n_j is a non leaf node:
    - if $|S_{i+1}|=1$, then there must exist in *E* a single edge connecting n_j to the unique node in S_{i+1};
    - if $|S_{i+1}|>1$ then there must exist in *E* a multi edge connecting n_j to all nodes in S_{i+1}.

Figure 6 shows two multi paths valid for the considered negotiation.

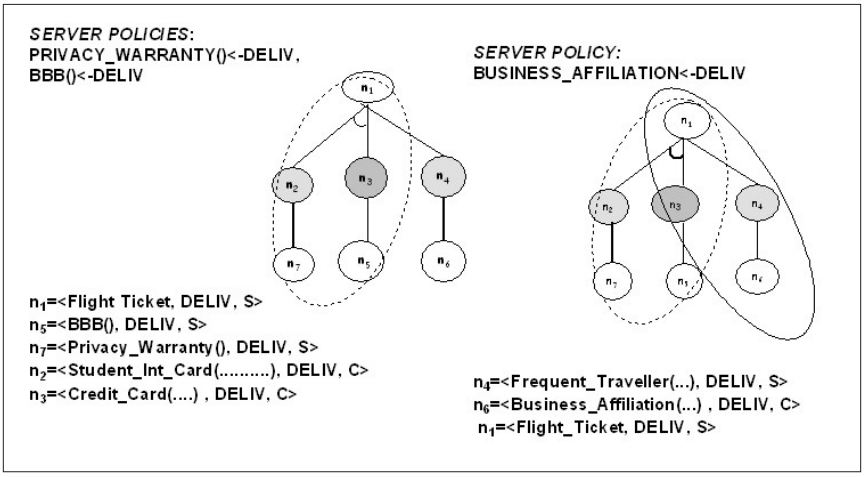
A trust sequence is a list of sets of certificates whose ordered disclosure leads to the grant of the requested resource, assuring at the same time the satisfaction of the disclosure policies of both the involved parties. This sequence can be obtained by simply grouping the sets of nodes composing a valid multi path, starting from the last set of the path, in such a way that the resulting sequence will be composed of as many sets as the height of the tree, each one containing certificates belonging alternatively to client and server party. Formally, a trust sequence is defined as follows.

Definition 4.3.

Trust Sequence: Let *S* be a server and *C* be a client. Let PB_s and PB_c be the policy bases associated with *S* and *C*, respectively. Let $X\text{-}Prof_s$ and $X\text{-}Prof_c$ be the *X*-Profiles associated with *S* and *C*, respectively. Let *R*, be the resource requested by *C* to *S*. A *trust sequence* *ts* for *R*, *S* and *C* is a ordered list of sets of certificates $[C_1,...,C_n]$ such that:

- $C_n = R$;
- For each set C_i, $i \in [1,n-1]$:

Figure 6. Examples of valid multi paths



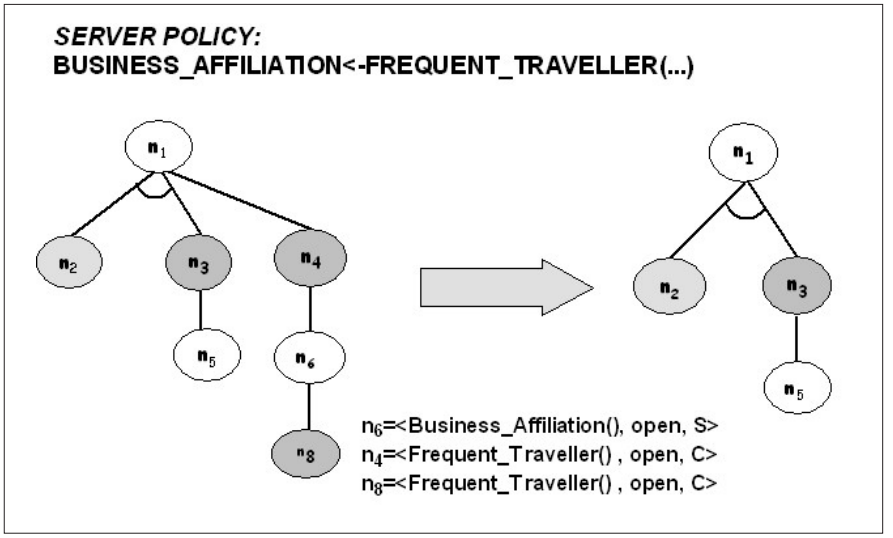
- All the certificates in C_i belong either to *C* or *S*;
- If the certificates in C_i belong to *C*, then the certificates in C_{i+1} belong to *S* and satisfy a corresponding policy of the form $c_{pi} \leftarrow T_1..T_n$, of *C* for each $c_{pi} \in C_i$ p=1,...,$|C_i|$. Vice versa, if the certificates in C_i belong to *S*, then the certificates in C_{i+1} belong to *C* and satisfy a corresponding policy of the form $c_{pi} \leftarrow T_1..T_n$, of S for each $c_{pi} \in C_i$ p=1,...,$|C_i|$.

Example 3.4

Consider the example in Figure 6. The paths shown in the figure are valid multi paths. In particular, the path on the left side, that is, [{n_1}{n_2, n_3},{n_7},{n_5}], is a valid multi path in that: (1) n_1={R}; (2) The state of each node is DELIV; (3) All the non leaf nodes in the sets belonging to the list are linked either by a simple or by a multi edge to some of the nodes in the sets following them in the list. For instance, {n_1} and {n_2,n_3} are linked by a multi edge, and each node in {n_2,n_3} is linked by a simple edge to a node in a set following it in the list: n_2 is linked to n_7, and n_3 to n_5, respectively. By Definition 3.8, the corresponding trust sequence is: [{BBB, Privacy_Warranty}, {Student_Intl_ Card,Credit_Card}, {Flight_Ticket}]. Note that the number of sets of certificates in the trust sequence is equal to the height of the negotiation tree and each set contains certificates, satisfied by a disclosure policy, belonging to C or S, alternatively.

The following theorem states the relationship between a trust sequence and a valid multi path. The formal proof is reported in Appendix.

Figure 7. Example of redundant path and the negotiation tree after the pruning



Theorem 1

Let C be a client requesting a resource *R* to a server S.

Let NT be the corresponding negotiation tree. For each trust sequence TS=[C_1,..,C_n={R}] associated with the negotiation, there is a valid multi path in NT consisting of all and only the terms satisfied by the certificates in TS. Additionally, for each valid multi path in NT, there is a corresponding trust sequence TS containing all and only the certificates corresponding to terms in the path.

Repeated Nodes Detection

Since parties are not always aware of counterpart policies, the evaluation of some policies can be recursive and create cycles. Repeated terms can be easily detected in the negotiation tree as soon as a term appears twice in the same path. In this case, the *tree manager* prunes the portion of the tree that creates the redundancy. The pruning is executed from the last repeated node (a terminal node) to the first instance of the term found going up towards the tree root. Obviously, each term is pruned only if it does not have any other edge in addition to the edge that creates the redundancy. Figure 7 shows an example of pruning. Suppose that in the negotiation depicted in Figure 5 the server adopts a policy stating that the submission of its Business_affiliation card can be executed only after receiving the Frequent_Traveller badge of the requester. In this case, nodes n_8, n_4 and n_6 are pruned because $T(n_4)= T(n_8)$=Frequent_Traveller, and $party(n_4) = party(n_8)=C$.

Note that the presence of many nodes in the negotiation tree referring to the same term $T(n,)$ and belonging to the same party, does not necessarily denote redundancy. Indeed,

if the repeated terms are not in the same path they do not denote a repetition and therefore the tree does not need to be pruned. In such a case the detection of the repetition can be exploited to speed up negotiation tree evolution. More precisely, consider two nodes n_1 and n_2, such that $T(n_1) = T(n_2)$, connected to the root by different paths. If the state of $T(n_1)$ is *DELIV*, it means that a valid multi path rooted at n_1 already exists. A pointer can thus be used to link n_2 to n_1, propagating the state of n_1 to n_2. As a result, n_2 can be immediately managed as a *DELIV* node, without the need of building again the sub tree rooted at n_1. By contrast, if the state of n_1 is *OPEN* and the height of the node is less than the current tree height the link is added anyway, in order to avoid redundant policies exchanges, but the state of both nodes is not modified.

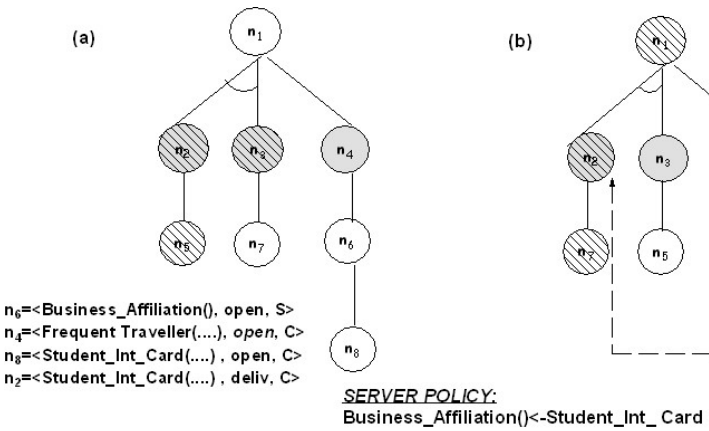
Example 3.5

With reference to the Example in Figure 5, suppose that the privacy warranty is unprotected and the corresponding policy is a delivery policy. As a result, the states of nodes n_2 and n_7 are updated and become DELIV (see Figure 8(a)). Moreover, suppose that the server requires the Student_Int_Card as a document in order to disclose its business affiliation. A new node, labelled n_8, is then added to the tree and linked to node n_2, since it refers to the same term. As a result, the state of n_2 is immediately set as DELIV, thus introducing a new valid multi path, as shown in Figure 8(b).

Certificate Exchange Phase

As remarked in the previous sections, both parties have a complete view of the state of a negotiation and consequently they can both be able to determine valid multi paths. The existence of a valid multi path is thus communicated by the first party that, by processing its policies, detects the path. If more than one valid multi path is determined in the same step, the party analyzes each of them and establishes the associated trust sequences.

Figure 8. Example of link usage (shadowed nodes denote deliv nodes)



Then, the party suggests the one it prefers with a Sequence message. Otherwise it signals to the other party the unique valid sequence determined. If the counterpart accepts the suggestion, the two parties begin the exchange of the certificates; otherwise a different sequence is proposed to the counterpart. This interaction goes on until the two parties establish an agreement on a sequence. Once the parties come to an agreement, the certificate exchange phase starts. Each party discloses its certificates, observing the order defined in the sequence. Note that, when a party discloses a set of certificates, it actually discloses one certificate at the time. Upon receiving a certificate, the counterpart verifies the satisfaction of the associated policies, checks for revocation, checks validity dates and authenticates the ownership (for credentials). Eventually, if further information is needed for establishing trust, it is the receiver's responsibility to check for new certificates using credential chains. For example, if a medical certificate was requested and the issuer is an unknown hospital, the receiver party has to check the validity of issuer certificate by collecting new certificates from issuer repository. The receiver then replies with an acknowledgment expressed with an ack message, and asks for the following certificate in the sequence, or whether it has received all the certificates of the set, and it sends a certificate belonging to the following set of certificates in the trust sequence. If no unforeseen event happens, the exchange ends with the disclosure of the requested resource.

Example 3.6

Consider the valid multi paths shown in Figure 8. The corresponding trust sequences are:

[{Very_Sign_Certificate, Privacy_Warranty},{Student_Int_Card, Credit_Card,},
{Flight_Ticket}] and [{Business_Affiliation}, {Frequent_ Traveller}, {Flight_Ticket}].

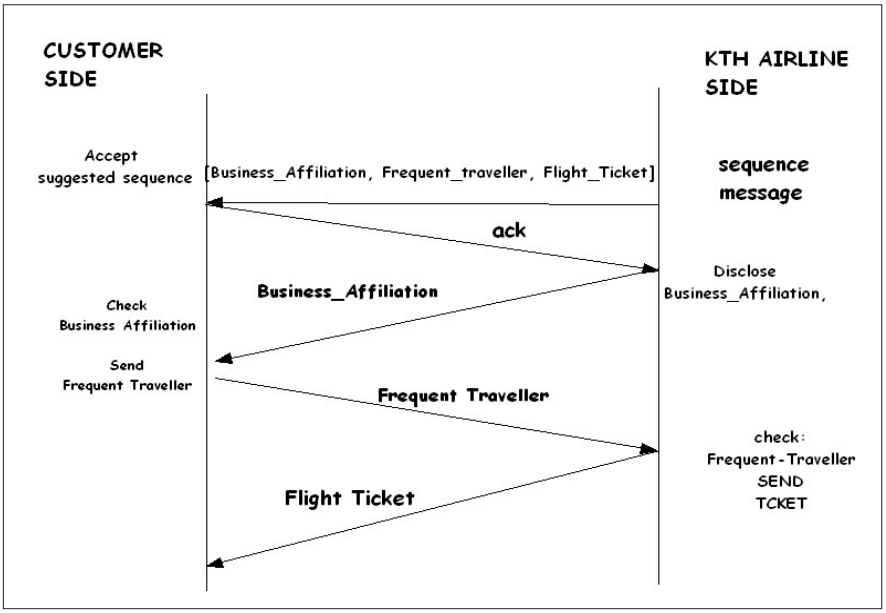
The two trust sequences are determined by the server party, which is the first who, with its delivery policies, determines two valid multi paths. Assume that the parties agree on the second trust sequence, since it is faster and easier to be executed. Figure 9 shows the messages exchanged by the two parties.

Related Work

Because of the relevance of trust negotiation for Web-based applications, a number of systems and research prototypes have been recently developed (Blaze & Fegeinbamum, 1999; Bonatti & Samarati, 2001; Herzberg & Mihaeli, 2001; Winsborough & Li, 2002), whcih we survey and analyze in what follows and compare with our proposal.

PSPL, proposed by Bonatti and Samarati in 2001, is part of a uniform framework for formulating and reasoning about information release on the Web. It is a protection language for expressing access control policies for services and release policies for client

Figure 9. An example of certificate exchange phase



and service portfolios. The language also includes a policy filtering mechanism, to provide policy disclosures and to protect privacy during policy disclosures. The main difference between PSPL and our language is that PSPL only provides a logical definition of the language constructs. Therefore no directly usable language is provided.

The Trust Policy Language (TPL) (Herzberg & Mihaeli, 2001) is an XML-based framework for specifying and managing role-based access control policies in distributed context where the involved parties are characterized by credentials, and digital certificates are used for authentication. There are two versions of TPL: Definite TPL (DTPL), and TPL itself. DTPL is a subset of TPL that excludes negative rules, and it is therefore monotonic.

TPL credentials, like Trust-*X* credentials, contain a reference to the site associated with the issuer. However, no protection for sensitive credentials is provided by TPL, since credentials are assumed to be accessible to anyone.

Reference site of issuers contained in credentials is used as a starting point for a collector-controlled search for relevant supporting credentials. One of the most important features of TPL is, indeed, the support of credential chain discovery, which is not yet fully supported by our system.

KeyNote (Blaze & Feigenbaum, 1999) is the most well known trust management language. It was designed to work for a variety of large and small scale Internet-based applications. It provides a single, unified language for both local policies and credentials. KeyNote policies and credentials, called "assertions," contain predicates that describe the trusted actions permitted to the holders of specific public keys. KeyNote, due to its intended use for delegation authority and the fact that trust negotiation uses attributes of the negotiation parties as the basis for trust, is poorly suited for trust negotiation.

Trust Builder (Seamons & Winslett, 2003) is one of the most significant proposals. It provides a set of negotiation protocols that define the ordering of messages and the type of information messages will contain, and of strategies for controlling the exact content of messages. A variety of strategies are defined to allow strangers to establish trust through the exchange of digital credentials and the use of access control policies that specify what combinations of credentials a stranger must disclose in order to gain access to each local service or credential. Trust Builder is the approach that more greatly influenced our work. For instance, we borrow from Seamons and Winslett's work (2003) the use of a tree structure to maintain the progress of a negotiation and keep track of possible alternative strategies.

Finally, the principle of separation of policy exchange from credential disclosure is also achieved by the parsimonious strategy proposed by Seamons and Winslett (2003) and by the PRUNES strategy (Yu, Ma & Winslett 2001). Both approaches are based on negotiation search tree, focusing on automatic strategies for policy exchange in order to avoid as much as possible disclosure of credentials. Yu, Ma and Seamons' (2001) work also ensures completeness of the negotiation strategy, where a negotiation strategy is said to be complete when it leads the negotiation to succeed whenever possible. According with the above definition, our approach can be considered complete as well as the one proposed in the referred work. Moreover, the authors refer to a security agent in charge of automatically carrying on negotiations, without user intervention. By contrast, we present a much more flexible approach where a user can eventually drive the negotiation process to either maximize protection, or maximize the number of potential solutions for negotiation.

Mobile Commerce with Trust-*X*

In this section, we discuss how the Trust-*X* negotiation system can be properly applied to mobile commerce, and show how our method can influence mobile commerce security. The section is organized as follows. We first introduce the main issues related to the development of a system supporting negotiations for mobile users. Then, we illustrate a variety of techniques to extend Trust-*X* in order to fully support mobile negotiations.

Open Issues

The area of trust negotiation for mobile systems is today a promising and challenging research area, since it is expected that the number of wireless clients accessing Internet will rapidly increase in the next few years. This will lead to an environment where the number of wireless clients accessing the Internet to perform mobile transactions will greatly exceed the number of clients accessing the Internet through networked computers. The development of a system supporting negotiations for mobile users presents significant challenges, mainly arising from the need of migrating trust negotiation concepts and their complex requirements into a mobile context. More precisely, we have

identified the following major aspects that need to be taken into account when dealing with mobile negotiations:

- Mobile devices, like cell phones, PDAs, laptops and portable MP3 players, have limited storage capacity, processing power, and network bandwidth compared to typical desktop computers. Such feature may be critical in mobile trust negotiations, since a conventional trust negotiation system requires a significant processing power for keeping track of the negotiation process and a certain storage capacity for collecting credentials and policies.
- Trust negotiation systems usually rely on credentials, signed using public keys managed by public key infrastructures. However, public key infrastructures are usually onerous for mobile devices.
- Mobile network topologies are often unpredictable and dynamically change. A mobile system must thus handle interactions that may take place in a variety of different configurations.
- A single user can possess multiple mobile devices. Since a fundamental requirement of trust negotiations is to ensure ownership of the credentials exchanged among the parties, a key issue is to allow independent devices to safely share and access user credentials.
- Networks can be easily compromised; thus suitable recovery mechanisms for mobile trust negotiations need to be developed.

In what follows, we focus on some of the issues listed above. In particular, we reason about security problems related to mobile negotiations and sketch a variety of possible solutions. We do not further discuss on issues related to network availability and quality of service since they are outside the scope of this chapter. For what concerns the mobile payments techniques, we would prefer to build on current technologies and take advantage of existing infrastructure and integrate them in our Trust-*X* framework rather than redesign them. For example, we may produce strategies and techniques for supporting mobile payments with multi-application devices but for technical specifications that address specific areas of mobile payments, such as cardholder authentication, passwords, and encryption we refer to specific organizations created for such purposes. In this way we would complement, rather than duplicate efforts. The aim of the proposed approaches is, instead, to adapt Trust-*X* in order to make it possible to perform trust negotiations and mobile transactions and payments in vulnerable environments by users having limited computational resources.

Extending Trust-*X* to Support Mobile Commerce Applications

Extending trust negotiation systems for the management of mobile users requires first to provide resource-compatibility with devices that typically have constrained re-

sources. A mechanism for storing credentials needs thus to be devised that keeps into account the limited capacity of mobile devices. End users require device ownership: although anyone can pick up a device, only its real owner must be allowed to carry out transactions involving personal credentials. A possible approach is to centralize credential storage into a single repository, allowing each user to maintain the associated profile on a secure server. This schema for credential management is sketched in Figure 10(a). Under this approach, a user can change the device from which he/she performs transactions without the need of moving the entire profile with himself/herself, and refer to it while negotiating services. Another important issue is whether it is practical for wireless clients to efficiently perform all the phases required to complete a trust negotiation, such as the storage of all required credentials, the processing steps that include costly cryptographic verifications, and the network communications with the other negotiation participants. One potential scalable solution is to offload trust negotiation from the thin client and conduct it out-of-band between the server and an agent managing the client's credentials via a higher-speed network connection. Alternatively, it is possible to adopt ad hoc strategies for negotiations in order to reduce the information to be exchanged as much as possible. Finally, another potential approach is to let the party having a consistent connection (called *negotiation driver,* for simplicity) with sufficient bandwidth to drive the process. Obviously, this approach can succeed only if the negotiation driver is a company supplying services able to prove its trustworthiness. Under this scheme, once the driver trustworthiness is ensured, the negotiation driver can keep track of the progress of the negotiation, freeing the mobile party from carrying this burden.

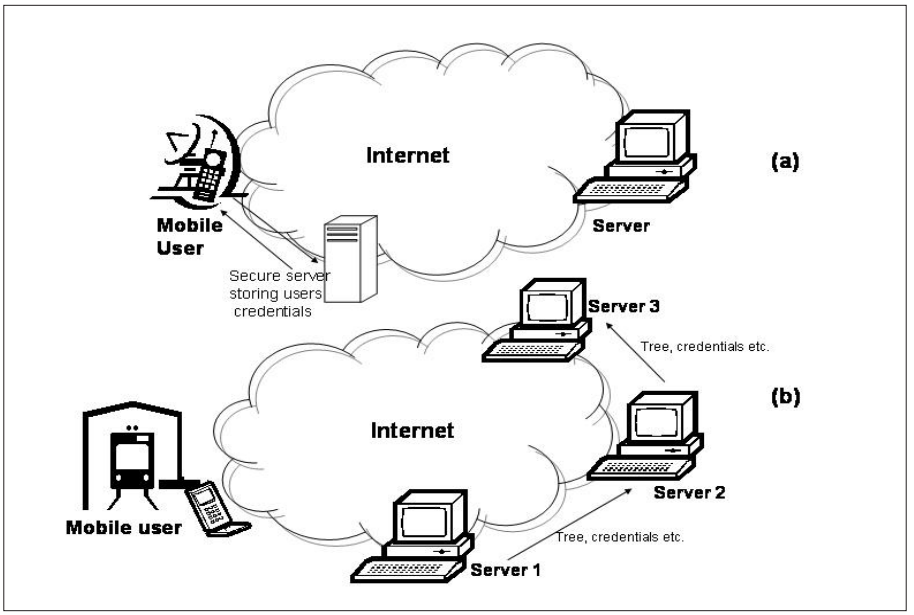
A possible further extension addressing the issues listed in the previous section is sketched in Figure 10(b). Such approach consists of providing the possibility of dynamically changing the server with which to perform the negotiation, while the mobile party is moving. For instance, suppose Alice is connected via a mobile phone to a Web server, purchasing clothes at an online store. Suppose she is travelling by train while performing the transaction. Then, if the transaction cannot be completed with the same server because of Alice's change of geographic position, the server can interrupt the negotiation and suggest Alice to connect to another closer server having the same capabilities, able to carry out the negotiation in an equivalent way. Or better, the server could automatically redirect the negotiation to another trusted server, after requesting the consensus of Alice for this operation. Clearly, this implies that the server with which Alice was performing the negotiation should transmit to the new one all the information related to the ongoing negotiation process in a secure way. For instance, if we assume that both the servers are Trust-*X* compliant this implies transmitting the negotiation tree built until the interruption, the determined trust sequences, if any, and the credentials Alice has sent to the server until that point. Intuitively, this approach is based on two strong assumptions: the trustworthiness of parties and the existence of a network of trusted servers that can be interchanged during negotiations. Finally, another interesting research issue is the possibility of exploiting already executed negotiation processes with a certain entity to simplify the next mobile transactions to be executed. In a scenario characterized by a mobile user and a negotiation driver, the driver can collect the information obtained by the user in previous successful negotiations to speed up next

processes. Furthermore, the driver can exploit the collected data to advertise the user the services it sells, selecting the products to advertise on the basis of user profile, and on the basis of the geographical location of the user. However, this last functionality requires addressing privacy issues that arise when private information is collected by a remote party for purposes other than the transaction which it was released for. This approach indeed relies on the Web server's possibility to store remote credentials and collect user personal information. Next example shows how a mobile commerce transaction may be employed using the approach previously introduced.

Example

Alice is a frequent flyer who usually makes use of her PDA to purchase flights tickets by KTH. Suppose she is carrying on a negotiation to buy a flight ticket for her next travel to a foreign country. Suppose, moreover, that she is travelling by train. Instead of executing the whole negotiation process in such a vulnerable environment in order to complete the transaction, Alice can just delegate the negotiation driver, that is, KTH, the task of memorizing the negotiation tree and skip the negotiation of requisites already proved in previous negotiations. Furthermore, there is no need of communicating the credit card number, which represents the most sensitive information to be exchanged, if KTH already knows it and has Alice's consent of maintaining it for use. KTH, by using Trust-*X*, can exploit Alice's collected data to advertise to her about the services it sells. For instance it may send Alice information about the possibility of booking hotels, renting cars and so on, on the basis of Alice's profile, and on the basis of her actual geographical location, as well as Alice's next destination.

Figure 10. Examples of possible topologies for mobile trust negotiations



Conclusions and Future Research Directions

Automated trust negotiation between strangers promises to extend trusted interactions to a broader range of participants than it is possible with traditional security approaches based on identity and capabilities. One of the most interesting applications for trust negotiation systems is given by the possibility of performing e-commerce applications. Mobile commerce, in particular, is an important branch of e-commerce requiring additional trust establishment capabilities.

In this chapter, besides introducing the basic principles of trust negotiation, we have presented Trust-*X*, a comprehensive XML-based framework for trust negotiations. We have mainly focused on disclosure policies and the various phases in which a Trust-*X* negotiation is articulated. Then, we have presented a number of possible extensions we are currently exploring for migrating the system into mobile scenarios. The work reported in this chapter is part of an on-going project aiming at fully supporting mobile commerce applications.

Additional future work includes the extension of *X*-TNL along several directions, such as for instance the possibility of specifying the credential submitter. Another extension we are currently investigating is the possibility of disclosing only portions of a credential during the negotiation process. This allows us to protect the elements of a credential in a selective and differentiated way. Finally, an implementation of Trust-*X* is in progress.

References

Bertino, E., Castano, S., & Ferrari, E. (2001). On specifying security policies for Web documents with an XML-based language. *Proc. of SACMAT' 2001, ACM Symposium on Access Control Models and Technologies,* Fairfax, VA.

Bertino, E., Ferrari, E., & Squicciarini, A. (2003). X-TNL - An XML based language for trust negotiations. *Fourth IEEE International Workshop on Policies for Distributed Systems and Networks*, Como, Italy.

Bertino, E., Ferrari, E., & Squicciarini, A. (n.d.). Trust-*X* – A peer to peer framework for trust establishment. To appear in *IEEE Transactions of Knowledge and Engineering*.

Blaze, M., Feigenbaum, J., Ioannidis J., & Keromytis, A. (1999). The KeyNote trust-management system. *RFC 2704*.

Bonatti, P., & Samarati, P. (2000). Regulating access services and information release on the Web. *7th ACM Conference on Computer and Communications Security*, Athens, Greece.

Brands. (2000). *Rethinking public key infrastructure and digital credentials.* MIT Press.

Damianou, N., Dulay, N., Lupu, E., & Sloman, M. (2001). The ponder policy specification language. *International Workshop on Policies for Distributed Systems and Networks,* LNCS 1995, Bristol UK.

Dierks, T., & Allen, C. (1999). The TLS protocol version 1.0. *RFC 2246*.

Herzberg, A., Mihaeli, J., Mass, Y., Naor, D., & Ravid, Y. (2000). Access control meets public infrastructure, or: Assigning roles to strangers. *IEEE Symposium on Security and Privacy,* Oakland, CA.

Persiano, P., & Visconti, I. (2000). User privacy issues regarding certificates and the TLS protocol. *Proceedings of the ACM Conference on Computer and Communication Security,* Athens, Greece.

Seamons, K.E., Winslett, M., & Yu, T. (2001). Limiting the disclosure of access control policies during automated trust negotiation. *Network and Distributed System Security Symposium,* San Diego, CA.

Seamons, K.E., Winslett, M., & Yu, T. (2002). Requirements for policy languages for trust negotiation. *Third IEEE International Workshop on Policies for Distributed Systems and Networks,* Monterey, CA.

Stallings, W. (1999). *Cryptography and network security: Principles and practice* (2nd ed.). Prentice Hall.

Subrahmanian, V. et al. (n.d.). Hermes: Heterogeneous reasoning and mediator system. *http://www.cs.umd.edu/projects/publications*

Winsborough, W.H., & Li, N. (2002a). Protecting sensitive attributes in automated trust negotiation. *ACM Workshop on Privacy in the Electronic Society*.

Winsborough, W.H., & Li, N. (2002b). Towards practical automated trust negotiation. *Third International Workshop on Policies for Distributed Systems and Networks,* Monterey, CA.

World Wide Web Consortium. (1998). Extensible markup language (XML) 1.0. *http://www.w3c.org/TR/*

Yu, T., Ma, X., & Winslett, M. (2000). PRUNES: An efficient and complete strategy for trust negotiation over the Internet. *Proceedings of the 7th ACM Conference on Computer and Communications Security,* Athens, Greece.

Yu, T., Winslett, M., & Seamons, K.E. (2003). Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security, 6*(1).

Endnotes

1 Like for credentials, we assume that data set names are unique, and that are registered through some central organization.

2 Prerequisite information are encoded using the same formalism we have developed for disclosure policies.

[3] With the term trust we refer to the fact that the sequence is composed by certificates whose corresponding disclosure policies are satisfied. Safety is not related, in this context, with certificates validity or their effective content.

[4] Given a node $n \in N$, we use the notation T(n) to denote the term in n; *state(n)* to denote the state of n; and *party(n)* to denote the owner of the term in n.

[5] For simplicity, we focus on the most relevant policies and related terms.

Appendix

Proof of Theorem 1

We start by proving the first part of the thesis. The proof is by induction on the length l of the trust sequence.

Basics

$l = 2$, then TS = [C_1,{R}]. Let us first suppose that $|C_1| = 1$ and let c be the unique certificate in C_1. Since, by hypothesis, *TS* is a trust sequence, then the Policy Base of S must contain disclosure policy of the form: R ← T, such that: (1) *P(T)* is the type of c; and (2) the X-Profile of C contains a certificate of type P satisfying the conditions in c(T). By Definition 4.1, *NT* contains two nodes n_1 and n_2 such that: (1) $T(n_1)$ = R; (2) $T(n_2) = T$; (3) $state(n_1) = state(n_2)$ = DELIV; (4) $party(n_1) = S$ and $party(n_2) = C$. Additionally, E contains the simple arc (n_1,n_2). Thus, by Definition 4.2, *NT* contains a valid multi path [S_1, S_2], such that $S_1 = \{n_1\}$, and $S_2 = \{n_2\}$, which proves the thesis. If $|S_1| > z > 1$, let c_{11},… c_{1z} be its elements. Since by hypothesis *TS* is a trust sequence, the Policy Base of S must contain a disclosure policy of the form: $R \leftarrow T_{11}$,... T_{1z}, such that $P(T_{1i})$ is the type of c_{1i}, i ∈ [1,z] and the X-Profile of C contains a certificate of type c_{1i} satisfying the conditions in $c(T_{1i})$, i ∈ [1,z]. By Definition 4.2, *NT* contains a node n_1 such that $T(n_1) = R$, and a node n_{1i} such that $T(n_{1i}) = T_{1i}$, i ∈ [1,z]. Additionally, $state(n_1) = state(n_{1i})$ = DELIV, i =1,…, z, whereas $party(n_1) = S$, $party(n_{1i}) = C$, i = 1, ….,z. Finally, E contains a multi edge {(n_1, n_{11}), …., (n_1, n_{1n})}. Thus, by Definition 4.2, *NT* contains a valid multi path [S_1, S_2], such that $S_1 = \{n_1\}$, and $S_2 = \{n_{11}, …., n_{1z}\}$, which proves the thesis.

Inductive Step

Let us consider a trust sequence *TS*, such that $l = h > 2$. Suppose that the thesis holds for trust sequences of height $h' < h$ and let us prove the thesis for h. Then, TS = [$C_{1...,}$ $C_{h-I,}$,{R}]. By inductive hypothesis, if we consider the trust sequence *TS'* obtained from *TS* by dropping set C_1, then there exists a valid multi path mp = [S_1, …, S_p], in the corresponding negotiation tree. Let us thus consider set C_1. If $|C_1| = |C_2| = 1$, or $|C_2|$ =

1 and $|C_1| = k > 1$, then using the same reasoning we have applied for $l = 2$ we can prove the thesis simply by concatenating to the sets of nodes in *mp* the nodes corresponding to the certificates in C_1, using the same strategy adopted above.

Thus, we are left to consider the case when $|C_1| = k > 1$, and $|C_2| = j > 1$. We can suppose, without loss of generality, that $j = 2$. Suppose moreover that the party of each node in S_p is C. Let c_{11}, …., c_{1k} be the elements in C_1 and c_{21}, c_{22} be the elements in C_2. Since, by hypothesis, *TS* is a trust sequence, the Policy Base of S must contain two disclosure policies of the form: $c_{2i} \leftarrow T_{i1} \ldots T_{in}$, $n \geq 1$, i = 1, 2 such that: (1) $P(T_{im}) \in S_p$, $m = 1, \ldots, n$, i = 1, 2; and (2) the *X*-Profile of *C* contains a certificate of type $P(T_{im})$ satisfying the conditions in $C(T_{im})$, $m = 1,\ldots, n$, $i = 1, 2$. Let us first consider the policy $c_{21} \leftarrow T_{11} \ldots T_{1n}$. Suppose first that $n = 1$; that is, the right side of the policy contains only the term T_{11}. By Definition 4.1, *NT* contains a node n_1 such that $T(n_1) = c_{21}$, and a node n_2 such that $T(n_2)$ is equal to the certificate type in T_{11}. Additionally, $state(n_1) = state(n_2)$ = DELIV, whereas $party(n_1)$ = S, $party(n_2) = C$. Finally, *E* contains a simple edge (n_1, n_2). If $n > 1$, we can apply a similar reasoning, the only difference being that the negotiation tree will contains a multi edge instead of a simple edge, and the same reasoning can be applied to the policy $c_{22} \leftarrow T_{21} \ldots T_{2n}$. Thus, by Definition 4.2, *NT* contains a valid multi path *mp'* obtained from *mp* by concatenating two sets of nodes: one corresponding to the nodes added for policy $c_{21} \leftarrow T_{11} \ldots T_{1n}$, and the other corresponding to the nodes for policy $c_{22} \leftarrow T_{21} \ldots T_{2n,}$ which proves the thesis.

Let us prove the second part of the thesis. Let $mp = [S_1, \ldots, S_k]$ be a valid multi path in *NT*. Let $S_i \in mp$ be a generic set of $n_1 \ldots n_k$ nodes. Suppose, without loss of generality, that the party of each node in S_i is *C*. By Definition 4.2 the state of all the nodes in S_i is DELIV; then, there must exist in the Policy Base of *C* a disclosure policy: $T(n_j) \leftarrow T(n_{1j}) \ldots T(n_{pj})$ for each node $n_j \in S_i$, and $party(n_j) = C$ for $j \in [1,k]$. Moreover, $n_{1j} \ldots n_{pj} \in S_j$ with $j > i$ and $party(n_{sj}) = S$, $s \in [1,p]$. By Definition 4.1, $Eval(T(n_j), Prof_C)$ = TRUE and $Eval(T(n_{si}), Prof_S)$ = TRUE for $s \in [1,p]$. Thus, for each node $n \in S_i$, there exists a certificate of type c_i such that $P(T(n))=c_i$ which satisfies the condition in $C(T(n))$. Each set $S_i \in mp$ has, therefore, a corresponding set of certificates $C_1 = \{c_1,..,c_k\}$. The corresponding trust sequence $TS = [C_1,...,C_n]$ is therefore obtained grouping all the certificates belonging to consecutive sets associated with the same party in *mp* into a unique set C_j until a certificate belonging to the other party is found. The resulting sequence satisfies Definition 4.3, which proves the thesis.