A service-based testbed for Trust Negotiation

A SERVICE-BASED TESTBED FOR TRUST NEGOTIATION

Filippo Agazzi, Michele Tomaiuolo
Dipartimento di Ingegneria dell'Informazione
Università di Parma, Italy

ABSTRACT

Trust Negotiation allows users to develop trust incrementally, by disclosing credentials step by step. This way, services and resources can be shared in an open environment, and access rights can be granted on the basis of peer-to-peer trust relationships. This article presents a service-based testbed for Trust Negotiation. At its core, it is created as a generic framework based on the WS-Trust standard. It integrates a modular trust engine and a rule engine, which is used as a policy checker. The system is mainly oriented at Web services composition and location-based social networking scenarios.

1 INTRODUCTION

The interoperability promised by Web services standards paves the way for the provision and use of services not just in intranets, but also in the global Internet. Online interactions may involve human users together with software agents, either autonomous or simply reactive. Such an open and dynamic environment is characterized by: (*i*) unreliable behaviour of users and agents, because of conflicting objectives and different ownership, (*ii*) incomplete knowledge of the global environment, and (*iii*) absence of central authority and globally trusted institutions, in general. Moreover, the potential user base of an application provided on the open Internet is still growing, with the mass adoption of social networking tools. Contacts among people often develop fully online, possibly with no body of previous knowledge to associate with an online identity. Given such a new way the Internet is being used today, the automatic or assisted management of trust relationships is a concrete necessity. Thus, the approach of Automated Trust Negotiation (ATN) is becoming relevant, because it allows unknown users and agents, desiring to share any resource or service, to establish a level of trust in an incremental way through the exchange of credentials.

In this scenario, the open selection and composition of services is made possible, since ATN simplifies the creation and management of trust bounds. In fact, delegation and workflow composition may only be applied on the basis of careful protection of resources and information. This problem can be solved in a fully distributed way if local trust relations are taken into account, for example using a Trust Management scheme, possibly in association with a Trust Negotiation protocol. However, to our knowledge, there is virtually no implementation of ATN for Web services which is practically and freely available. This situation motivated us to design and implement an open source generic framework for ATN, which is distributed together with some components allowing its use in a Web services scenario. Another application area for the framework is constituted by social networking, especially in the case of pervasive and distributed platforms, exploiting location-based services.

The rest of the article is organized as follows: Section II presents an overview and a literature review of ATN; Section III describes a generic Trust Negotiation framework for Web services, based on the WS-Trust standard; Section IV provides details about practical implementation and use of such a framework, as a testbed for generic Web services; Section V describes in particular the use of the framework in the context of location-based social networking; finally, some concluding remarks are provided.

2 BACKGROUND

Trust is an important aspect of human life, and it has been studied under different points of view, for example in the context of psychological and sociological sciences, or to draw specific economical models (Luhmann 1979, Barber 1959, Deutsch 1962). Gambetta (2000) defines trust as a "subjective probability with which an agent assesses that another agent or a group of agents will perform a particular action...". Though this definition is founded on the mathematical concept of probability, Castelfranchi and Falcone (2001) argue it still hides many important details. Instead, they present trust using a socio-cognitive approach, providing a deep analysis of a party's believes, and the way they can influence trust. In particular, they list the beliefs about competence, disposition, dependence and fulfillment as important components of trust in every delegation, even towards non-cognitive software service providers. Instead, delegation towards people, organizations and social entities requires the delegating entity to hold additional believes about willingness, persistence and self-confidence of the partner, at least for the specific domain of the delegation. Then, using the socio-cognitive approach, trust can be evaluated as a continuous function of its constituents (Castelfranchi and Falcone 2003), more precisely of the certainty of its constituent beliefs.

One source of trust is based on direct experience and knowledge, arising from past interactions with a certain agent. Another source of trust is based on societal experience, or *reputation*, which consists of observations by a society of an agent's past behaviour. Such observations can then be made available to other agents, who themselves have not interacted directly with the evaluated agent. Various mechanisms are being used to manage reputation, i.e. to aggregate these indirect observations of the various experiences of participants in the system and to provide aggregate results or raw data to interested agents. Sabater and Sierra (2002) classify such mechanisms as: (*i*) *witness reputation*, where information about the target agent is requested to witnesses and collected directly by the evaluator, (*ii*) *neighborhood reputation*, which uses the social environment of the target agent and its relations with neighbors, and (*iii*) *system reputation*, which requires the presence of institutional structures considered trustworthy by all involved agents. Huynh, Jennings and Shadbolt (2006) deal in particular with open environments and use slightly different forms of data aggregation: (*i*) *witness information*, to be acquired directly, (*ii*) role-based trust, granted according to the membership in a trustworthy group, and (*iii*) *certified reputation*, based on authenticated references collected by the target agent itself.

Though trust is a continuous function, evaluated on the basis of its various considered constituents, the decision to delegate is necessarily discontinuous in its nature. The delegating entity can just decide to delegate or don't delegate, and this decision has to take into account not only the degree of trust, but even other factors. These factors, including the importance of the goal, the perceived risk of frustrating the goal, the increased dependence on the trustee, and all other costs or possible damages associated with the delegation, will all influence a threshold function which will be eventually compared with the degree of trust for deciding whether to delegate or not.

Following this approach, security is deeply intertwined with both the degree of trust and the threshold function. In fact, security can certainly influence positively the trust on the partner, especially if security includes auditing mechanisms, certifications and confidentiality. On the other hand, trust can provide a foundation for delegation. Delegation often comes in the twofold

aspect of delegation of duties (performing actions or achieving goals), and delegation of corresponding permissions (rights to access the needed resources). In *Trust Management* schemes (Li, Mitchell and Winsborough 2005), delegation can be effectively modulated according to the degree of trust.

An actual *Trust Negotiation* process is intended as the flow of credentials and policies between two entities through a sequence of requests and releases. In this process, digital credentials and policies are exchanged step by step, to increase the level of trust between involved parties (Winsborough, Seamons and Jones 2000; Winslett et al. 2002). In such a process, credentials and policies are considered as sensitive resources, to be protected by appropriate access policies, along with other kinds of resources.

With respect to the management and computation of policies in Trust Negotiation, a particularly important element is the *policy compliance checker*. Starting from a policy and a set of credentials, the policy compliance checker must be able to find the credentials which satisfy the policy, if they are effectively available as a subset of all disclosed credentials. For this purpose, it is also necessary to translate each credential from its original format into an assertion of the policy language.

Considering the example of a client requesting a service, one of the problems to solve is how the client comes to know which credentials it is required to present, and how the policies protecting the service and the credentials are disclosed. A *negotiation strategy* defines when and which credentials and policies must be disclosed and inserted into a message to send to the other party; how much computational load to dedicate to the negotiation (e.g., the maximum number of rounds) and other decisions about the behaviour to pursue during the negotiation. Moreover, a successful negotiation is not always possible, since for example one of the two parties does not possess sufficient credentials: the strategy has to determine the moment to abandon the negotiation, since it is not possible to conclude it with success. There is a vast choice of possible negotiation strategies, each one with its peculiar features. An important distinction can be drawn upon the level of prudence in the disclosure of credentials and policies. Winsborough, Seamons and Jones (2000) consider an Eager Strategy, a Parsimonious Strategy, a Prudent Strategy. A *family of strategies* is defined as a set of reciprocally compatible and interoperable strategies. Yu, Winslett and Seamons (2001) present the DST (Disclosure Tree Strategy) family. An important advantage regards the fact that a negotiating agent can choose, among a set of strategies belonging to the same family, the closest one to its own requisites. Moreover, this way it can adopt different strategies, during the different stages of a negotiation.

3 A STANDARD-COMPLIANT ATN PROTOCOL FOR WEB SERVICES

Web services are an important application area for Trust Negotiation, especially if they are provided in an open global context. For this reason, we'll introduce here a generic Trust Negotiation protocol for Web services. The protocol is designed in conformance to relevant standards for Web services security. Thus, the description of the protocol will be preceded by a brief overview of those standards.

1.1 Relevant standards for Web services security

SOAP Web services can exploit the SOAP header as an extensible container for message metadata, which provides developers with a set of options also covering the most typical security issues. The so-called WS-* specifications are designed in order to be composed with each other.

WS-Security supports the definition of security tokens inside SOAP messages and uses XML Security specifications to sign or encrypt those tokens or other parts of a SOAP message. Other specifications, including WS-SecureConversation, WS-Trust, WS-Policy, WS-SecurityPolicy, provide additional SOAP-level security mechanisms. The WS-Trust standard defines mechanisms for mediating trust relations among entities in the context of Web Services. It considers a security model in which a Web service can request that a received message proves a set of claims (e.g. name, key, privileges, etc) or, more commonly, that it carries a security token representing a relation between the sender and some other entity, trusted by the service provider. In this context, a service provider can request a client, before accessing its services, to present a token released by a trusted entity. A new client would probably not possess a proper token to access the service, in advance. For this reason, WS-Trust defines a protocol for allowing a client to contact an authority, trusted by the service provider, to request the token. Such an authority is defined as a Security Token Service (STS). An STS, on his turn, can define the requirements which clients have to satisfy to obtain the release of a token. As an STS is responsible for releasing those tokens, it is also known as a "token issuer".

In Fig.1, arrows represent possible paths of communication among the Requestor (client), the Web service Provider, and the STS. The Requestor contacts the STS for receiving a token. The STS has the duty to verify that the Requestor possesses the necessary attributes for obtaining a token. If the policy of the STS is satisfied, the STS releases a token. At this point, the Requestor can send a message to the Web service Provider, attaching the obtained token.

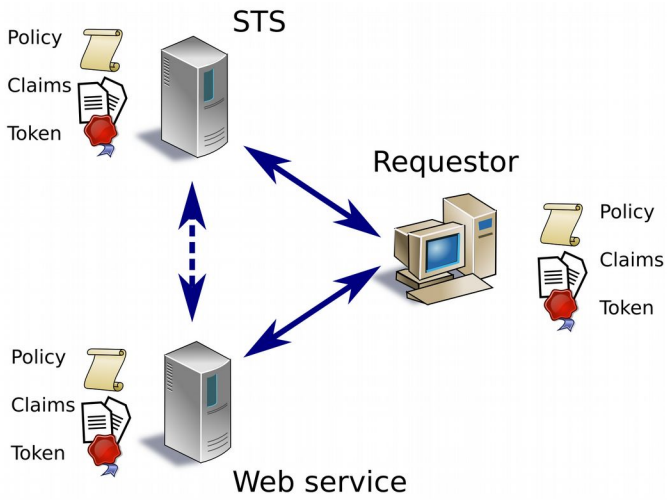

Figure 1: WS-Trust architecture

The security token released by the STS must have some features, in particular: (*i*) being verifiable as effectively released by the STS, and (*ii*) effectively authorizing the requester to use certain services. These features depend on the type of token being released: various technologies may be used to implement the token, such as X.509 and SAML. SAML is well fit for this scenario as it provides a secure way to make assertions about some subjects and their attributes. Otherwise these features may be guaranteed on the basis of a previous agreement, i.e., a secret, shared between the Web service and the STS bound to the service. In fact, an STS can be a platform-level Web service, bound to one or more Web services, for which it plays the role of a

trusted authority. A Web service may trust the signature of the STS, or it may request an STS to validate the token, or validate it autonomously.

A Requestor may be informed about the necessity to use a security token released by an STS, as the needed Web service can publish a WS-Policy document where a certain IssuedToken is requested. The interaction between a client and an STS occurs through a request-response protocol.

In particular, a RequestSecurityToken is used to request a token, and a RequestSecurityTokenResponse for responding to the request. Each request must be associated with an action. According to the WS-Trust standard, an STS may be requested to release, renew, cancel or validate a token. The Requestor may add claims, expressed in a certain “dialect” depending on the application. The Requestor may also specify a service which the request applies to, if the STS is associated with multiple Web services; in this case, the exact endpoint reference of the Web service has to be specified.

The response may convey a token through a RequestedSecurityToken element. Additionally, it may convey other proofs through an RequestedProofToken element, containing data which the Requestor may use to demonstrate to be authorized for using the token. For example, it may contain a secret encrypted with the public key of the Requestor.

1.2 Definition of a generic service-oriented ATN protocol

An STS is normally integrated into a system using a single round of messages, i.e. a RequestSecurityToken (RST), sent from the Requestor to the STS, followed by a RequestSecurityTokenResponse (RSTR), sent from the STS to the Requestor. However, in some scenarios, more steps may be needed before a token is obtained. In fact, the WS-Trust standard foresees the extension of this basic mechanism, named “Negotiation and challenge framework”, which is depicted in Fig.2.

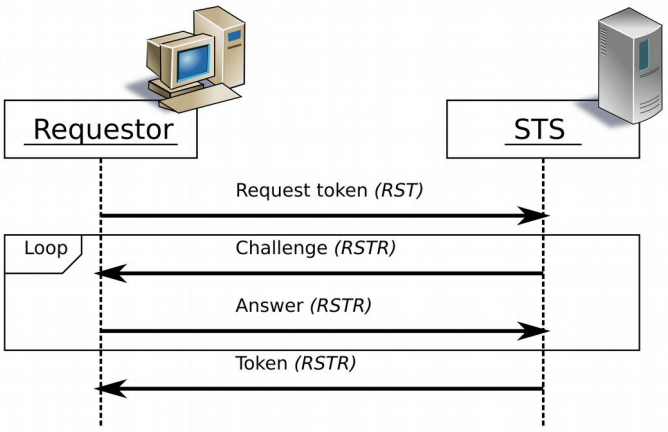

Figure 2: WS-Trust - Negotiation and challenge framework

The message exchange starts with a RST message, for requesting the token. Then, an arbitrary number of RSTR messages can be exchanged between the Requestor and the STS. Those RSTR messages may convey any additional information needed for completing the transaction, before finally transmitting the token. The WS-Trust standard defines some elements for proposing a “challenge” to the other end, including: SignChallenge, BinaryExchange, KeyExchangeToken.

However, it does not specify how to use such elements, or even other arbitrary elements, in a transaction. For example, Policy elements may be used by both parties to exchange their respective policies.

In this work, we propose a generic protocol for ATN. We decided to use some elements of the model proposed in (Lee and Winslett 2008), with particular reference to the content schemas. However, we organized the protocol to better distinguish the two fundamental phases of the negotiation: (*i*) the initialization, and (*ii*) the real exchange of credentials and policies. The aim is to deal with the transmission of credentials only if the two parties can reach an agreement about the protocol to use.

In particular, in the initialization phase, the parties use an extensible TNInit element in a single turn of messaging. It contains information useful for defining the parameters of the following negotiation, and for verifying if there is the necessary compatibility, before beginning a real negotiation. A TNInit element can contain: a SignatureMaterial, for proving the possession of a private credential; a StrategyFamily, for identifying a supported family of strategies; a TokenFormat, for specifying the supported type of security token. Essentially, the token can be handled as an opaque object by the client.

In the negotiation phase, the parties use an extensible TNExchange element. It can contain PolicyCollection and TokenCollection elements, for transporting policies and credentials, respectively, disclosed to the other party during the negotiation. Moreover, it can contain TokenType, RequestedSecurityToken and OwnershipProof elements, for conveying the requested token and the associated proofs of ownership.

4 A TESTBED FOR ATN IN OPEN WEB SERVICES

Following the design of a generic ATN protocol for Web services, we have also realized a working implementation. It is mainly intended as an experimentation framework, for testing both the functionality and performance of the proposed protocol in various scenarios. However, most of its components are reusable for creating specific open SOA-based applications, especially in the case of dynamic service selection and workflow composition through agents (Negri et al. 2006; Poggi, Tomaiuolo and Turci 2004, 2007).

The framework is available as part of the open source dDelega project (Tomaiuolo 2013), at https://github.com/tomamic/dDelega. dDelega is the result of ongoing work started with the development of a security layer for JADE, one the most widespread FIPA-compliant multi-agent systems (Poggi, Tomaiuolo and Vitaglione 2005).

In particular, it integrates a trust engine, in compliance to WS-Trust specifications. It also integrates an advanced rule engine for compliance checking against disclosure policies. These engines can be used by parties in a WS environment, by means of translator components that have been realized, in order to complete the integration. At a more basic level, the implementation exploits a number of frameworks developed under the Apache Foundation umbrella, including Axiom, Axis, Rahas, Rampart.

1.3 Integrating a modular trust engine

The trust engine must be able to evaluate which policies and credentials have to be inserted into the message at each round of the negotiation, on the basis of current state of negotiation and policies and credentials received at the previous round.

TrustBuilder2 (TB2) is a framework for Trust Negotiation, developed at the University of

Illinois for providing a flexible and extensible tool for this research area (Lee, Winslett and Perano, 2009).

TrustBuilder2 has not been realized for usage in the context of Web services. However his modular structure allows it to be extended for: (*i*) using different policy languages, (*ii*) implementing different negotiation strategies, and (*iii*) providing support for different types of credentials.

In particular, after a proper translation we defined, it is able to evaluate policies expressed according to the WS-SecurityPolicy language. Starting from received policies and credentials, it is able to analyze them and take decisions about which credentials and policies to disclose, according to the chosen negotiation strategy. It uses a policy compliance checker, which has the duty of finding one or more minimal sets of credentials satisfying a given policy. In TB2, the main components of ATN are represented as interfaces, which can be implemented and extended to add new functionalities. They can be distinguished as:

- Strategy module: regarding negotiation strategies.
- Policy compliance checker: to find a set of credentials satisfying a policy.
- Query interfaces: to provide access to resources, including local policies and credentials.
- Credential chain module: to build and validate chains of credentials, during a negotiation.

TrustBuilder2 is designed according to a model of negotiation with two main phases. The first phase is characterized by the exchange of messages containing data structures, called InitBrick, for communicating the information needed to initialize a negotiation. After this phase, the main negotiation rounds take place, characterized by the exchange of data structures called TrustMessage, i.e. objects containing policies and credentials to exchange during the negotiation.

As shown in Fig. 3, in this research work TrustBuilder2 is used as a trust engine for Automated Trust Negotiation. A mechanism has been realized for translating “TB2 messages”, i.e. InitBrick and TrustMessage objects, into “WS-Trust messages”, i.e. RSTR messages containing TNInit and TNExchange elements, which are exchanged in the context of a “Negotiation and challenge framework”, as defined by WS-Trust. The translation process uses some Apache libraries, including Axiom for creating XML structures, and Rahas for representing WS-Trust elements, in particular. This way, a client agent and/or an STS service may realize a negotiation using the TB2 framework internally, but exchanging standard WS-Trust messages publicly. After a successful negotiation, the client may finally access the desired service. Depending on the nature of the access token, the Web server may be able to verify it autonomously, or it may need the participation of the STS. In our tests, we implemented a token based on a signed structure, including a secret value, the client key, and a lifetime value, which could be verified directly by the Web service.

Policy and credentials are represented as abstract classes and interfaces in TB2, in such a way to make the tool independent from the type of policies and credentials used. The current implementation of this research work uses X.509 credentials, with a possible extension to SAML. In TB2, credentials are organized in chains; i.e. when a credential, released by an authority, is sent, then the whole chain has also to be sent. The CredentialChainMediator component uses algorithms to build, process and validate chains of credentials. This allows administrators to create decentralized authorities, valid for the different parties participating in a negotiation process. It allows TB2, when processing a chain, to verify the issuer of a credential released by an entity, starting from the verification of the root certificate of the chain.

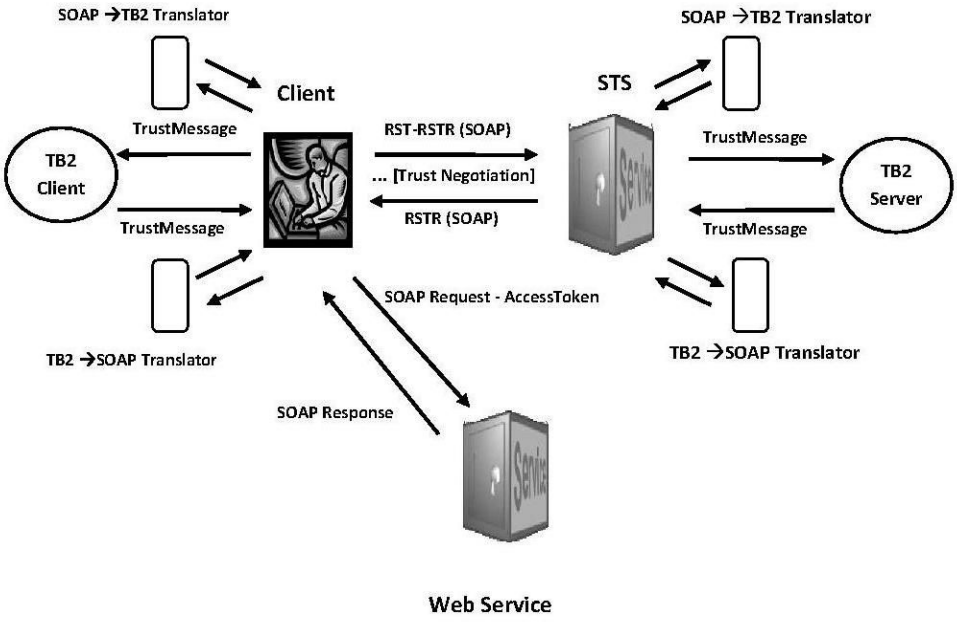

Figure 3: A testbed for ATN in open Web services

Our implementation also requires a user to specify, though configuration files, information about some components to be used by a client and a server, with respect to TB2 functionalities. This allows users to customize negotiation strategies, types of credentials and policy languages to be used in a certain application. The credential loader module can also be customized to load particular credentials into the system; it has access to a list of available credentials. The profile manager module uses the same customization to decide which class loader to use, according to the type of credentials used by the PolicyManager. A policy loader contains information for the PolicyManager, to decide which policy class loader to use.

1.4 Using a rule engine as a policy checker

A fundamental aspect of TB2 is the logic it uses for the functioning of its compliance checker component. In TB2, the problem of finding a set of credentials satisfying a policy is reformulated into the so-called “many pattern/many object match” problem, i.e., to find the objects matching the given patterns. Here, credentials are considered as objects and policies as patterns, in a problem which can be solved using a production rule engine. The rules of such engines have a standard format, with: an LHS (left hand side), the part of the rule defining the conditions; and an RHS (right hand side), the part of the rule defining the action to perform in the case when the conditions of the LHS are satisfied.

TB2 includes the Clouseau component, that is an expert system using the Jess (Java Expert System Shell) rule engine, which provides APIs for integration into a Java application. The rules, representing the policies of a Trust Negotiation process, define constraints on credentials. Jess implements the Rete algorithm (Forgy 1982), which allows to solve the “many pattern/many

object match” problem. Using an engine of this kind in a Trust Negotiation process requires to introduce rules for representing policies, which specifies the patterns. The knowledge base, instead, is determined by acquired credentials. An inference can be realized by finding a set of credentials satisfying the policy, which is exactly the duty of the policy checker in TB2. Thus, a policy checker is nothing more than an expert system based on production rules. Jess does not support natively any object for representing credentials or policies. Instead, to use credentials in Jess and to insert them into its working memory, it is necessary to define their format explicitly. Then, through JessComplianceChecker class, an assert command must be constructed and executed. This requires quite cumbersome code, for constructing these commands as an “assert(...)” string, starting from the object representing the credential.

Instead, in this work we have customized the TrustBuilder2, extending it for using a different rule engine as a policy checker. In particular, we used the Drools rule engine (Sottara, Mello and Proctor 2010) for the policy checker component instead of Jess, supported by the currently available version of TB2. Drools is based on the so-called ReteOO algorithm, i.e., an adaptation of the Rete algorithm for object oriented systems. In Drools there are two main storage areas: a Production Memory, where rules are stored, and a Working Memory, where known facts are stored. For Trust Negotiation, the Production Memory can be used for storing the policies as rules, while the Working Memory can be used for storing the credentials as facts. As an important advantage with respect to Jess, facts in Drools are represented as Java objects and they can be put directly into the Working Memory. This has allowed us to develop a policy checker with a much leaner code than the Jess policy checker. Moreover, the tool is completely open-source, on the contrary of Jess; it is continuously updated, with the addition of new features, and it has the attentions of a vast and lively community of developers.

1.5 Performance evaluation

The ATN process, as described in the previous sections, was analyzed from the point of view of performance. The evaluation regarded the influence of the various components of the system and the conversions required by those components for communicating. For these tests, a scenario has been realized, in which:

- the client requests a token;
- the STS sends a policy requesting a chain of credentials;
- the client, protects one of the credentials in the chain with a policy, which he discloses to the STS;
- then, the STS discloses the credentials satisfying the client's policy;
- thus, the client discloses the credential chain initially requested by the STS;
- finally, the STS sends the requested token.

Including the initialization phase, the whole process takes 4 rounds, in which both the client and the STS send a message to the other party.

As shown in Tab.1, the execution times vary around a mean value of 6 seconds, including the signature and encryption of SOAP messages, and 4.8 seconds without any signature and encryption. Considering instead a minimal negotiation process of three rounds, the execution time is around 4 seconds. Decreasing the credentials required by the policy from 3 to 1, the execution time does not vary proportionally, but it is reduced only by around 1 second. This means that a significant part of the computation load is absorbed by TB2, for the evaluation of policies, in addition to the basic workload imposed by the WS-* stack (Novakouski et al. 2010,

Rodrigues et al. 2011).

The results are in accordance with those obtained by some authors of TB2 (Lee 2008), which report that almost half of the total time of execution is used by the policy checker. Another significant comparison is with the execution of a negotiation using only the TB2 tool, in which a TB2Client and a TB2Server communicate directly, through a dedicated socket, without any conversion, signature or encryption: in the same scenario with 4 rounds, as described above, the process takes 1.2s in TB2, against the 6s required by the whole Web services infrastructure implemented in this work.

Table 1. Initial performance results

4 rounds, with Enc & Sign	6.0s
4 rounds, w/o Enc & Sign	4.8s
3 rounds, w/o Enc & Sign	4.0s
3 rounds, 1 credential requested by STS	3.0s
4 rounds, TB2, no WS	1.2s

It is worth noting that more efforts may be dedicated to the optimization and fine tuning of various components of the system. Thus, performance may be improved in many aspects. For example, the inclusion of policy statements into Drools is now a process involving various steps and conversions. In future releases of the framework, this process will be streamlined, enabling a more direct inclusion of policies and improving efficiency.

5 A TESTBED FOR ATN IN PROXIMITY-BASED SOCIAL NETWORKING

Another potential area of application for Trust Negotiation is constituted by online social networks, especially in the case of distributed pervasive platforms, supporting location-based activities. In fact, this kind of social networks provides a good example of an open scenario, where it is interesting to develop and evaluate pervasive adaptive applications. Our experimentation with Trust Negotiation, in this case, was performed over Blogracy (Franchi, Poggi and Tomaiuolo 2013), a distributed social networking system, exploiting at its core an OpenSocial Container and the BitTorrent protocol. OpenSocial is a public specification that defines a component hosting environment (container) and a set of common application programming interfaces (APIs) for web-based applications and was originally developed to support interoperability among social networking platforms. Moreover, Blogracy has an additional P2P layer that can be used for the devices to share resources in a decentralized way and it uses Attribute-Based Encryption in order to make resources (especially on the P2P layer) and communications accessible only to those having the appropriate permissions. We have decided to implement a pervasive layer built on top of Blogracy, named for simplicity UBA, modeled as an agent based system. In fact, agent-based technologies are well-suited for online social networks (Franchi and Tomaiuolo 2012, Poggi and Tomaiuolo 2013), especially considering: (*i*) the networks massive scale and (*ii*) the interoperability requirements that are becoming increasingly relevant, especially in the context of pervasive computing. In some cases, agents may also share with users a common understanding of the exchanged messages, on the basis of Semantic Web technologies (Poggi 2009, Tomaiuolo et al. 2006).

Users in a social network can be linked with multiple kinds of relationships. In Blogracy,

these relationships are expressed as belonging to a group. These groups essentially work like Google Plus circles and are: (*i*) unique to a user, and (*ii*) associated with other users. When a user adds other users to a given group, those users can access the resources associated with the group. We also have two additional kinds of groups: (*i*) Proximity groups and (*ii*) Location groups. Proximity groups are centered on each member of the social networking system and represent physical closeness to such member. Proximity groups are extremely fluid, in the sense that users can physically move and consequently the set of users belonging to a Proximity group varies in time. Each user configures the hysteresis of his Proximity group, i.e., how long the other users are considered part of it after they are no longer physically close to him. On the other hand, a Location group (*i*) is associated with the users in the proximity of a given location (e.g., a classroom or a museum room), (*ii*) has a host, i.e., a node that both identifies and supports the group and (*iii*) is associated with a location profile, which can be hosted either on the device itself or on a different node. In fact, a location, although logically different from a regular user, works in the same way and a Location group is essentially a Proximity group for the location.

For providing these functionalities, each node of the social network hosts multiple agents, with different levels of agency. As shown in Fig. 4, some of the more important agents are (*i*) the Neighborhood Manager agent (NM), which cooperates with lower level agents to discover the users in its neighborhood; (*ii*) the Trust Negotiator agent (TN), that is involved in the decisions regarding privacy and data access, and (*iii*) the OpenSocial agent, that provides a bridge towards the underlying Blogracy modules (Franchi and Tomaiuolo 2012, Poggi and Tomaiuolo 2013). In particular, a Neighbourhood Manager (NM) and a Trust Negotiator (TN) are directly involved in the Trust Negotiation process. Since a typical mobile device has several ways to scan its surroundings – Bluetooth, WiFi Direct, regular WiFi, Near Field Communication (NFC) etc. – a node has specific agents to discover other nodes using these protocols. The NM Agent is responsible for aggregating information from these agents, trying to present a consistent view. The Neighborhood Manager agent informs the OpenSocial agent when users enter and leave a Proximity or Location group and the latter notifies the OpenSocial container about it.

Then, the TN Agent of the discovering node can start a trust negotiation with the discovered agent, disclosing some information. If the negotiation succeeds, the discovering node: (*i*) knows the actual user-id of the discovered node; (*ii*) adds the discovered node to its Proximity group, which means that it is able to access only resources part of the Proximity group. However, typically during the negotiations each of the two TN Agents obtains some information on the other user and consequently decides to which additional groups to add him. This process can be automatic or semi-automatic (awaiting human confirmation) depending on the preferences.

One particular source of trust in social networks may be represented by reputation, which may be available in various forms. In the case of open systems, certified reputation may represent a quite simple source of trust. Certified reputation, in fact, requires an agent to provide a number of references about its past behaviour, certified and authenticated by third-party agents. It is a simple protocol which resembles the references provided by a person who applies for a job.

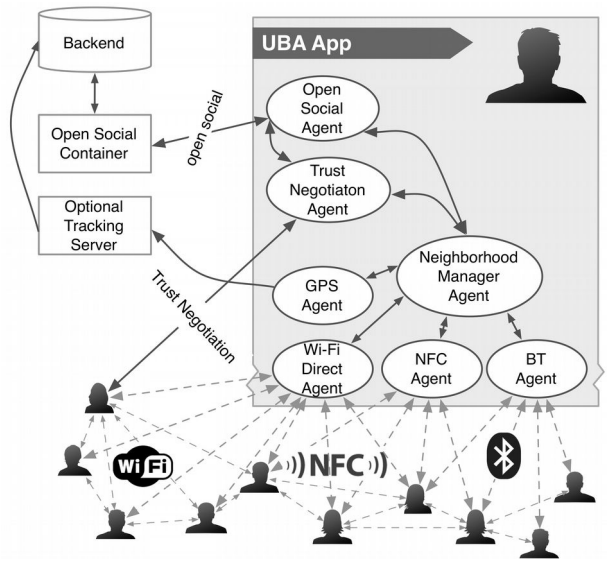

Figure 4. A testbed for ATN in proximity-based social networking

Those references has to be previously obtained and collected by the referee agent, by asking its partners to certify their ratings about its performance during past interactions. Then, collected references can be made available when some other agent needs to evaluate its trustworthiness, for granting access to resources or membership into a protected group. These ratings may allow the referee to gain the trust of new partners, as they attest its achievable performance in a particular task, according to its previous interaction partners. However, a rational agent will possibly present only the best ratings, which may overestimate its performance in a particular task, without guaranteeing a minimum level. In fact, trust evaluations based on certified references tend to be less accurate than other models, like witness reputation, collecting all bad and good ratings. Neverthless, certified references can be useful for trust evaluation in the absence of other sources of information and because of their wide applicability. In fact, references may be obtained also from just a small number of interactions, and they require few computational and communicational resources, since they are collected and provided directly to the evaluator (Huynh, Jennings and Shadbolt 2006). Authenticated references may also be easily used in a Trust Negotiation process, while other forms of direct trust and reputation may require additional protocols. As with other credentials, references need to be evaluated on the basis of their issuer. This fits better the case of open MAS, where there is no guarantee about the honesty of agents, lies and collusions have to be considered possible, and in general no single institution is supposed to be absolutely trustful or omniscient.

6 CONCLUSIONS

The presented framework provides the ground for experimenting Trust Negotiation protocols in the context of generic Web services. It allows users to create trust automatically, by incrementally disclosing credentials. Modular applications can integrate services provided in an open environment, on the basis of peer-to-peer trust relationships. Interoperability among such services is guaranteed by the conformance to standard protocols for Web services. The realized ATN system is composed of various components and requires various format conversions for messages, policies and credentials. After an evaluation of the computational overhead of a

complete negotiation process, we suggest the use of Trust Negotiation for releasing session tokens, granting access to a number of cohesive services in a given time interval. Besides service composition and generic Web-based applications, the framework can also be used as a testbed for Trust Negotiation in distributed social platforms. In fact, especially in the case of location-aware applications, unknown users may need to establish some level of trust before interacting, when meeting at a certain place or at a certain event. In this sense, this research work paves the way for further analysis in this kinds of application scenarios, where the developed framework may be used to adapt a powerful trust engine and a well known rule engine for use with very different kinds of protocols and credentials.

References

Bertino, E., L. D. Martino, F. Paci, and A. C. Squicciarini. 2010. “Standards for web services security.” In *Security for Web Services and Service-Oriented Architectures*, 45-77. Springer.

Forgy, C. L. 1982. “Rete: A fast algorithm for the many pattern/many object pattern match problem.” *Artificial intelligence* 19, no. 1: 17-37.

Franchi, E., A. Poggi, and M. Tomaiuolo. 2013. “Open social networking for online collaboration.” *International Journal of e-Collaboration (IJeC)* 9(3):50–68. IGI Global.

Franchi, E., and M. Tomaiuolo. 2012. “Software Agents for Distributed Social Networking.” In *Proceedings of the 13th Workshop on Objects and Agents (WOA)*, Milan, Italy.

Huynh, T. D., N. R. Jennings, and N. R. Shadbolt. 2006. “An integrated trust and reputation model for open multi-agent systems.” *Autonomous Agents and Multi-Agent Systems* 13(2):119–154.

Lee, A. J. 2008. *Towards practical and secure decentralized attribute-based authorization systems*. ProQuest.

Lee, A. J., and M. Winslett. 2008. “Towards standards-compliant trust negotiation for web services.” In *Trust Management II*, 311–326. Springer.

Lee, A. J., M. Winslett, and K. J. Perano. 2009. “Trustbuilder2: A reconfigurable framework for trust negotiation.” In *Trust Management III*, 176–195. Springer.

Li, N., J. C. Mitchell, and W. H. Winsborough. 2005. “Beyond proof-of-compliance: security analysis in trust management.” *Journal of the ACM (JACM)* 52(3):474–514.

Negri, A., A. Poggi, M. Tomaiuolo, and P. Turci. 2006. “Dynamic Grid tasks composition and distribution through agents.” *Concurrency and Computation: Practice and Experience* 18(8):875–885.

Negri, A., A. Poggi, M. Tomaiuolo, and P. Turci. 2006. “Agents for e-business applications.” In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 907–914. ACM.

Novakouski, M., S. Simanta, G. Peterson, E. Morris, and G. Lewis. 2010. *Performance analysis of WS-Security mechanisms in SOA-based Web services*. No. CMU/SEI9-2010-TR-023. Carnegie-Mellon.

Poggi, A. 2009. “Developing ontology based applications with O3L.” *WSEAS Transactions on Computers* 8(8):1286–1295.

Poggi, A., and M. Tomaiuolo. 2013. “A DHT-based multi-agent system for semantic information sharing.” In *New Challenges in Distributed Information Filtering and Retrieval, Studies in Computational Intelligence (SCI)* 439:197–213. Springer.

Poggi, A., M. Tomaiuolo and P. Turci. 2007. “An Agent-Based Service Oriented Architecture,” in

Proceedings of the 8th Workshop on Objects and Agents (WOA), 157–165.
Poggi, A., M. Tomaiuolo and P. Turci. 2004. “Extending JADE for agent grid applications,” in *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 352–357.
Poggi, A., M. Tomaiuolo, and G. Vitaglione. 2005. “A security infrastructure for trust management in multi-agent systems.” In *Trusting Agents for Trusting Electronic Societies, Lecture Notes in Computer Science (LNCS)* 3577:162-179. Springer.
Rodrigues, D., D. F. Pigatto, J. C. Estrella, and K. R. Branco. 2011. “Performance evaluation of security techniques in web services.” In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, 270–277. ACM.
Sabater, J., and C. Sierra. 2002. “Reputation and social network analysis in multi-agent systems.” In *Proc. 1st Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 475-482. ACM.
Sottara, D., P. Mello, and M. Proctor. 2010. “A configurable Rete-OO engine for reasoning with different types of imperfect information.” *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1535–1548.
Tomaiuolo, M. 2013. “dDelega: Trust Management for Web Services.” *International Journal of Information Security and Privacy (IJISP)* 7(3):53–67. IGI Global.
Tomaiuolo, M., P. Turci, F. Bergenti, and A. Poggi. 2006. “An ontology support for semantic aware agents.” In *Agent-Oriented Information Systems III, Lecture Notes in Computer Science (LNCS)* 3529:140–153. Springer.
Venanzi, M., M. Piunti, R. Falcone, and C. Castelfranchi. 2011. “Facing openness with socio-cognitive trust and categories.” In *Proc. 22nd Int. Conf. on Artificial Intelligence*, 400–405. AAAI Press.
Winsborough, W. H., K. E. Seamons, and V. E. Jones. 2000. “Automated trust negotiation.” In *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX)*, 88–102. IEEE.
Winslett, M., T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. 2002. “Negotiating trust in the Web.” *Internet Computing* 6(6):30–37. IEEE.
Yu, T., M. Winslett, and K. E. Seamons. 2001. “Interoperable strategies in automated trust negotiation.” In *Proc. of the 8th ACM Conference on Computer and Communications Security*, 146–155.

Author biographies

FILIPPO AGAZZI is a freelance computer engineer working in Parma. His research focuses on Web services, rule-based systems and information security. His email address is agazzi@ce.unipr.it.

MICHELE TOMAIUOLO is an assistant professor at University of Parma. His research interests include trust management, peer-to-peer and social networks. His email address is tomamic@ce.unipr.it.