Towards Automated Trust Negotiation in MAS

Federico Bergenti
Dipartimento di Matematica
Università degli Studi di Parma
federico.bergenti@unipr.it

Leonardo Rossi, Michele Tomaiuolo
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
leonardo.rossi@studenti.unipr.it
michele.tomaiuolo@unipr.it

ABSTRACT

This paper tackles the notable concept of automated trust negotiation and presents preliminary results on its integration in the realm of multi-agent systems. First, a review of the relevant literature on automated trust negotiation is given and basic ideas are discussed. Then, a motivated introduction of a novel protocol for automated trust negotiation in multi-agent systems is presented and the basic features of the protocol are discussed.

1. INTRODUCTION

Traditionally, the problem of identity management is reduced to *Public Key Infrastructures* (*PKIs*). However, in practice, all efforts to deploy an X.509 [6] infrastructure have fallen below expectations. According to a number of proposals, e.g., PolicyMaker [1], KeyNote [2], Simple Distributed Security Infrastructure [8] and Simple Public Key Infrastructure [3], the very foundation of digital certificates needs to be reconsidered, in the struggle to make digital certificates really useful in application scenarios. The main rationale behind such an urged reconsideration is: *what computer applications need is making decisions about keyholders as users; not simply accessing their real-life identities*. Applications often need to make decisions about whether to grant access to a protected resource and the real-life identity of the requestor is just one of the diverse inputs that the decision process needs. In available PKIs, such decisions are taken on the sole basis of the keyholder's real-life name. However, the keyholder's name does not make much sense to a computer application: it is just a key that indexes an entry in a database. PKIs simply exploit two mandatory requirements: *(i)* the name being unique; and *(ii)* the name being uniquely associated with the information needed to support the decision process. Unfortunately, it is extremely unlikely that the name that we use to identify a person could scale up to the Internet because it would miss such uniqueness requirements.

This work deals with trust management in open and decentralized agent-based environments and it addresses the mentioned issues of traditional identity management solutions. The proposed analysis and related solutions are geared towards P2P networks, intended not just as a mere technology, but rather as an abstraction that captures webs of trust relationships where parties interoperate directly, without any reliance on centralized authorities.

In particular, this paper shows how it is possible to join multiple certificates in a delegation chain, expressing a degree of trust between two agents in a fully decentralized manner. This kind of delegation supports secure collaboration also among agents that do not have a direct acquaintance.

This paper is organized as follows: next section describes the main elements of automated trust negotiation. Section 3 reviews the oblivious attribute certificates scheme and it discusses its main usage pattern and characteristics. Finally, Section 4 drafts a generic protocol for automated trust negotiation in multi-agent systems that exploits the techniques presented in the previous sections.

2. BASICS OF TRUST NEGOTIATION

The traditional client-server model views computer systems in terms of computational resources and related data centralized in few servers, which respond to requests of clients. Clients are supposed to have basic capabilities and they mainly rely on the resources of servers for their tasks. The multi-agent model reverses such a view and it describes systems in a P2P fashion: each agent has some resources to share and some services to offer to the community of agents. Thus, according to the context, each agent could easily to play the role of either client or server.

The enforcement of all aspects of security and trust in the access to the resources made available by agents is a mandatory requirement to turn agent-based P2P networks into a more widespread paradigm of cooperation among loosely coupled agents. The secure management of trust relationships, i.e., the ability to precisely control the flow of delegated permissions to trusted entities, is a fundamental requirement to allow the composition of disparate services provided over the network.

Moreover, since the recent widespread adoption of the Internet in consumer markets, all contacts among people are often fully digitalized and there is still no definite solution to the problem of identity management. Actually there is no body of knowledge to associate with a name and the simple idea of trying to build an on-line, global database of names is obviously unfeasible.

Given such a new way people is today using the Internet, a novel scheme of authorization has recently become relevant because it provides a scalable and easily extendible model to protect a generic resource. Such a new scheme, called *Automated Trust Negotiation* [11][13][14][15], is gaining more and more interest in the community of researchers and practitioners. This approach allows unknown users aimed at accessing some resource to establish a level of trust in an incremental way through the exchange of credentials. A credential is defined as a digital certificate attesting, via a digital signature, the association of one or more attributes to an entity [16]. A brief example should clarify the basics of this approach. Alice releases a signed credential to Bob to assert that Bob works for her; by showing such a credential, Bob can demonstrate to Carl that he works for Alice. In such a scenario, Alice plays the role of *introducer* [1] of Bob toward Carl; Carl, through Alice's signature on the credential, is assured about the assertion being authentic.

In order to create the needed signature, an asymmetric cryptographic system, based on couple of public/private keys, must be used. The signature is applied by encrypting a hash of the credential with the private key, and associating it with the corresponding public key that is used to verify the credential, which therefore cannot be falsified. Each credential is associated with the public key of its subject too. Consequently, the subject can demonstrate to possess the corresponding private key, so attesting the ownership of the presented credential. Notably, the entity that originally issued and signed the certificate is not requested to participate directly in the process of trust negation.

2.1 Credentials

A credential can be considered as the digital counterpart of a paper document, e.g., a driving license. Such a paper credential basically asserts the ability to drive a car, and it is accompanied by other information, e.g., name and date of birth. Each piece of information the paper document contains is made of an *attribute name* and a corresponding *attribute value*. Finally, there is a signature asserting the authenticity of the document.

The owner of a credential can further sign a credential owned by a third subject. This way, a credential chain can be created, which can be used to demonstrate a relation, possibly an indirect one, between the subject who presents it and the (well-)known subject who released the first credential in the chain.

Attribute in a credential can be considered sensible or not. The case of non-sensible attributes does not require any particular care. On the contrary, for the case of sensible attributes, it is necessary to build a certain level of trust between negotiating parties via a structured list of release conditions. Such release conditions are generally known as *policies*. Different languages have been defined to represent policies [9][6][3] in an appropriate and expressive way.

Unfortunately, in real-world cases, the definitions of policies for credentials may not be sufficient and thus [13] calls for the introduction of a so-called *acknowledgment policy*. It is easily shown that the simple fact of possessing, or not possessing, a credential exposes an entity to security risks. Actually, the unauthorized diffusion of reserved information is a problem caused by access control policies themselves. When an interlocutor does not possess a credential, it is not associated with any related policy, and consequently he/she behaves in a different way than someone actually owning the credential. By simply observing the pattern of communication, a third party can infer if someone owns a credential or not. These are the cases where trust negotiation provides its full benefits. Digital credentials are exchanged step by step, to increase the level of trust between involved parties, and the flow of credentials between two entities through a sequence of requests and releases is what is actually intended with trust negotiation.

2.2 Negotiation Strategies

The execution of a negotiation requires some agreement on a common protocol, with the intended agreement that each subject is free to apply a possibly different strategy. The characteristics of a negotiation are defined by the adopted strategies. Some of the tasks of such strategies are related to which credentials are released, when they are released, which parties are required to unlock the release of another credential and when the negotiation closes, successfully or not.

The success of a negotiation is not always possible. One of the subjects could not have all the needed credentials, or one of the subjects could implement a policy imposing a cyclic dependency. Therefore, it is worth defining properties that should be expressed, in the best possible way, by a strategy:

1. A strategy should bring a negotiation to success, when such a possibility exists. Strategy having such a property are said to be *complete*.
2. Ideally, a strategy should avoid the release of information which is not strictly required to bring the negotiation to an end.
3. A strategy should truncate a negotiation when it cannot bring to a successful conclusion.
4. A strategy should recognize a cyclic dependency among credentials and policies.
5. The strategy should be reasonably efficient.

In the literature, the following strategies are most commonly considered.

Eager Strategy. This strategy is complete and efficient. Participants release all their credentials as soon as the relevant policy is satisfied, without waiting the credential to be requested. This strategy is very simple and brings the negotiation to success whenever it is possible. Nevertheless, it reveals more credentials than those strictly needed to create the minimum level of trust.

Parsimonious Strategy. In this strategy, the number of exchanged credentials is minimized. It is reasonably efficient and it concludes with success whenever it is possible. At the beginning, parties exchange credential requests, but not the credentials themselves. All possible release sequences are then explored. When the exploration requires some unprotected credentials to be exposed, the path is compared with others. The path that brings the negotiation to success with the minimum number of exposed credentials is selected and followed. Unfortunately, due to the possible limitations in the level of cooperation between two subjects, the global minimum solution is not guaranteed.

PRUNES Strategy. The *PRudent NEgotiation Strategy* allows establishing trust without revealing irrelevant credentials, while remaining reasonably efficient. In [17] the communication complexity is shown to be $O(n^2)$, and the computational complexity to be $O(nm)$, where n is the number of credentials and m is the size of the policy regulating the release of credentials.

3. THE OBLIVIOUS ATTRIBUTE CERTIFICATES SCHEME

Paper documents often encapsulate different attributes about their owners. For example, a driving license commonly reports the date and place of birth and the current postal address. Such pieces of information which, in the digital counterpart of the paper document, are superfluous to validate a policy borrow a loss of privacy. In the X.509 standard, the values of some attributes of a certificate, e.g., name and date of birth, are not considered sensible and so they are revealed freely.

In order to address such an issue, [8] presents a new type of certificate: the *Oblivious Attribute Certificate* (*OACert*). In an OACert the certificate scheme guarantees to its owner the

possibility to select which attributes to use and how to use them. The basic idea of OACert is very simple: instead of saving the attribute values directly in the certificate, the certification authority saves the *cryptographic commitment* [1][6] of the attribute value.

The scenario comprises three types of entities: some *Certificate Authorities* (*CA*), some certificate holders and some service providers. The concept of CA is not necessarily bound to a hierarchical environment as in X.509. When speaking of OACerts, the concept of CA simply identifies an entity capable of issuing a certificate. An OACert is an assertion about the certificate holder, digitally signed by a CA. Each OACert contains one or more attributes. When the system of cryptographic commitments is secure, the certificate does not disperse any information about sensible attribute values; so, the content of the OACert can be made public. In such cases, the certificate holder can show its OACert without having to worry about the privacy of its attributes.

The generic scenario proceeds as follows: *(i)* a CA generates an OACert for its future holder; *(ii)* each CA and each certificate holder own a unique public/private key pair; *(iii)* a service provider, when presented with a request from a certificate holder, performs an access control on the basis of the attributes of the certificate holder, certified in the OACert.

An attribute in an OACert can be used in different ways for:

1. Opening the commitment and thus revealing the value of the attribute.
2. Using a *Zero-Knowledge Proof* protocol to prove that an attribute value satisfies a condition, without revealing more information.
3. Using a special protocol, called *Oblivious Commitment Based Envelope* (*OCBE*) [8], that warrantees that the receiver would finally receive a message only when the attribute value satisfies a requested condition, without revealing more information about the attribute value itself.

The following example discusses the scenario and it should clarify the roles of the various entities involved. Let's suppose that Alice needs to demonstrate to Bob that she is older than 21, but she wants to keep her actual age private. We need a protocol ensuring that Alice succeeds in demonstrating the required condition without Bob knowing her actual age.

Alice, the certificate holder, establishes a secure communication channel with Bob, the service provider and, at the same time, she proves their ownership of the OACert to Bob. Bob verifies the signature and the validity period of the OACert, then he verifies that the certificate has not been revoked using, e.g., the standard technique described in [6]. Moreover, Bob verifies that Alice owns the private key corresponding to the public key included in the OACert. Subsequently, if such an initialization process worked fine, Alice requests the public key to Bob in order to decipher the document and Bob answers by sending his policy.

After such an initialization phase, Alice can now read subsets of attributes by using proper protocols. In order to read multiple attributes, Alice performs the same protocol repeatedly. There are three protocols that can be used to read attributes, and each one of them is characterized by a different complexity of computation and communication, and a different level of privacy loss [8].

1. *Direct show*. This protocol is used when Alice trusts Bob and she simply reveals the attribute values to him, or when Alice has very restricted computing power. This protocol is very efficient, but it supports the minimum level of privacy. Bob actually knows the attribute values, and he can also convince others about this.
2. *Zero-knowledge show*. Alice uses zero-knowledge proofs to demonstrate that her attributes satisfy some property that Bob requires. This protocol is much more expensive than the direct show protocol, but it offers better protection of privacy. Bob learns if some attribute values satisfy his policy, but he cannot convince others about his ownership of values. Actually, Bob does not learn the exact attribute value, if multiple values satisfy the policy.
3. *Oblivious show*. Alice interacts with Bob using the OCBE protocol. Bob does not learn anything about attribute values. Among the three types of protocols, the oblivious show offers the best protection of privacy. Moreover, it often requires a computational power similar or even less than the zero-knowledge show protocols.

This last case, i.e., the oblivious show is worth some discussion. Informally, the OCBE protocol enables Bob to send an enciphered message to Alice in such a way that Alice can read the message if and only if the value of its commitment satisfies a predicate. The protocol as a whole is considered oblivious if, at the end of the protocol, Bob cannot capture any information about the Alice's commitment value.

4. A GENERIC PROTOCOL FOR AUTOMATED TRUST NEGOTIATION

The aim of automated trust negotiation is to establish trust between two unknown parties through the release of certificates. OCBE, together with OACert, simplifies the process of trust negotiation reducing the number of iterations and the overall number of exchanged certificates.

Here, we study the characteristics of a protocol for automated trust negotiation to be used in a distributed and decentralized environment. In order to fully exploit all such characteristics, the introduction of this new protocol requires the realization of new types of certificates and related supports. The resulting software framework, implemented in Java using an XML representation of certificates adhering to SAML specifications, has been prototyped but its description is unfortunately out of the scope of this paper.

4.1 General Requirements

An open network like the Internet requires the greatest flexibility from an authentication protocol: it needs to fit all cases in reasonable ways, and it should be able to expand its functionalities by incorporating existing resources. Let's discuss some of the problems which could emerge in the introduction of a new protocol by means of sample scenarios.

Alice and Bob do not share the same CA, they do not know each other and moreover they are in located in two different security

domains. In practice, this is the common case of Alice and Bob not sharing the same CA and, e.g., Bob does not know the CA that issued the certificate to Alice. In this case, Bob would probably not trust the CA of Alice and he would not accept Alice's certificate.

This problem and its numerous variants can be solved in different ways, according to the actual situation. For example, the previous scenario is addressed by the introduction of a hierarchy of CAs with Bob trusting the root of the hierarchy. In fact, this is the approach used in the X.509 certificate scheme. Otherwise, the problem can be solved by means of the introduction of a more flexible chain of trust as foreseen in the PGP certificate scheme.

Another interesting case is exemplified as follows. Alice intends to buy a book from Bob's bookstore. Alice has a discount if she demonstrates that she is a member of ALI (Associazione Lettori Italia). Let's suppose Bob trusts ALI only and Alice is instead a member ALP (Associazione Lettori Parma), which is a local section of ALI. ALP issued an OACert to Alice, attesting that she is a member of ALP. If ALP has a valid certificate, not containing other sensible information, issued by ALI and accrediting it as a section of ALI, then a certificate chain can be created, to demonstrate that Alice satisfies Bob's policy. Alice sends such a certificate chain to Bob without revealing information about other attributes in her OACert. Finally, she starts a zero-knowledge proof protocol to demonstrate that she is a member of ALI, indirectly.

Another subtle problem arises when different CAs identify the same property with different names or encodings. For example, the prefecture may use DoB to name the attribute identifying the date of birth in a driving license, while the municipality may use the word date-of-birth to identify the same piece of data in an identification card. It could also happen that different CAs use different encodings to convert an attribute to and from an integer value.

In order to solve such a problem, each CA publishes its encoding mechanism online, and it signs it using its private key. When Alice shows her certificate to Bob, she points to Bob which encoding mechanism is used for her attributes signed by a certain CA. Bob can then regulate his policy according to the encoding.

Finally, let's consider the following scenario [8]: Alice and Bob want to exchange their certificates about their wages. Alice's policy allows showing the certificate only to those who have a salary greater than $100k per year. Similarly, Bob shows his own certificate only to those earning more than $80k per year. Using the standard trust negotiation techniques, neither Alice nor Bob would be able to release their certificates before the other one. Instead, let's suppose that both Alice and Bob use OACert and the OCBE protocol. In this case the problem can be solved. In fact, Alice and Bob exchange their certificates without the other part being able to discover the real salary value. Bob uses the OCBE scheme to send his salary value to Alice, together with a proof about the attribute value in the OACert certificate, with the condition that Alice can open the message if and only if her salary is greater than $80k. Bob can be sure that the value of his salary is revealed to Alice only if her salaries satisfy Bob's policy. The policy is enforced without knowing Alice's salary value. It is worth noting that Alice's policy does not conflict with Bob's one; actually it is not activated, as at the end of the transaction Bob does not know Alice's salary value.

4.2 Openness Requirements

In order to boost the adoption of a novel protocol for establishing trust between two unknown entities, a number of notable issues have to be addressed in a practical and generic way. The following is a list of characteristics that a protocol for exchanging credentials should provide to demonstrate flexibility and adaptability to different situations.

Support for different types of credentials. The protocol should support and include different types of certified credentials. These should comprise standard certificates, e.g., X.509, as well as new proposed formats, e.g., OACert, hidden credentials and anonymous credentials.

Support for attributes without certification. The protocol should support non-certified attributes. This is important to allow the use of attributes that cannot be certified by any means.

Support for signed credentials. The protocol must support the use of signatures to certify a credential, without requiring necessarily a hierarchy of CAs. Each entity in the system should be able to sign a credential. This way, the system can fit different situations, including the case where a more rigid hierarchy of CAs effectively exists.

Support for different cryptographic algorithms. The negotiation protocol must support the use of different cryptographic systems to improve the efficiency of trust negotiation. The more the protocol is unbound from a particular implementation, the more it can adapt to the evolution of the technology.

Separation of the two fundamental concepts of credential and attribute. Credentials are sometimes considered the same entity of the attributes they convey. Under such a misconception, when Bob requests Alice to demonstrate the actual possess of a certain certificate, Bob should also satisfy all policies associated with all attributes that the certificate contains, even for those attributes that are superfluous to demonstrate the simple possession of the certificate. The greater the number of policies to satisfy, the greater is the risk of failing the negotiation, even when a solution is actually possible. If the separation of the concepts of credential and attribute is respected, then it is possible to demonstrate the possession of a credential without revealing any information about the attribute itself.

Selective revelation of attributes. The holder of a credential must be able to select which attributes to reveal to other parties. For example, a driving license includes a number of attributes like name, surname, age and address. If the other entity needs to know the age attributes, it is obviously inconvenient to reveal other data in the license, especially if they are considered sensible.

4.3 Usability Requirements

Here we present some features that a protocol should provide, at least selectively. Looking at real situations, each single piece of information that the protocol leaks is considered sensible, to a certain degree. For example, in many countries privacy must be protected to respect laws. The certificate holder must be allowed to decide if, when and to whom to reveal the information contained in a certificate. Also, the very possession of an attribute or a credential can be considered sensible information on its own. This also applies to the validating party. Each policy can be considered sensible. Or probably the very existence and

availability of a resource, or the reception of a request for a resource, can be sensible.

Obviously, there are lots of facets, more or less important in different scenarios, to be considered to enable the success of a transaction. For example, in a case the protection of the actual value of an attribute could be a mandatory requirement, while in another case this would be perfectly acceptable. The following is a list of the major cases we need to take into account in designing a protocol for supporting decentralized interactions.

No proof that an attribute satisfy a policy. A credential holder can demonstrate that his/her attribute satisfies a policy without revealing the effective attribute value. For example, Alice can demonstrate that she is older than 21 using her driving license only, without revealing any further piece of information about her age.

Ignorant use of a credential. In this case Alice obtains a resource from Bob without revealing the very fact of possessing a credential. Alice reveals the attributes to Bob, as required by the policy associated to the requested resource, without revealing the authenticity of provided data to Bob. Bob must arrange in a way to release the resource to Alice anyway, thug being sure that she will not be able to access it if her data are false, without even knowing if Alice's data were certified or not.

Protecting the sensibility of the request for a resource. Alice should be able to reveal her request for a resource if and only if Bob satisfies a certain policy.

Ignorant use of an attribute. Alice and Bob conclude a transaction in which Alice receives the requested resource if and only if the attributes in her credential satisfy Bob's policy. At the end, Bob does not learn anything about Alice's attribute values, even if the value would satisfy the policy or not.

Sensible attributes and policies. A particular situation can require that even policies are to be considered sensible. For example, let's suppose Bob's policy states that Alice must be older than 25, and Alice's is 30 years old. Alice can know if she satisfies Bob's policy without revealing her precise age or learning the threshold defined in the policy.

Apart from these fundamental features, there are additional features which are desirable for a certificate exchange protocol.

Univocal interpretation of policies and credentials. The central role played by policies and credentials requires CAs to publish and sign the encoding algorithms used in such a way to make a further format negotiation simple.

Policy Language. A good language to express policies efficiently must be available. Moreover, it must enable its extension for uses outside the contest where it was originally designed.

Communication Strategies. The protocol must allow the use of different strategies, each one persecuting different aims. We can consider optimal the situation where different strategies could change at any time, even while being executed.

Efficiency. The efficiency of a protocol can be evaluated on the basis of the size and number of exchanged messages, in the case of a certain request. Apart from network traffic, also computational capacity required to execute the protocol should be taken into account. For example, the use of cryptographic systems produces an overhead not compatible with an embedded or otherwise limited system.

5. CONCLUSIONS

In this paper we discussed the notable idea of automated trust negotiation and we started the process of integrating it into multi-agent systems. The inherent openness and scalability of multi-agent systems inhibit the simple adoption of available models of automated trust negotiation and a step forward urges. This work identifies the basic features of a novel protocol for allowing agents negotiating trust and it also shows the basis of its integration in multi-agent systems. Unfortunately, for editorial reasons, the presentation of a prototype Java framework that implements the presented ideas lacks and it is reserved for a future paper. Briefly, such a framework allows:

1. Creating credentials, containing one or more obscured attributes associated with their subject, potentially issued by a third entity;
2. Releasing, at the same time or separately, a signature to attest the authenticity and the credential itself;
3. Verifying that credentials were not altered;
4. Evaluating an access request, verifying the possession of some attributes;
5. Using one of the implemented protocols to verify a pre-requisite.

Therefore, the framework implements a drop-in solution to many, if not all, problems in managing trust negotiation in real-world multi-agent systems and it has been developed using the criteria and the desiderata that this work structured and motivated.

6. REFERENCES

[1] Blaze, M., Feigenbaum, J., and Lacy, J. 1996 "Decentralized trust management". In Procs. 17th Symposium on Security and Privacy. pp. 164-173, IEEE Computer Society Press.

[2] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A. 1999. "The KeyNote Trust-Management System Version 2." RFC 2704.

[3] Bonatti, P. A., and Samarati, P. 2002. A uniform framework for regulating service access and information release on the web. Journal of Computer Security, vol. 10(3):241–272.

[4] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T. 1999. SPKI certificate theory. IETF RFC 2693.

[5] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.

[6] Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons K., and Winslett, M. 2004. No Registrations Needed: How to Use Declarative Policies and Negotiation to Access Sensitive Resources on the Semantic Web. In Procs. European Semantic Web Symposium. Heraklion, Greece.

[7] Housley, R. et al. 2002. Internet X.509 PKI Certificate and CRL Profile. IETF RFC 3280. http://www.ietf.org/rfc/rfc3280.txt

[8] Li, J., and Li, N. 2006. OAcerts: Oblivious attribute certificates. IEEE Trans. Dependable Sec. Comput., vol. 3(4):340–352.

[9] Li, N., Mitchell, J. C., and Winsborough, W. H. 2002. Design of a role-based trust management framework. In IEEE Symposium on Security and Privacy, pp. 114–130.

[10] Rivest, R. L., Lampson, B. "SDSI - A Simple Distributed Security Infrastructure". http://people.csail.mit.edu/rivest/sdsi11.html

[11] Seamons, K. E., Winslett, M., and Yu, T. 2001. Limiting the disclosure of access control policies during automated trust negotiation. The Internet Society. ISBN 1-891562-11-8.

[12] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. Acm Press Frontier Series. ACM Press, New York, NY, 19-33. DOI= http://doi.acm.org/10.1145/90417.90738

[13] Winsborough, W. H. 2002. Towards practical automated trust negotiation. In Procs 3rd International Workshop on Policies for Distributed Systems and Networks. pp. 92–103. IEEE Computer Society Press.

[14] Winsborough, W. H., and Li, N. 2002. Protecting sensitive attributes in automated trust negotiation. In WPES, pp. 41–51. ACM. ISBN 1-58113-633-1.

[15] Winsborough, W. H., and Li, N. 2006. Safety in automated trust negotiation. ACM Trans. Inf. Syst. Secur., vol. 9(3):352–390.

[16] Winslett, M., Ching, N., Jones, V. E. and Slepchin, I. 1997. Using digital credentials on the world wide web. Journal of Computer Security, vol. 5(3):255–266.

[17] Yu, T., Ma, X., and Winslett, M. 2000. Prunes: an efficient and complete strategy for automated trust negotiation over the internet. In Procs 7th ACM Conference on Computer and Communications Security, pp. 210–219. ACM. ISBN 1-58113-203-4.