CERIAS Tech Report 2004-68

TRUST NEGOTIATION: CONCEPTS, SYSTEMS AND LANGUAGES

by E. Bertino, E. Ferrari, A.C. Squicciarini

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

TRUST NEGOTIATIONS: CONCEPTS, SYSTEMS, AND LANGUAGES

Trust negotiation is a promising approach for establishing trust in open systems such as the Internet, where sensitive interactions sometimes occur among entities with no prior knowledge of each other. The authors provide a model for trust negotiation systems and delineate the features of ideal trust negotiation systems.

During the last decade, data and service interchanges throughout the Internet became not only possible but essential. Nowadays, interactions involving entities such as businesses, military and scientific partners, and companies and their cooperating partners or customers are becoming everyday occurrences. In such a scenario, traditional assumptions for establishing and enforcing access control regulations no longer hold. The entities need to authenticate and trust each other to exchange sensitive information and resources. Interactions are further complicated because the interacting entities usually belong to different security domains or don't have preexisting relationships.

To address such issues, researchers have proposed trust management as a new approach for protecting open, decentralized systems, in contrast to traditional tools for securing conventional systems. Matt Blaze and his colleagues first coined the term *trust management*[1] to denote a distinct component of security in network services. Trust establishment begins in open systems with the identity-based access control mechanisms usually adopted in conventional systems[2] (such as databases and operating systems). According to such a paradigm, each subject is uniquely identified by an ID (for example, a login name or IP address)—that is, the means for proving its trustworthiness. However, in an open environment, identity is not always relevant when determining whether to trust a subject, but other properties are crucial in determining parties' trustworthiness.

An emerging approach that uses an entity's properties to establish trust is *trust negotiation* (TN).[3] A TN consists of iteratively disclosing certified digital credentials. These credentials verify properties of their holders to establish mutual trust. Thus, TN deals with concepts such as formulating security policies and credentials, determining whether particular sets of credentials satisfy the relevant policies, and deferring trust to third parties.

This article discusses TN systems by first describing their basic elements and then identifying the features of ideal examples. In defining TN requirements, we consider both language and system requirements. After surveying the most interesting proposals researchers have pre-

1521-9615/04/$20.00 © 2004 IEEE
Copublished by the IEEE CS and the AIP

ELISA BERTINO
Purdue University
ELENA FERRARI
University of Insubria at Como, Italy
ANNA SQUICCIARINI
University of Milan, Italy

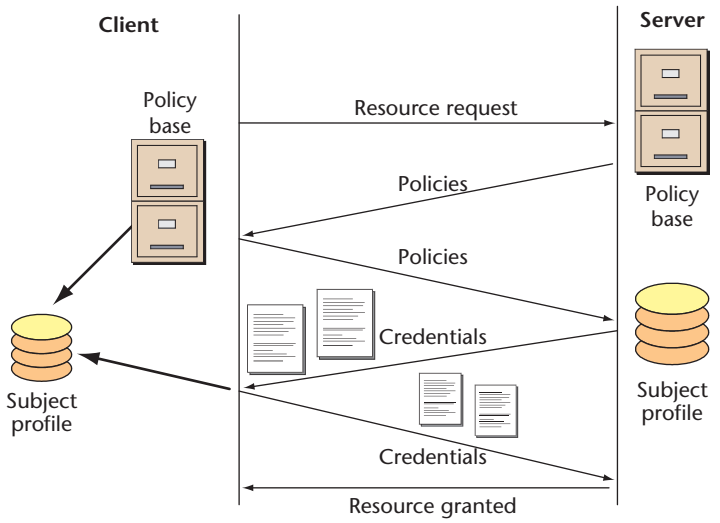

Figure 1. A trust negotiation (TN) process. In the example, negotiating participants perform a TN in which the client obtains a service after exchanging policies and credentials with the server.

sented so far, we also outline the field's future research directions.

Basic Concepts

A TN consists of a bilateral disclosure of digital credentials; it represents statements certified by given entities who verify the properties of their holders. Trust is thus incrementally built by iteratively disclosing digital credentials according to ad hoc resources—namely, disclosure policies.

Building Blocks

A TN involves a *client*, or the entity asking for a certain resource, and a *server*, the entity owning (or, more generally, managing access to) the requested resource. In spite of these names, which can be misleading, the model is basically peer to peer; both entities can possess sensitive resources they must protect, so we must equip them with a compliant negotiation system. A *resource* comprises sensitive information and services, whereas an *entity* includes users, processes, roles, and servers. The term resource is intentionally generic to emphasize that the negotiations we refer to are general purpose—that is, a resource is any sensitive object (such as financial information, health records, or credit-card numbers) that has a set of policies protecting its disclosure.

Figure 1 sketches a typical negotiation process. During negotiation, entities incrementally establish trust by iteratively disclosing digital credentials to verify properties of the negotiating parties. Credentials are typically collected by each party in appropriate repositories, also called profiles. Another key component of any TN is a set of access control policies, known as *disclosure policies*, which govern access to protected resources by specifying credential combinations that entities must submit to obtain authorization.

To carry out a TN, parties usually adopt a strategy implemented by an algorithm that defines which credentials to disclose, when to disclose them, and whether to succeed or fail the negotiation. Numerous strategies exist for negotiating trust, each with different properties with respect to speed of negotiations and caution in releasing credentials and policies. The algorithms record the progress of negotiation strategies in ad hoc data structures, typically negotiation trees[4] or graphs, on which performance and TN algorithms' computational effort can be evaluated.[5] A strategy's efficiency depends on its communication and computational costs: communication cost includes the size and number of messages exchanged. A negotiation's communication and computational costs strictly depend on the adopted strategy and vary from exponential, in the case of a brute-force strategy, to more efficient strategies.

Digital Credentials

Digital credentials are assertions describing one or more properties about a given subject, referred to as the owner, certified by trusted third parties. Thus, a set of digital credentials identifies and describes entities; trusted third parties are *Certification Authorities* (CAs). Digital credentials are often compared to the paper credentials we carry in our wallets. Both contain properties about the owner and must be unforgeable and verifiable. To ensure such properties, credentials are digitally signed using PKI.[2] Typically, a digital credential contains a set of properties specified via name–value pairs that are signed by the issuer's private key and can be verified by using the issuer's public key. Although some proposals exist for encoding digital credentials,[6] until now, no widely accepted standard exists for their representation.

The X.509 V3 standard[2] for public-key certificates takes a step in this direction. Even though the original intent of X.509 certificates was simply to bind a key to a name, version V3 adds an extensibility mechanism to the original X.509 certificate format. The extensions include fields such as additional subject identification information, key attribute information, policy information, and certification path constraints. However, because the X.509 certificate was not conceived for online ne-

gotiations, it does not properly support attributes or protect privacy. As a result, researchers have recently proposed other formats that can better support an entity's property description or that can achieve privacy and can't be forged.[7]

Disclosure Policies

Disclosure policies state the *conditions* under which a party can release a resource during a negotiation. (Conditions are constraints against the interacting parties' credentials and their properties.) Depending on their content, credentials might be sensitive—for example, a credential might contain private attributes about an individual such as a credit-card number. Because of digital credentials' sensitive nature, their disclosure must be carefully managed according to policies that specify the conditions under which parties can disclose them.

We also regard disclosure policies as sensitive information because they are often related to organizations' business and governance processes. Therefore, recent research considers disclosure policies as sensitive as other resources.[4,8,9] Clearly, the presence of sensitive disclosure policies will add new requirements to TN processes. Entities must gradually establish trust, and policies for the involved resources must be sent to the other party according to the level of trust established.

TN Requirements

We consider some dimensions more relevant for evaluating policy languages than others. We have classified the dimensions into two main groups: those related to the adopted language and those related to the system and its components. The requirements we have devised are a partial list; other requirements will likely surface as research and deployment of negotiation systems progress, especially given the increasing number of researchers actively contributing to this area.[10]

Language Requirements

TN policy languages[4,11] are a set of syntactic constructs (for example, credentials and policies) and their associated semantics that encode security information to be exchanged during negotiations. Good TN languages should thus be able to simplify credential specification and express a range of protection requirements through specification of flexible disclosure policies. The dimensions we have identified to reach these goals deal with language expressiveness and semantics.

Well-defined semantics. A well-defined policy language should have a simple, compact, formally defined semantics, remaining independent of the language's particular implementation. We might effectively express the semantics using various formalisms such as logic programs or relational algebra.

Monotonicity. The monotonicity requirement specifies that once a set of credentials allowing the disclosure of a certain resource is found, the disclosure of additional credentials and policies should only result in the grant of additional resources, if possible. This aspect implies that the parties must carefully handle negation: if a policy requires that a subject must not have a given property to obtain a resource, then it isn't enough that the subject simply fails to disclose the corresponding credential. Rather, verifying such negative conditions should be carried out by directly checking the properties that the credential holder possesses with the credential issuer authority. Checking the absence of a credential can be managed at the policy level as long as the policy owner has the capability to perform such a check.

Credential combination. A set of different credentials might likely describe the set of properties characterizing a given subject. Thus, a policy language should be expressive enough to require submission of a combination of credentials, using conjunction and disjunction operators.

Authentication. Each party can have multiple identities stated by different credentials issued and signed with different public keys to prevent collusion. At runtime, the credential submitter (that is, the CA or a delegated entity by means of credential chains) thus will have to demonstrate the knowledge of the private key associated with the public key used to sign the credential.

Constraints on property values. Each credential is usually a structured object conveying information about a subject's properties. A name–value pair typically represents each property. Credentials can be associated with a given credential type, thus simplifying credential specification and management. A policy language should include constructs to constrain the requested credentials to have a certain type and restrict valid property values. For example, a rental car agency might ask users to submit a driver's license and, further, that the driver's license be issued after a given date.

Intercredential constraints. To better evaluate remote party properties, policies might express constraints to compare values of different credentials belonging to the same subject, even if they use different keys.

Sensitive policy protection. By analyzing policies' content, outsiders might infer sensitive information about the parties. Thus, disclosure policies must be protected in the same way as other resources, with a fine-grained control over their disclosure. We can handle policy protection at the language or system level. In the first case, the policy language must have constructs to express constraints on policy disclosure, whereas in the second case, the runtime system must check disclosure policies and dynamically add constraints instead of disclose them unconditionally.

Unified formalism and use of interoperable languages. These requirements focus on the applicability of negotiation approaches. We believe that in designing negotiation languages, it's essential to focus on solutions that can be effectively adopted in real environments and easily integrated in existing contexts. The first requirement deals with uniformly protecting credentials and policies, thus simplifying protection mechanisms. The latter requirement facilitates transmission and interoperability among negotiation participants. In this respect, using metalanguages such as XML[12] might facilitate submitting and exchanging credentials and policies.

System Requirements

A negotiation system supports a TN. It usually consists of several modules and a runtime system with related algorithms supporting all the protocols underlying a negotiation.

The challenging aspect in developing such systems is to devise solutions that trade off the requirements that often conflict with each other. On the one hand, such systems should be flexible, scalable, and portable. On the other, they should support advanced functions, such as support for credential chains, authentication of multiple identities, and complex compliance-checker modes whose efficient implementation is often difficult. In particular, the compliance checker must interpret a remote policy and check whether a set of local credentials exists that satisfies the received policy. Our requirements at the system level are as follows.

Credential ownership. During a negotiation, when a remote credential is received, the runtime system must challenge the sender to prove the ownership of the private key associated with the public key used to identify the subject in the credential. The system can use various security protocols for such a task, but a key issue is to integrate the negotiation framework with the existing tools and systems, maximizing security controls over the exchanged data.

Credential validity. The validity of the exchanged credentials is fundamental to ensuring the whole negotiation functions correctly. Thus, each time a credential is received, the credential content's integrity must be verified by using a digital signature to guard against forgery. Furthermore, the runtime system must always check for expired or revoked credentials.

Credential chain discovery. The credentials needed during a negotiation might not be readily available locally. A runtime system should include extra machinery and tools for credential chain discovery to retrieve in real time credentials that are not locally cached.

Privacy protection mechanisms. Disclosing policies and resources should ensure a good protection of the parties' privacy goals, which typically entail disclosing the minimum set of information necessary to succeed in the process. Intuitively, unconditionally disclosing policies and credentials might leak sensitive information. Complementary mechanisms should be integrated with the system to address the parties' privacy requirements.

Support for alternative negotiation strategies. The negotiation system should support various negotiation strategies—for instance, by maximizing information protection or considering first the computational effort required. A well-designed system should provide numerous strategies, leaving the parties free to choose one. The strategies might be chosen independently by the parties or negotiated just like the other resources. In the first case, it's essential to ensure negotiations function correctly, even if parties do not adopt the same strategy.

Fast negotiation strategies. We can expect that in many scenarios there will be standard, off-the-shelf policies available for widely used resources (such as Visa cards and passports). Thus, it is likely that in negotiations involving such common resources, the same sequences of credentials will be used several times to perform

similar negotiations. In such cases, it might be useful to keep track of the sequences of credentials exchanged more often instead of recalculating them for each negotiation. The strategies thus should include approaches to let the parties establish trust by using precomputed sequences, if desired. Additionally, once two parties have successfully negotiated, the system might exploit such a negotiation to speed up subsequent negotiations. Finally, when commonly used resources are involved, an ideal system should automatically select and suggest the policies to be exchanged, even when parties are total strangers.

Systems and Prototypes

Because of the relevance of TN for Web-based applications, researchers have recently developed several systems and research prototypes.[4,8,11] We have surveyed and analyzed these based on the requirements we just discussed. Our results show that several design goals still remain unsatisfied.

Analyzing Existing Negotiation Systems

Until now, the best-known trust-management system was Keynote.[13] Keynote is designed to work for various large- and small-scale Internet-based applications. It provides a single, unified language for both local policies and credentials. Keynote credentials, called assertions, contain predicates that describe the trusted actions permitted by the holders of a specific public key. As a result, Keynote policies do not handle credentials as a means of establishing trust, mainly because the language was intended for delegation authority. Therefore, it has several shortcomings with respect to TNs.

The Trust Establishment (TE) Project at Haifa Research Lab has developed a tool for enabling trust relationships between strangers based on public-key certificates. A key element of the system is a Trust Policy Language (TPL),[11] specified using XML.[12] A distinctive feature of the system is that it extends traditional role-based access control systems by validating certificates and then mapping the certificates' owners to a role. A policy, specified by the resource's owner, states the rules for mapping entities to roles. A *role* in TPL is a group of entities that represent a specific organizational unit (for example, managers or doctors). Each role has one or more rules defining how a certificate owner can become a role member. The TE system also includes an intelligent certificate collector that automatically collects missing certificates from certificate repositories, allowing the use of standard browsers that can only pass one certificate to the server. However, it does not support sensitive credentials. One of the TE's basic assumptions is that credentials can be disclosed whenever they are requested. Furthermore, the TE system does not address sensitive policies.

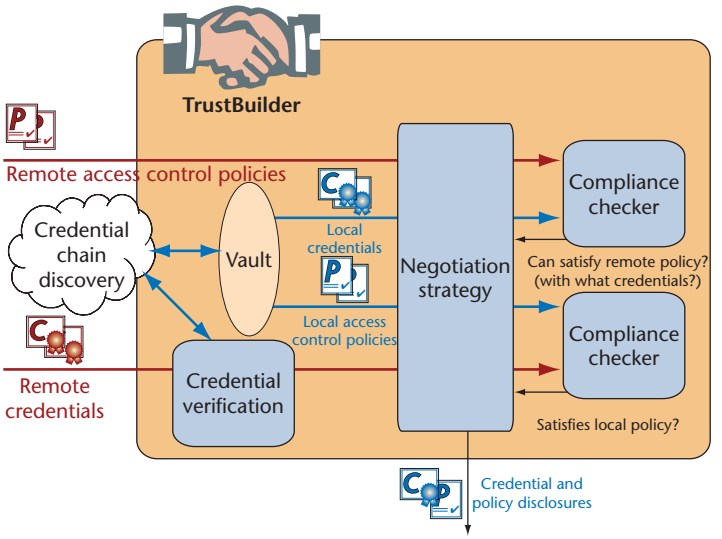

Figure 2. A negotiation using TrustBuilder.[14] This diagram illustrates how Alice's agent verifies remote credentials, demonstrates Alice's ownership of her credentials, checks to see if Alice's credentials satisfy Bob's disclosed access control policies, checks to see if Bob's credentials satisfy Alice's access control policies, and determines what policies and credentials to disclose to Bob. Bob's TrustBuilder security agent provides corresponding functionalities.

The Internet Security Research Lab (ISRL)[14] at Brigham Young University is an active research center in trust management. Researchers of ISRL have developed the TrustBuilder system to support TN. TrustBuilder currently represents one of the most significant proposals in the negotiation research area. It provides a broad class of negotiation strategies, as well as a strategy- and language-independent negotiation protocol that ensures the interoperability of the defined strategies within the TrustBuilder architecture. Each participant in a negotiation has an associated security agent that manages the negotiation. During a negotiation, the security agent uses a local negotiation strategy to determine which local resources to disclose next and to accept new disclosures from other parties. The TrustBuilder architecture includes a credential verification module, a policy compliance checker, and a negotiation strategy module, which is the system's core. During a negotiation, each agent adopts a local strategy to determine which local resources to disclose and whether to terminate the negotiation. The system also relies on a credential verification module, which performs a va-

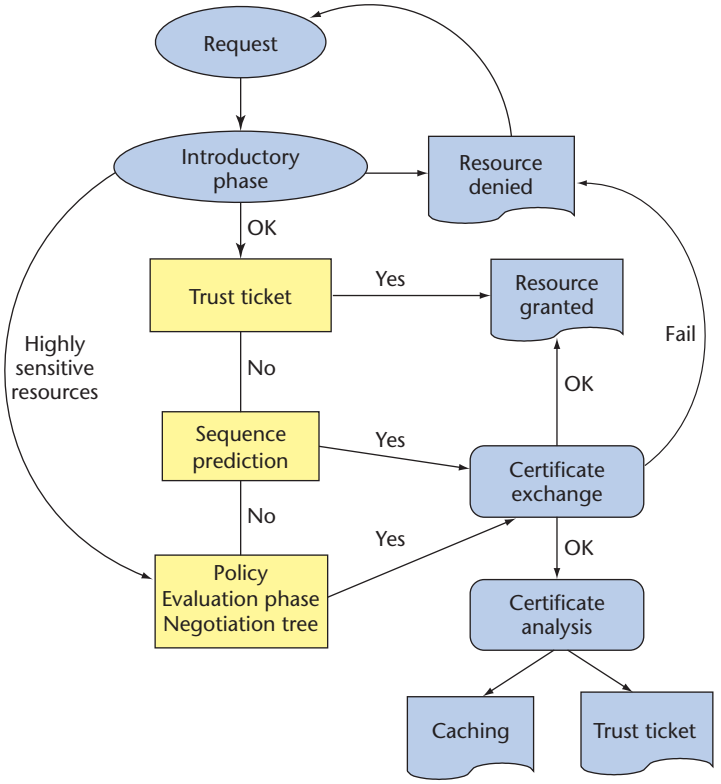

Figure 3. Phases of a Trust-X negotiation. A Trust-X negotiation can follow different approaches, including executing a traditional policy evaluation phase, exchanging trust tickets, or using a predefined trust sequence of credentials.

lidity check of the received credentials. Figure 2 presents an example of a TrustBuilder negotiation.

Recently, researchers at ISRL have explored other issues in the negotiation area, such as supporting sensitive policies (obtained by introducing hierarchies in policy definitions),[8] and privacy protection mechanisms (obtained by introducing dynamic policies, the policies dynamically modified during a negotiation) into the TN architecture.[15]

Unipro[9] is a unified scheme for modeling resource protection, including policies in TN. Unipro models policies as a first-class resource and provides a fine-grained control over policy disclosure, giving subjects more flexibility in expressing authorization requirements.

Another interesting contribution in the TN area is given by the role-based trust-management (RT) framework, also from ISRL. The RT framework is part of an ongoing project to address security problems in decentralized environments. It provides a policy language with a well-defined semantics, a deduction engine, and the use of application-domain-specification documents that help distributed subjects maintain a consistent use of policy terms. The RT language exploits new approaches to protect sensitive attributes by introducing the notion of attribute acknowledgment policies, which participants establish in association with attributes that they consider sensitive, regardless of whether or not they satisfy those attributes. The authors also have defined a *trust target graph* protocol to support attribute-based access control (ABAC) systems. The protocol is similar to the Disclosure Tree Protocol used in TrustBuilder, but it supports a realistic ABAC language, whereas the Disclosure Tree Protocol supports only logic-based languages and does not protect against unauthorized disclosure. The deduction engine is similar to TE, but it's more powerful. Compared to TE, RT's advantages include a declarative logic-based semantic foundation, strongly typed credentials and policies, and more flexible delegation structures.

Finally, Trust-X[16] is a framework for TNs that supports all aspects of negotiation and was specifically conceived for a peer-to-peer environment. The first component of Trust-X is an XML-based language, called X-TNL, for specifying Trust-X certificates and policies. Like the RT language, Trust-X has a typing credential system and addresses the issues of vocabulary agreement using XML namespaces.[12] Using namespaces combined with the certificate type system helps TN software correctly interpret different credentials' schema, even if they are issued by different entities that do not share a common ontology. Trust-X certificates are either credentials or declarations. (A credential states its owner's personal characteristics, whereas declarations collect personal information that does not need to be certified but that might help in better customizing the offered service.)

A novel aspect of X-TNL is its support for special certificates, called *trust tickets*. Trust tickets are issued on successfully completing a negotiation and can speed up subsequent negotiations for the same resource. Additionally, X-TNL provides a flexible language for specifying policies and a mechanism for policy protection, based on the notion of policy preconditions. A Trust-X negotiation consists of a set of phases that are sequentially executed according to the flow Figure 3 illustrates. As the figure shows, Trust-X enforces a strict separation between policy exchange and resource disclosure. This distinction results in an effective protection of all the resources involved in negotiations.

Trust-X is a flexible system, providing various

Table 1. Language comparison.

Requirements	RT	TPL	Trust-X	Keynote	TrustBuilder
Well-defined semantics	Y	Y	Y	Y	Y
Monotonicity	Y	N	Y	Y	Y
Credential combinations	Y	Y	Y	Y	Y
Constraints on property values	Y	Y	Y	N	Y
Intercredential constraints	Y	Y	Y	N	Y
Credential chains	Y	Y	P	N	N
Authentication	Y	N	N	N	N
Sensitive policies	Y	N	Y	N	Y
Unified formalism	N	Y	Y	Y	N
Interoperable languages	N	Y	Y	N	N

(Key: Y means Yes, N means No, and P means Partial support)

Table 2. System comparison.

Requirements	RT	TPL	Trust-X	Keynote	TrustBuilder
Credential validity	Y	Y	Y	N	Y
Credential ownership	N	N	P	N	N
Support for alternative negotiation strategies	Y	N	Y	N	Y
Fast negotiation strategies	N	Y	Y	N	N
Privacy protection mechanisms	Y	Y	Y	Y	Y
Credential chain discovery	Y	Y	P	N	N

(Key: Y means Yes, N means No, and P means Partial support)

TN strategies that allow better trade-offs between efficiency and protection requirements. In particular, Trust-X supports three different negotiation modes. The first, based on trust tickets, can be adopted when the parties have already successfully completed a negotiation for the same resource. The second mode, based on using specific abstract data structures called *negotiation trees*, performs a runtime evaluation of the negotiation's feasibility by determining a sequence of certificate disclosures that can successfully end the negotiation. The last mode exploits a notion of similarity between negotiations and is based on the observation that a service provider usually handles many similar negotiations. Finally, the system includes an architecture for negotiation management.

Comparing the Systems

To better assess the current state of the art, we compared the systems we've described here on the basis of the requirements we listed earlier. Tables 1 and 2 summarize our results. All the considered languages have a well-defined semantics, and all but TPL are monotonic. However, because of the difficulties in efficiently implementing nonmonotonic languages, the actual TPL implementation is restricted to a monotonic version, known as DTPL.

None of the proposals are complete, even though current systems address significant subsets of relevant requirements. Most of the TN systems currently available are based on some unrealistic assumptions that limit the approach's applicability. For instance, those systems usually assume that all credentials associated with a party are at the party site. However, in many application environments, credential storage is not centralized. Thus, TN systems should include credential retrieval mechanisms to be used during negotiations. (The only system addressing this requirement is from Brigham Young University's ISRL; see http://isrl.cs.byu.du/.)

Furthermore, none of the existing systems addresses how to obtain credentials, assuming that the entity disclosing credentials has full responsibility for obtaining and caching them locally. Also, none provide real protection against attacks on negotiating parties, such as credential or identity theft.[2]

Finally, another area of investigation is related to the interoperability among different TN languages. The lack of centralized control over the In-

ternet makes it unrealistic to assume that all "strangers" will adopt the same system.

In addition to the open research directions we have already discussed, several key issues must be investigated to make TN systems usable and reliable. A first issue deals with developing a solid underpinning theory for trust languages and algorithms. The languages should be powerful enough to express a range of protection requirements and should support credential chains and authentication of the submitter. The systems should be able to properly carry out negotiations, trading off between computational effort and requirements such as privacy protection. The systems should also be strong enough to limit damages caused by intruder attacks or interception.

Another important issue is in developing efficient strategies for policy enforcement. In this respect, policy enforcement is a constrained problem because policies usually are specified as constraints on credential properties. As such, it is important to explore the possibility of expressing constrained entailment techniques to the TN environment and to investigate issues such as constrained consistency and redundancy in this context. An additional issue is using data-mining techniques to analyze data collected on negotiation processes with the aim of improving the efficiency of the negotiation strategies. For example, by using association rule discovery techniques, we might determine whether the majority of users own credentials in combination. In this case, the negotiation strategies we might prefer would use such credential combinations instead of other possible combinations.

Several other interesting issues are related to scalability and autonomy. Thus, we believe that even though current TN systems are comprehensive in terms of the functions they support, a strong need exists for new research in this area to lead the development of next-generation TN systems. CiSE

References

1. M. Blaze and J. Feigenbaum, "Decentralized Trust Management," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 1996, pp. 164–173.
2. *Proc. Trust Management, 1st Int'l Conf.* (iTrust 2003), LNCS, vol. 1, Springer, 2003.
3. K. Seamons et al., "Protecting Privacy During On Line Trust Negotiation," *Proc. 2nd Workshop on Privacy Enhancing Technologies*, LNCS, Springer, 2002, pp. 129–143.
4. E. Bertino, E. Ferrari, and A. Squicciarini, "Trust-X: A Peer to Peer Framework for Trust Negotiations," to appear in *IEEE Trans. Knowledge and Data Eng.*, IEEE CS Press, June 2004.
5. W. Stallings, *Cryptography and Network Security: Principles and Practice*, 2nd ed., Prentice Hall, 1999.
6. T. Bray, D. Hollander, and A. Layman, *Namespaces in XML*, W3C Recommendation, Jan. 1999.
7. S. Brands, *Rethinking Public Key Infrastructure and Digital Credentials*, MIT Press, 2000.
8. K. Seamons et al., "Requirements for Policy Languages for Trust Negotiation," *Proc. 3rd IEEE Int'l Workshop on Policies for Distributed Systems and Networks*, IEEE CS Press, 2002, pp. 68–79.
9. T. Yu, X. Ma, and M. Winslett, "PRUNES: An Efficient and Complete Strategy for Automated Trust Negotiation over the Internet," *Proc. 7th ACM Conf. Computer and Communication Security*, ACM Press, 2000, pp. 210–219.
10. K. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation," *Proc. Network and Distributed System Security Symp.* (NDSS 01), 2001.
11. A. Herzberg and Y. Mass, "Relying Party Credentials Framework," *Proc. RSA Conf.*, 2001, pp. 23–39.
12. T. Yu and M. Winslett, "A Unified Scheme for Resource Protection in Automated Trust Negotiation," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 2003, pp. 110–122.
13. M. Blaze et al., *The KeyNote Trust-Management System*, RFC 2704, Sept. 1999.
14. A. Herzberg et al., "Access Control System Meets Public Infrastructure, Or: Assigning Roles to Strangers," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 2000, pp. 2–14.
15. N. Li, J.C. Mitchell, and W.H. Winsborough, "Design of a Role-Based Trust Management Framework," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 2002, pp. 114–130.
16. E. Bertino, E. Ferrari, and A. Squicciarini, "X-TNL: An XML Based Language for Trust Negotiations," *Proc. 4th IEEE Int'l Workshop on Policies for Distributed Systems and Networks*, IEEE CS Press, 2003, pp. 81–84.

Elisa Bertino is a professor of computer science and the research director of the Center for Research Education in Information Assurance and Security (CERIAS) at Purdue University. Her research interests are in security, privacy, database systems, multimedia systems, and object-oriented technology. She is a fellow of the ACM and IEEE. She received the IEEE Computer Society Technical Achievement Award in 2002. Contact her at bertino@cerias.purdue.edu.

Elena Ferrari is a professor of database systems at the University of Insubria at Como, Italy. Her research interests include database and Web security and temporal and multimedia databases. She has a PhD in computer science from the University of Milan. She is on the editorial board of the *VLDB Journal* and the *International Journal of Information Technology* (IJIT). She is a member of the ACM and IEEE Computer Society. Contact her at elena.ferrari@uninsubria.it.

Anna Cinzia Squicciarini is a PhD student at the University of Milan, Italy. Her research interests include trust negotiations, privacy, models and mechanisms for privilege and contract management in virtual organizations, and Web services access control models. She has a degree in computer science from the University of Milan. Contact her at squiccia@dico.unimi.it.