



Negotiating Trust on the Web

To enable interactions across security domain boundaries, the TrustBuilder trust negotiation system establishes trust between strangers by gradually disclosing credentials.

**Marianne Winslett
and Ting Yu**
University of Illinois at
Urbana-Champaign

**Kent E. Seamons,
Adam Hess, Jared Jacobson,
Ryan Jarvis, Bryan Smith,
and Lina Yu**
Brigham Young University

Increased connectivity and data availability enable new ways of conducting business, but they also create new security vulnerabilities. For example, to streamline a financial transaction, an organization might want to give certain strangers – that is, parties from outside its security domain – access to some of its local resources. Before doing so, however, the organization must establish firm policies outlining the types of strangers who can access the resources, as well the types of data and services the organization will make available to them. Traditional access-control policies describe access conditions in terms that only apply to parties within the local security domain. Clearly, new kinds of access-control policies are needed.

Trust negotiation can allow strangers to access sensitive data and services on the Internet.^{1,2} Trust negotiation is the iterative disclosure of credentials and requests for credentials between two parties, with the goal of establishing sufficient trust so that the parties can complete a transaction. Trust negotiation should be ubiqui-

tous: available anytime, anywhere, at all layers of software, wherever strangers might wish to interact, including mobile devices and intelligent environments. Traditional approaches to establishing trust either minimize security measures (for example, they do not verify credentials) or assume that the parties are not strangers and can present a local identity (login, capability, or credential) to obtain service. Trust management systems such as PolicyMaker,³ KeyNote,⁴ simple public key infrastructure/simple distributed security infrastructure (SPKI/SDSI),⁵ and Delegation Logic⁶ support delegation of authority, but are not helpful for establishing trust between strangers using general-purpose credentials.

Our system, TrustBuilder, supports automated trust negotiation between strangers on the Internet. TrustBuilder lets negotiating parties disclose relevant digital credentials and access-control policies and establish the trust necessary to complete their interaction (see the sidebar, “TrustBuilder in Action,” for an example scenario). TrustBuilder is intend-

TrustBuilder in Action

Figure A shows a trust negotiation between Julie, a general contractor, and RainhandleR (an actual company, although the details presented here are fictitious). Julie wants to buy a rain dispersal system from RainhandleR over the Web. She fills out the online order form, checking a box to indicate her exemption from sales tax. On receiving the order, RainhandleR will want to see Julie's credentials—a valid credit card and a current reseller's license. Although Julie is willing to show her reseller's license to anyone, she will only show her credit card to members of the Better Business Bureau. To establish enough trust to complete this transaction, RainhandleR can disclose its BBBOnline membership credential to Julie. Julie can then disclose her credit card and reseller's license to RainhandleR, and purchase the system.

Also disclosed during negotiation are supporting credentials owned by the issuers of Julie's and RainhandleR's credentials. Julie protects her Visa card credential with a policy that requires RainhandleR to disclose its BBBOnline credential. Julie adopts TrustBuilder's Simple negotiation strategy, which discloses credentials as soon as their policies have been satisfied, helping to establish trust quickly. RainhandleR discloses credentials using TrustBuilder's Relevant strategy (see Figure 1 in the main article).

If RainhandleR were also to adopt TrustBuilder's Simple strategy, its first message would disclose its BBBOnline credential and the policy governing the tax exemption service. Julie would disclose both her reseller's license and credit card in the next message, and the negotiation would conclude successfully.

Figure A elides many details of trust negotiation. For example, practical policies specify constraints regarding expiration, revocation, and attribute values. Further, the negotiation messages must agree on a protocol and set of negotiation strategies, establish a

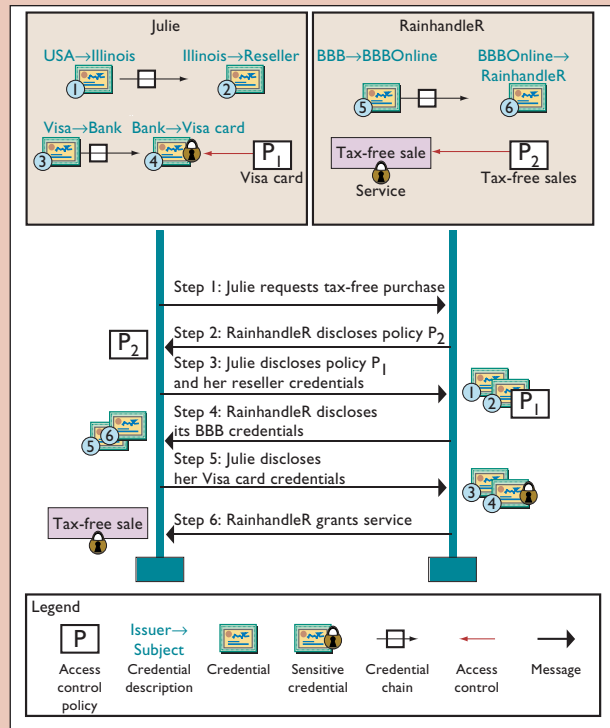


Figure A. An example of trust negotiation. Julie and RainhandleR disclose their credentials in a series of steps controlled by their access control policies.

confidential session for the negotiation, and answer challenges to prove credential ownership.

ed for use in any situation where two entities from different security domains need to establish trust – business-to-business and retail interactions, cooperative work and joint ventures, medical records, mobile computing, and so on.

Negotiating Trust

Strangers generally do not establish trust solely on each other's global identity (social security number, institutional tax ID, and so on). They usually base trust on each other's *attributes*, such as place of employment, citizenship, age, and organizational memberships. *Digital credentials*, the online analogues of paper credentials (a driver's license, passport, or employee ID card, for example) can describe these attributes, usually expressed as name/value pairs. A digital credential is a digitally signed assertion by a credential issuer about the credential owner. It is signed using the issuer's private key and verified using the issuer's public key. A credential often includes the public keys of the entities it

describes, so the entities can use their corresponding private keys to answer challenges or otherwise prove that the credential refers to them. Other approaches are also possible.⁷ Digital credentials can be implemented using, for example, X.509v3 certificates or signed XML statements.

To automate trust negotiation, each party must establish *access control policies* to protect its sensitive resources, including credentials and services, from inappropriate access. Each policy should specify the digital credentials strangers must present to access the protected resource. Policies can themselves be sensitive resources. For example, the Web site for a secret joint venture of three companies might be protected by a policy that restricts access to employees of the appropriate departments in those companies. An outsider who sees the policy could guess that the three companies are engaged in a secret project and gain a competitive advantage.

Although trust negotiation can fill an important need on the Internet, many questions remain. For

Credential Chain Discovery

We can combine credentials to form chains, in which the subject of one credential is the issuer of the next. Bob's policies describe what issuers he trusts for what purposes. To satisfy Bob's policies, Alice must send him credential chains that lead to issuers he trusts.

Suppose Visa issues credentials to banks, authorizing them to issue credit cards to customers. An online merchant need not know all banks that issue Visa credit cards, as the merchant's policies can require customers to submit their credit card along with their card issuer's certification from Visa. Users store some credentials; issuers store others. When a security agent gathers a chain of credentials during negotiation, it can do so using a forward, backward, or hybrid approach.

The IBM Trust Establishment system's collector searches for credentials in a bottom-up fashion.¹ The Query Certificate Manager (QCM) system and Secure Dynamically Distributed Data-log (SD3) support distributed credential chain discovery through

backward searching.^{2,3} RT₀, a role-based trust-management language, provides a type system for credential storage that supports forward, backward, and hybrid algorithms for credential chain discovery.⁴

References

1. A. Herzberg et al., "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., May 2000, pp. 2-14.
2. C. Gunter and T. Jim, "Policy-Directed Certificate Retrieval," *Software: Practice and Experience*, vol. 30, no. 15, Sept. 2000, pp. 1609-1640.
3. T. Jim, "SD3: A Trust Management System with Certified Evaluation," *IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., May 2001, pp. 106-115.
4. N. Li, W.H. Winsborough, and J.C. Mitchell, "Distributed Credential Chain Discovery in Trust Management (Extended Abstract)," *Proc. 8th ACM Conf. Computer and Comm. Security*, ACM Press, New York, Nov. 2001, pp. 156-165.

example, how can developers use credentials and policies to implement trust negotiation in a scalable, reliable, private, secure, and highly available manner, and how can trust negotiation be integrated with all the protocols and standards already in use for accessing data over the Internet? How can trust negotiation facilities decide what credentials and other resources to disclose, and when? And how do we know that trust negotiation software will really operate "correctly"?

The TrustBuilder Approach

How can Alice learn what credentials she needs to access Bob's service? Alice and Bob could disclose their credentials and policies to a trusted third party, Cindy, and rely on her to determine whether they should trust each other. Unfortunately, such a third party could be an attractive target for attack, as well as a potential bottleneck for ubiquitous trust negotiation. We do not currently know how to implement trusted third parties for trust negotiation in a secure and scalable manner.

Alternatively, Alice and Bob could eschew third parties, and use zero-knowledge proofs to show that their credentials satisfy the relevant policies without ever disclosing the policies (or perhaps even the credentials). Currently, we do not know how to implement this approach efficiently.

The current implementations of TrustBuilder use a third approach, in which Alice and Bob negotiate trust directly by disclosing their credentials and/or policies to each other. In the simplest version of this approach, Alice discloses each of her credentials whose policy has been satisfied by Bob's disclosed credentials. Then Bob does the same, and

the process repeats. This shotgun approach can result in many unnecessary credential disclosures and needless rounds of negotiation when failure is inevitable, however. A better alternative, which considers need-to-know, is for Alice and Bob to disclose those policies that are relevant to the current negotiation. When policies contain sensitive information and cannot be disclosed freely, Alice and Bob can use a directed acyclic graph of policies to protect the information and allow gradual establishment of trust, as well as modular development of protection for sensitive resources.⁸

Credential Management

Where do credentials come from? To briefly summarize a complex process, credentials are issued by the same kinds of organizations that issue paper credentials today. To avoid attack, credentials are typically created offline and then either securely distributed to their new owners or made available for pickup in a semipublic database. Each issuer can use locally created identities to refer to the parties mentioned in its credentials, rather than a globally unique identity that would allow easy tracking of the parties' activities.

Who obtains the credentials needed to satisfy an access control policy? Busy servers might require clients to collect credentials when it is necessary and possible. Presumably, the clients will be sufficiently motivated to obtain the credentials needed to gain access to the desired service, beyond those already in their possession (see the sidebar "Credential Chain Discovery"). This model is too simplistic, however, and many credentials are too sensitive for public display. In some cases,

it is more efficient for servers to store and disclose their own credentials. In other cases, it is unrealistic for a server to expect a client to provide all the information needed to satisfy a policy. Consider, for example, a policy that says the client must provide a credit card that has not been revoked. The server should check for revocation, rather than rely on the client to submit a credit card revocation certificate.

Policy Development

An ideal policy language for trust negotiation should be able to specify each party's responsibility for satisfying the different parts of a particular policy.⁹ Suppose Alice wants to satisfy one of Bob's policies. Bob's copy of TrustBuilder can examine his policy to determine what information he must provide. Currently, Bob must obtain the credentials needed to satisfy the parts of his policy that are encapsulated in certain standardized function calls that are external to the policy language. For example, an external function can check for credit card revocation. Alice must provide all credentials needed to satisfy the remainder of the policy. If Bob's policy requires Alice to be a student at an accredited university, for example, Alice must present credentials that prove she is a student at a particular university, and that the university is accredited.

Policies must also be monotonic,³ in the sense that if a set of credentials disclosed by Alice satisfies Bob's policy, the policy should also be satisfied if Alice discloses additional credentials. For example, convicted felons may not purchase guns in the U.S. It would not make sense for an online gun purchase policy to say that Alice can buy a gun as long as she does not submit a convicted-felon credential. It would make sense, however, if the online gun-purchasing service could check an online listing of convicted felons before letting Alice buy a gun. When Alice has full control over her credentials, the gun service cannot distinguish whether Alice does not possess a certain credential or is simply unwilling to disclose it. A policy language must also let Bob specify which of the identities mentioned in a credential (if any) Alice must authenticate to. For example, if Alice presents a birth certificate during negotiation, does she need to prove that she is the child, the mother, the father, or none of these?

The lack of centralized control in the Internet makes it unlikely that a single language will be used for all policies. Researchers have already proposed several policy languages for use in trust management,^{4,10,11} and we expect policy language dialects to evolve for use in specific domains. We expect

that it will be possible to download certified decision engines for popular languages, such as descendants of KeyNote, IBM Trust Policy Language, and Role-Based Trust-Management Language (RT), from well-known servers on the Internet.

A policy language *compliance checker* accepts a set of credentials and a policy and returns a subset of the credentials that satisfy the policy, if such a subset exists. To do this, the policy compliance checker translates the credentials from a neutral format, such as XML, into statements in the policy language.¹² The policy language must have well-defined semantics, so both parties will always agree on whether a certain set of credentials satisfies a particular policy.

We have found that it is as hard to write a good policy as it is to write code in general; thus, policy capture and analysis tools will be very important.¹⁰ In the long term, common resources will likely come with standard policies already in place, and available tools will help users tailor these policies. With policies in place for many resources, we also expect policy management and updating to be an important area for tool development.

Negotiation Strategies

Different parties might have different requirements for how much computation they are willing to do, how freely they disclose resources, how interested they are in extracting information from the other party in the negotiation, and other such strategic decisions. For such decisions, each party relies on its trust negotiation strategies. A trust negotiation strategy tries to establish trust whenever possible. The number of possible strategies is almost limitless.

Because the Internet is a free-wheeling place with decentralized control, all parties that might need to negotiate trust – that is, all users, organizations, and software entities – should be free to choose whatever negotiation strategy meets their current needs. For example, parties running on portable thin clients, such as PDAs and smart cards, which only have low-end CPUs and limited memories, might prefer strategies that require little computation and memory resources but make irrelevant disclosures. More generally, two strangers will rarely wish to use the same strategy, and a party might prefer to use different strategies in different situations. How can we know that the strategies chosen by two parties will interoperate correctly?

We have identified four conditions that guarantee correct interoperation.

- Both strategies must use the same *protocol*.

TrustBuilder Protocol I

A trust negotiation protocol defines message ordering and information type. When a stranger (Alice) tries to access a resource R belonging to a stranger (Bob), the request triggers negotiation. Bob will start his end of the negotiation with the TrustBuilder Protocol I call `TrustBuilder_handle_disclosure_message` (\emptyset, R), shown in Figure A. Alice and Bob exchange messages until Bob discloses R or sends an empty disclosure message, signaling failure. Each message from Alice contains a set of Alice's local credentials and policies. This is also true for Bob's messages.

```

TrustBuilder_handle_disclosure_message ( $m, R$ ) {
  Input:  $m$  is the last message received.
          $R$  is the resource to which access was originally requested.
  TrustBuilder_check_for_termination( $m, R$ ).
  TrustBuilder_next_message( $m, R$ ).
}

TrustBuilder_next_message ( $m, R$ ) {
  Let  $M$  be the message sequence so far.
  Let  $L$  be this party's local resources and policies.
  // Let a local strategy suggest the next message.
   $m' = \text{Local\_strategy}(M, L, R)$ .
  Send  $m'$  to the other party.
  // Note: duplicate disclosures are forbidden.
  TrustBuilder_check_for_termination( $m', R$ ).
}

TrustBuilder_check_for_termination ( $m, R$ ) {
  If  $m$  is empty, then exit with failure.
  If  $m$  contains  $R$ , then exit with success.
}
    
```

Figure A. Pseudocode for TrustBuilder Protocol I. The two parties exchange messages, each containing a set of local credentials and policies, until R is disclosed or the negotiation fails.

While a strategy controls the content of trust negotiation messages, a protocol defines message ordering and the type of information messages will contain. For practical use, a few simple trust negotiation protocols suffice (see the sidebar on TrustBuilder Protocol 1).²

- Whenever one party's strategy recommends the disclosure of a resource, the credentials previously disclosed by the other party must satisfy the resource's policy.
- If the policies of two parties theoretically allow a successful trust negotiation, their strategies must recommend a sequence of disclosures that will establish trust.
- If a successful negotiation is not theoretically possible, the strategies must realize this and recommend termination of the negotiation.

To give parties the freedom to choose among strate-

gies, we have identified a maximal set of strategies known as the Binding Tree Strategy set, which satisfies these four conditions and supports structured credentials.² BTS includes a wide range of strategies: from those in which a party discloses every credential whose policy is satisfied, to those that disclose the absolute minimum of information necessary to keep the negotiation going. The Relevant Strategy, shown in Figure 1, belongs to this strategy set.

Architecture

Figure 2 shows a TrustBuilder security agent, which negotiates on behalf of a party to mediate strangers' access to the party's local resources. Alice's credential or policy is *disclosed* if Alice's security agent sends it to Bob's security agent during the negotiation, and Bob's service is disclosed if Bob's security agent gives Alice access to it.

Alice's security agent uses a policy compliance checker to determine which of Alice's policies are satisfied by Bob's disclosed credentials so that none of Alice's local resources is disclosed to Bob's agent before the resource's policy has been satisfied. Alice's agent also uses a compliance checker to determine which of Alice's credentials satisfy Bob's disclosed policies. When Alice's security agent receives a credential from Bob's agent, her credential verification module performs a validity check, including signature verification, revocation check, and credential chain discovery when necessary. The verification module also handles Bob's agent's demands that Alice demonstrate her possession of a private key that matches a certified public key.

Figure 3 (next page) presents the TrustBuilder architecture for ubiquitous trust negotiation. During negotiation, each security agent accepts new disclosures from the other agent, and uses a local strategy to determine which local resources to disclose next and whether to terminate the negotiation. The agent and the strategy also rely on the policy compliance checker and credential verification modules to carry out their functions.

Alice's negotiation strategy takes the current status of the negotiation, including Alice's credentials and policies, and all the credentials and policies Bob's agent has disclosed in previous rounds of the negotiation, and suggests the next message for Alice's agent to send to Bob's agent. The sample strategy code in Figure 1 recommends the disclosure of every credential whose policy is satisfied and is relevant to the negotiation – that is, the credential's type matches the credential variable that appears either in the target service's policy or in a policy of another relevant credential. Although not shown in

the figure, the message containing the disclosure of a policy for Alice's credential C also mentions which of Bob's disclosed policies C relates to, so that Bob will understand why Alice is sending him this particular policy. The "TrustBuilder in Action" sidebar shows this strategy in action.

Deploying TrustBuilder

Although the formal guarantees in the previous section are reassuring, a system is only as secure as its implementation. How can a party ensure that its trust negotiations are conducted in private and are not vulnerable to attack?

Conducting trust negotiation over a secure channel can solve many privacy and vulnerability problems. One way we accomplish this for interactions on the Web is to conduct TrustBuilder negotiations over an encrypted secure socket layer/transport layer security (SSL/TLS) session. In addition to possible attack, risks include compromised trust negotiation software and compromised private keys. To guard against attacks at the user level, we can place a user's credentials, private keys, and trust negotiation software on a smart card. The card should contain policy compliance checkers and a credential verification module from certified software providers, because those components must be trusted. To guard against server attacks, critical components can be placed in a cryptographic coprocessor.

To reach our goal of ubiquitous, scalable trust negotiation, we are designing and implementing reusable TrustBuilder components in a variety of computational environments: Web application servers, simple object access protocol (SOAP) remote procedure calls, Corba interceptors, SSL/TLS, and Internet protocol security (IPsec). Our TrustBuilder prototypes support X.509v3 certificates as the credential format and XML as the policy language. The policy language and compliance checker use IBM Research's Trust Establishment software.¹⁰

One prototype extends a Web application server to support trust negotiation between a client and server communicating through HTTP over SSL, as Figure 4 shows. Another deployment extends the TLS handshake¹³ to incorporate trust negotiation into TLS client-server authentication.¹⁴

The TLS TrustBuilder prototype extends PureTLS, a freely available Java implementation, to overcome several limitations in current TLS client-server authentication:

- certificates (credentials) are transmitted in plain text during the initial TLS handshake;

```

Input:
(m1, ..., mk): the sequence of disclosure messages sent and
received so far.
L: this party's local resources and policies.
R: the resource to which access was originally requested.
Output:
m: a disclosure message.
Pre-condition:
R has not been disclosed and the negotiation is not terminated.

D = ∪1 ≤ i ≤ k mi;
m = ∅;
For every local credential C relevant to R
if (C's policy is satisfied by D)
then m = m ∪ {C};
else m = m ∪ {C's policy};
m = m - D; // remove duplicate disclosures
return m;
    
```

Figure 1. Pseudocode for TrustBuilder's Relevant Strategy, a member of the Binding Tree Strategy set of strategies. It can interoperate with all other strategies in the BTS set.

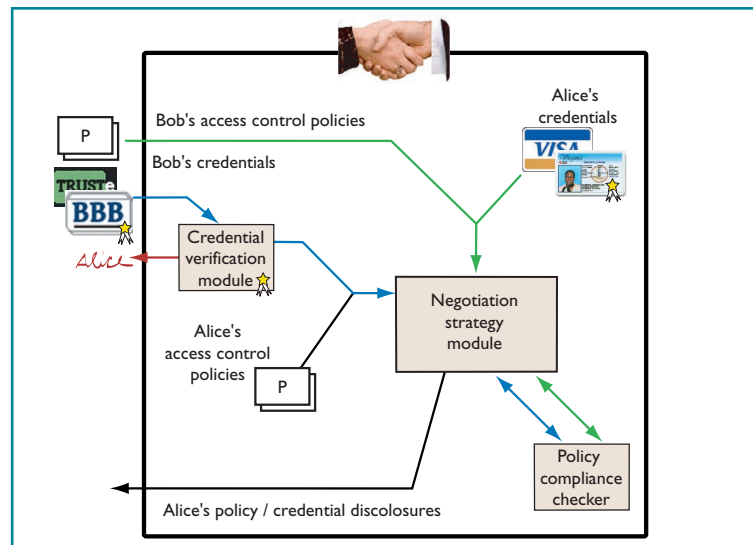


Figure 2. TrustBuilder security agent architecture. Alice's agent verifies Bob's disclosed credentials, checks whether Alice's credentials satisfy Bob's disclosed access control policies and if Bob's credentials satisfy Alice's access control policies, and determines what policies and credentials to disclose to Bob. Bob's security agent provides a corresponding functionality.

- the client and server can only disclose a single certificate chain with each other;
- only the server can specify a list of names that it trusts as certifying authorities;
- the server must disclose its certificate first; and
- neither party can request additional certificates during negotiation.

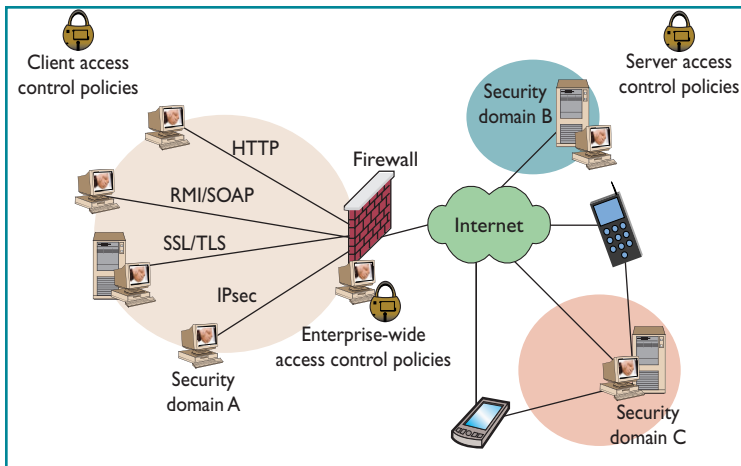


Figure 3. TrustBuilder architecture for ubiquitous trust negotiation. TrustBuilder security agents negotiate trust between strangers across security domains. TrustBuilder enforces client, server, and enterprise-wide access control policies.

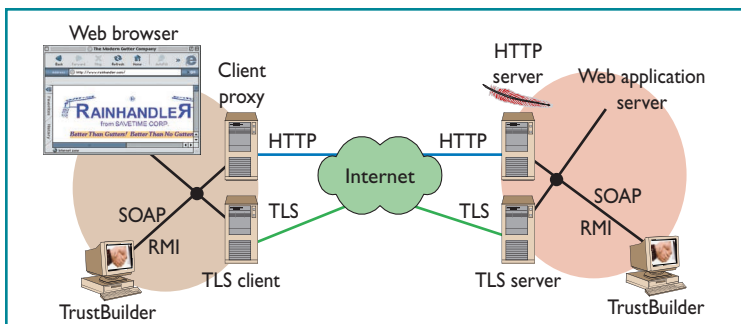


Figure 4. The TrustBuilder prototype architectures. Our Web application server supports trust negotiation through HTTP over SSL. The transport-layer security client-server prototype extends the TLS handshake to incorporate trust negotiation into TLS client-server authentication.

In the TLS TrustBuilder prototype, a client and server first establish a standard TLS session. Once the client requests a sensitive resource, the server initiates a trust negotiation by leveraging the TLS rehandshake facility. Because the rehandshake occurs during an already encrypted TLS session, the disclosures are protected from an eavesdropper.

We are also designing an out-of-band TrustBuilder prototype that lets a user store credentials on a single, secure server that negotiates trust on the user's behalf whenever the user contacts a sensitive service from another device. This will reduce the need for a user to replicate credentials and keys on every device and will hasten the move toward ubiquitous trust negotiation. Other areas for future work include enhanced privacy protection during trust negotiation, client-side access control, and prevention of attacks on trust negotiation.

Further information on TrustBuilder is available

at <http://isrl.cs.byu.edu/> and <http://dais.cs.uiuc.edu/>.

Conclusions

Automated trust establishment between strangers promises to extend trusted interactions to a broader range of participants than is possible with traditional security approaches based on identity and capabilities. With automated trust establishment between strangers, the number of sensitive business processes that can be accomplished electronically will grow substantially, which in the long run will enhance market efficiency and reduce business costs.

How can trust negotiation become popular, since people have not rushed out to obtain private keys? We believe that people will start to collect public and private keys, and credentials to accompany them, once there is a clear benefit in doing so. Adoption will be a grass-roots campaign, spurred by business and government initiatives, when automated services can streamline time-consuming, costly procedures and service providers demand credentials before granting access to these services. To simplify adoption, TrustBuilder's trust negotiation infrastructure is backward compatible with today's Internet environment and can be deployed locally as an extension to existing protocols.

We cannot control what recipients do with the information in a disclosed credential. As a result, policies might demand credentials from the requesting party certifying that it will not, for example, resell the information gathered during trust negotiation. Complicating matters, the parties can sometimes infer credential information from the policy disclosures made during negotiation. Privacy guarantees for parties negotiating trust are an interesting area for future research.

Integrating trust negotiation with Single Sign-On is another interesting area to explore. This would allow a client that has negotiated trust with a server to interact with servers in the same trusted domain without renegotiating trust at each Web site. □

Acknowledgments

This research was sponsored by the U.S. Defense Advanced Research Project Agency (DARPA) through the following contracts and grants: N66001-01-1-8908, F33615-01-C-1805, F30602-98-C-0222, F30602-97-C-0336, and DACA-88-94-0029.

References

1. P. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web," *Proc. 7th ACM Conf. Computer and Comm. Security*, ACM Press, New York, Nov. 2000, pp. 134-143.

2. T. Yu, M. Winslett, and K.E. Seamons, "Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation," to appear in *ACM Trans. Information and System Security (TISSEC)*, vol. 6, no. 1, Feb. 2003.
3. M. Blaze, J. Feigenbaum, and J. Lacey, "Decentralized Trust Management," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 164-173.
4. M. Blaze et al., "The Keynote Trust Management System, Version 2," Internet Engineering Task Force RFC 2704, Sept. 1999, www.ietf.org/rfc/rfc2704.txt.
5. C. Ellison et al., "SPKI Certificate Theory," IETF RFC 2693, Sept. 1999, www.ietf.org/rfc/rfc2693.txt.
6. N. Li, B.N. Grosof, and J. Feigenbaum, "A Practically Implementable and Tractable Delegation Logic," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 27-42.
7. J. Park and R. Sandhu, "Binding Identities and Attributes Using Digitally Signed Certificates," *Proc. 16th Ann. Computer Security Applications Conf.*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 120-127.
8. K.E. Seamons, M. Winslett, and T. Yu, "Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation," *Proc. Network and Distributed System Security Symp.*, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 109-125.
9. K.E. Seamons et al., "Requirements for Policy Languages for Trust Negotiation," *Proc. 3rd Int'l Workshop Policies for Distributed Systems and Networks (POLICY 02)*, IEEE CS Press, Los Alamitos, Calif., 2002, pp. 68-79.
10. A. Herzberg et al., "Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 2-14.
11. A.D. Keromytis, *Strongman: A Scalable Solution to Trust Management in Networks*, PhD diss., Computer Science Dept., Univ. of Pennsylvania, 2001.
12. A Herzberg and Y. Mass, "Relying Party Credentials Framework," *Proc. 2001 RSA Conf.*, Springer-Verlag, Berlin, Germany, 2001, pp. 328-343.
13. E. Rescorla, *SSL and TLS, Designing and Building Secure Systems*, Addison-Wesley, Boston, 2001.
14. A. Hess et al., "Advanced Client-Server Authentication in TLS," *Proc. Network and Distributed System Security Symp.*, IEEE CS Press, Los Alamitos, Calif., 2002, pp. 203-217.

Marianne Winslett is a professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Her research interests include security, databases, and high-performance computing. She is currently vice-chair of ACM SIGMOD, and a member of the *ACM Transactions on Database Systems* editorial board. She is a member of the ACM and the IEEE Computer Society.

Ting Yu is a PhD student in the Department of Computer Science, University of Illinois at Urbana-Champaign. His research interests include trust negotiation in open systems, access control and authentication in distributed environments, and database management systems. He has a BS in computer science from Peking University and an MS in computer science from the University of Minnesota, Twin Cities.

Kent E. Seamons is an assistant professor in the Computer Science Department at Brigham Young University and the director of the Internet Security Research Lab. His research interests include trust negotiation, security, distributed systems, and parallel I/O. He received a BS in computer science from Brigham Young University and a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE Computer Society, the ACM, the Internet Society, and Usenix.

Adam Hess is a graduate student in computer science at Brigham Young University and a research assistant in the Internet Security Research Lab. His research interests are in trust negotiation, access control models, and network security. He received a BS in computer science from Brigham Young University.

Jared Jacobson is a graduate student in computer science at Brigham Young University and a research assistant in the Internet Security Research Lab. His research interests are in trust negotiation, secure protocols, peer-to-peer security, and secure group management. He received a BS in computer engineering from Brigham Young University.

Ryan Jarvis is a graduate student in computer science at Brigham Young University and a research assistant in the Internet Security Research Lab. His current research addresses privacy protection during trust negotiation. He received a BS in computer science from Brigham Young University.

Bryan Smith is a graduate student in computer science at Brigham Young University and a research assistant in the Internet Security Research Lab. His research interests include strategies for trust negotiation and artificial intelligence. He received a BS in computer science from Brigham Young University.

Lina Yu is a graduate student in computer science at Brigham Young University and a research assistant in the Internet Security Research Lab. Her research interests are in protecting privacy during trust negotiation and trust negotiation mediators. She received a BS in computer science from the University of Science and Technology, Beijing.

Readers can contact Winslett at winslett@cs.uiuc.edu and Seamons at seamons@cs.byu.edu.