



# An Introduction to Automated Trust Negotiation

Marianne Winslett

University of Illinois at Urbana-Champaign

# Talk outline

- ◆ **Why do we need automated trust negotiation?**  
(Written for people who are not experts in security)
- ◆ Theoretical and practical issues raised by automated trust negotiation
- ◆ Example results

# Buying shirts from a stranger



Choose merchandise  
Produce credit card  
Scan card into computer  
Automated phone call  
Sign receipt  
Compare signatures



# Buying shirts from a stranger



**Forgery**

Choose merchandise

Produce credit card

Scan card into computer

Automated phone call

Sign receipt

Compare signatures

**Credit Limit**



**Theft**

**Revocation**



**Expiration**

**Impersonation**  
**Impersonation**



**Privacy**  
**Privacy**

# Goal

- ◆ Same ease of interaction between strangers on line (but for a grander goal)
- ◆ And with improved security and privacy
- ◆ And *ubiquitous*



# Ubiquitous

- ◆ All kinds of parties
  - People, organizations, software entities, hardware entities
- ◆ Wherever they might be
  - Any interaction across security domain boundaries---mobile or stationary
- ◆ Whatever they might be doing

# What might they be doing?

- ◆ Financial transactions
  - Purchases, auctions, account management
- ◆ Viewing sensitive documents
  - Medical records, military data
- ◆ Registration
  - School, voting, passports, marriage license, visa, library card
- ◆ Government and business
  - Adoption, citizenship, work permit, joint ventures
- ◆ Anything requiring paper credentials today

# Broader context: move toward open computing systems

Ability to form relationships and cooperate to solve urgent problems

- Joint military activities, joint corporate ventures, crisis management
- Unanticipated resource sharing across organizational boundaries

# Example: supply chain management

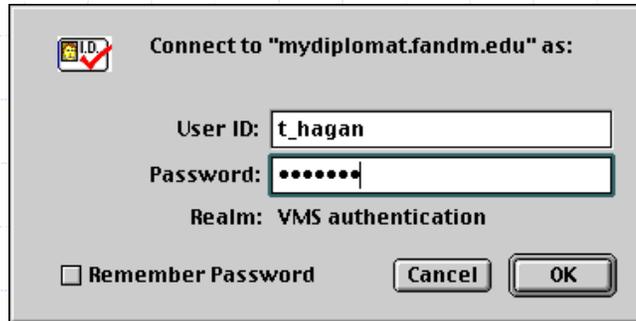
- ◆ Auto parts supplier accesses corporate database of auto manufacturer to determine number of widgets needed for planned production schedule
  - Employed by widget manufacturer (Employee ID)
  - Have privilege to plan production of widgets (delegation from production planning department of the supplier)

# Example: kindergarten registration

- ◆ Child's age over 5 (birth certificate)
- ◆ Be parent or legal guardian (from birth certificate or court order)
- ◆ Residency within the school district (driver's license)
- ◆ Child has had all required immunizations (clinic records)
- ◆ **Attribute-based** access control

# Traditional access control

◆ Assumption: I already know you



Connect to "mydiplomat.fandm.edu" as:

I.D.

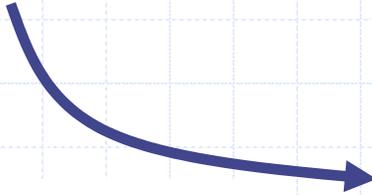
User ID:

Password:

Realm: VMS authentication

Remember Password

Not a member?



Thanks for your interest. Please fill out the information below in order to register. Fields marked with an "\*" are required fields.

*First Name	<input type="text"/>
Middle Name	<input type="text"/>
*Last Name	<input type="text"/>
*Email/Login:	<input type="text"/>
Alternate Email	<input type="text"/>
*Password	<input type="password"/>
*Verify Password	<input type="password"/>
Department/Faculty	<input type="text"/>
Institution	<input type="text"/>
Street Address	<input type="text"/>
Street 2	<input type="text"/>
City	<input type="text"/>
State/Province	<input type="text"/>
Country	<input type="text" value="UNITED STATES"/>
Zip/Postal Code	<input type="text"/>
Phone and extension	<input type="text"/>
Fax	<input type="text"/>

# Traditional access control for decentralized systems

- ◆ Identity-based (logins, identity credentials)
- ◆ Administered centrally
- ◆ Prevent resource sharing beyond organizational boundaries
- ◆ Hinder rapid, effective response to threats, opportunities
- ◆ Limit on-line service offerings

# Can we digitize (and improve) the paper-based approach?

## ◆ My credit card

- Digitize and make verifiable, unforgeable
- Provide way to prove ownership or delegation of authority to use

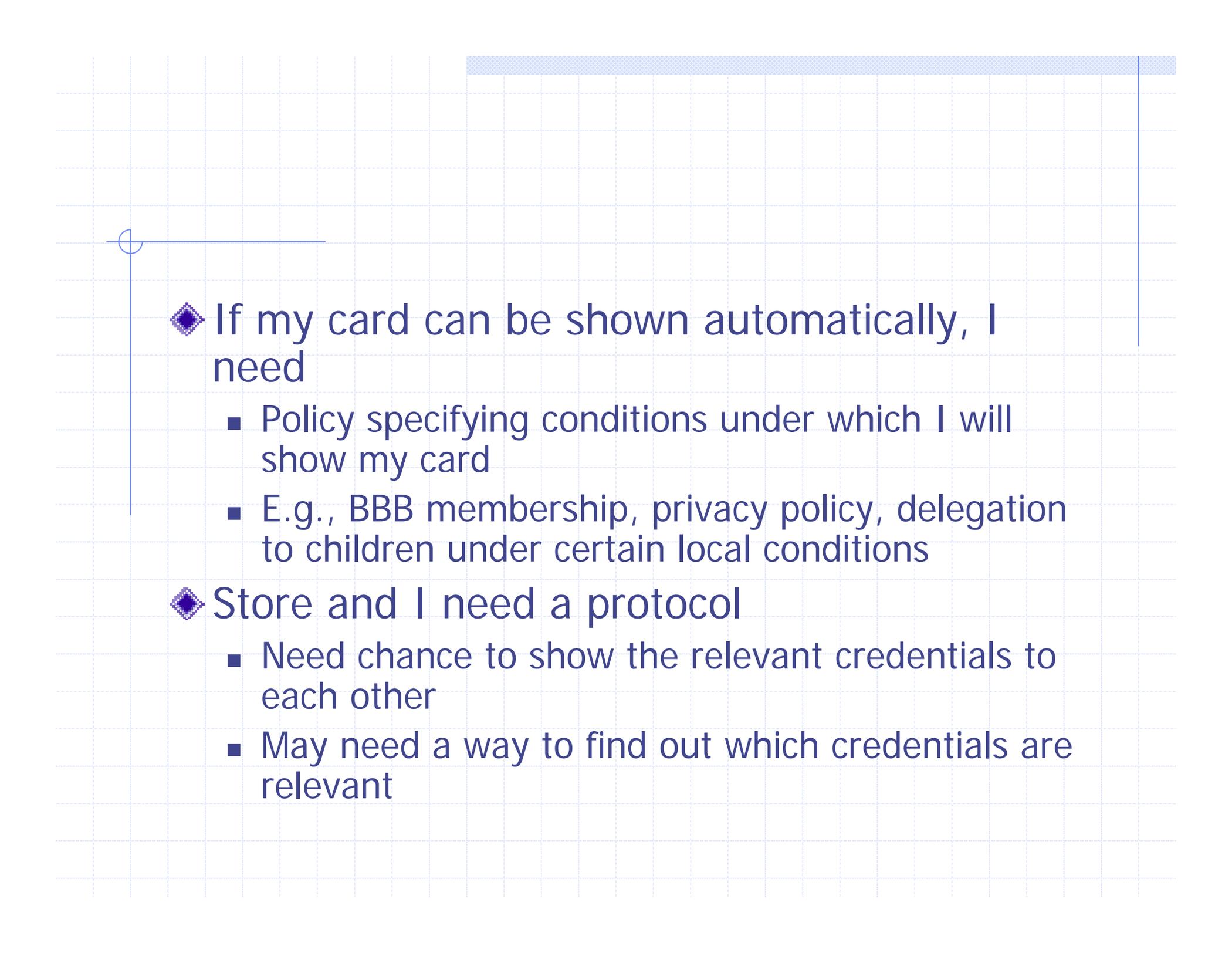
## ◆ Store recognizing the credit card

- Read and interpret fields of card
- Verify ownership/delegation



◆ Store needs policy for credit card acceptance

- Acceptable issuers
- Require ownership/delegation to be demonstrated
- Check for expiration
- Contact card issuer
  - ◆ Revocation, credit limit



◆ If my card can be shown automatically, I need

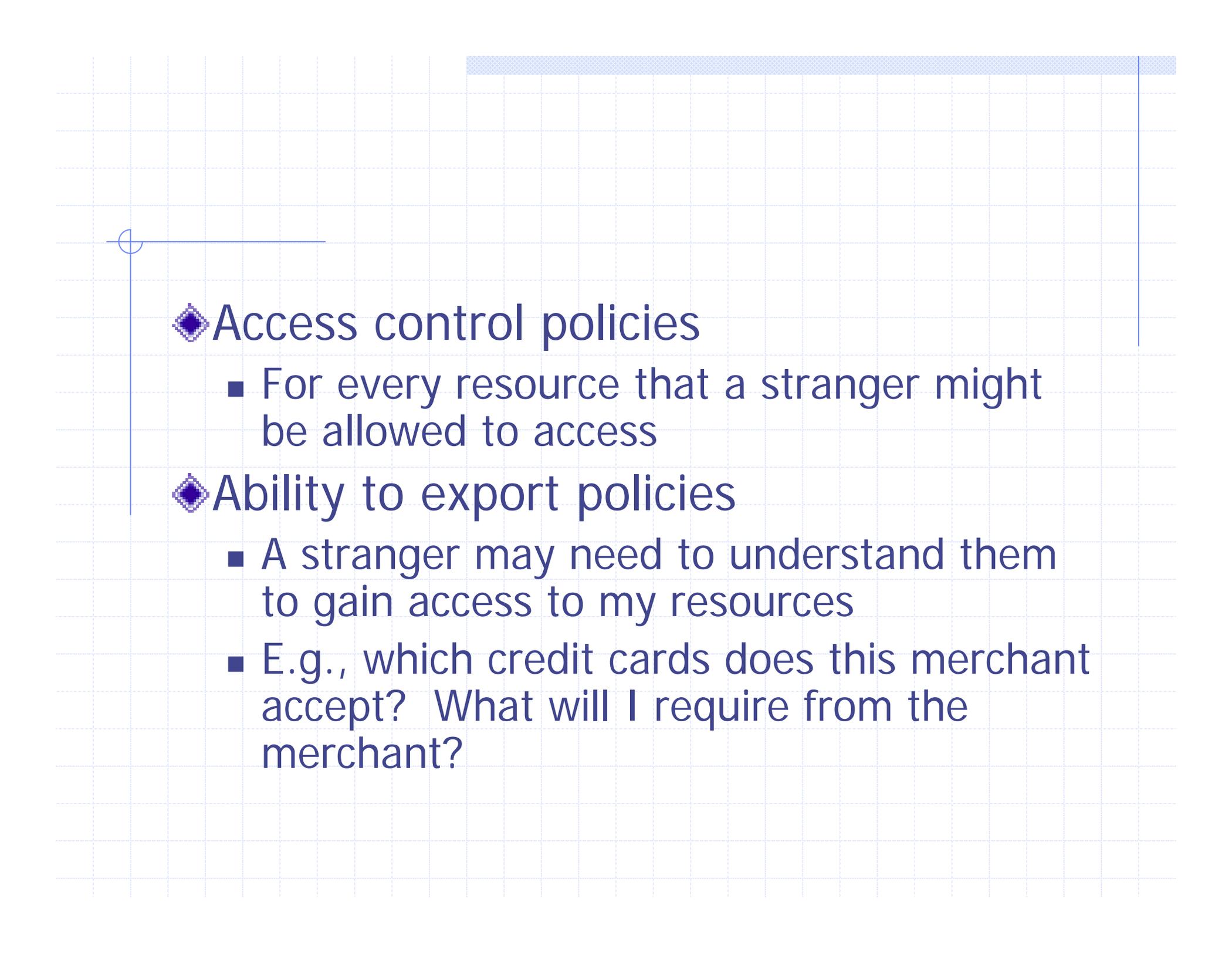
- Policy specifying conditions under which I will show my card
- E.g., BBB membership, privacy policy, delegation to children under certain local conditions

◆ Store and I need a protocol

- Need chance to show the relevant credentials to each other
- May need a way to find out which credentials are relevant

# We know how to do most of this!

- ◆ Credentials: X.509 and *beyond*
  - Improve privacy, nonforgeability, single-versus multiple-use, ...
  - Standard languages/ontologies for expressing credential contents
- ◆ Domains of trust: PKI and beyond
  - VISA International, BBB, my employer, etc., as roots of PKI hierarchies



## ◆ Access control policies

- For every resource that a stranger might be allowed to access

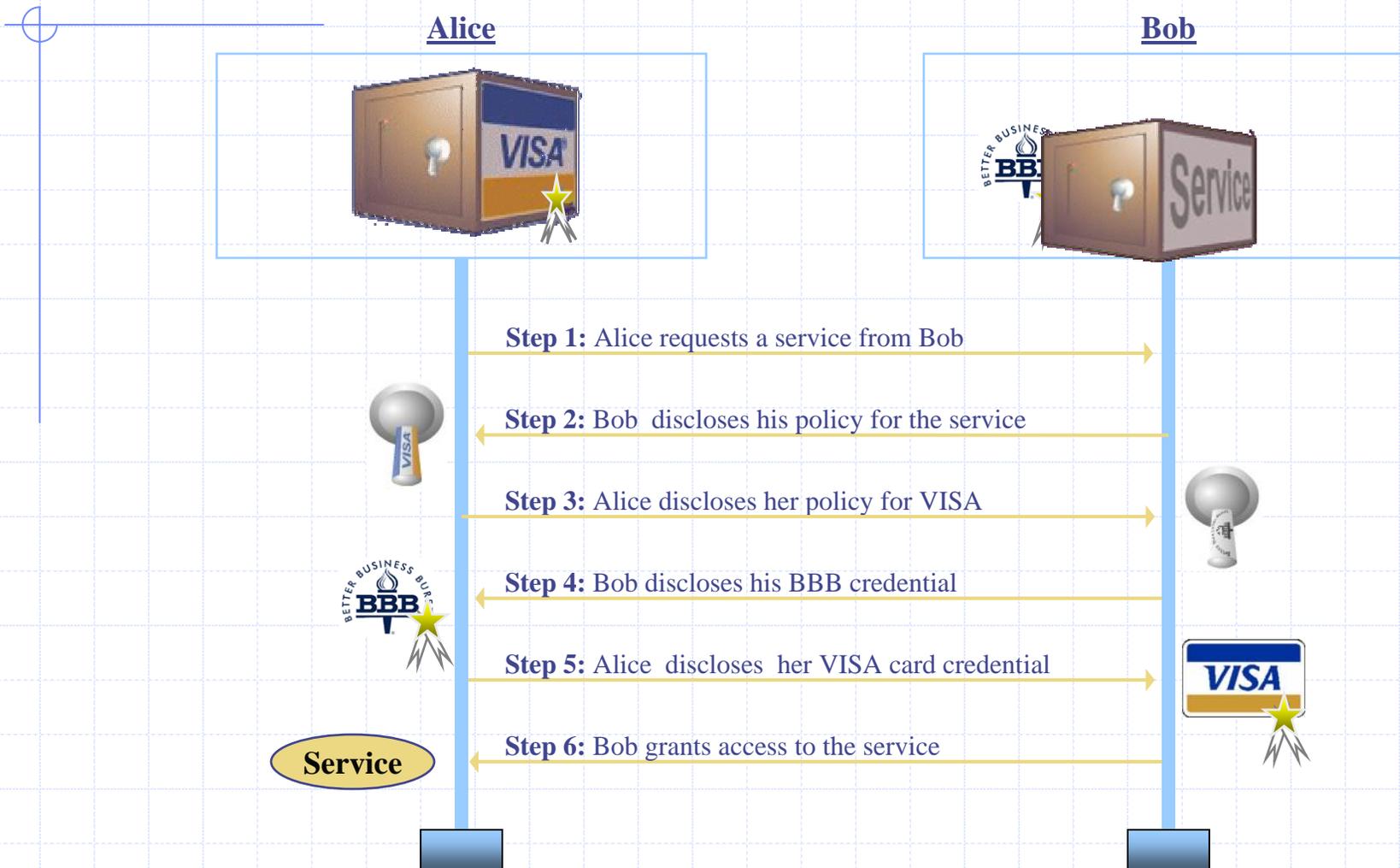
## ◆ Ability to export policies

- A stranger may need to understand them to gain access to my resources
- E.g., which credit cards does this merchant accept? What will I require from the merchant?

# Pulling it all together

- ◆ Parties decide which credential issuers they trust for what purposes
- ◆ Parties turn those trust decisions into access control policies
  - All organizations authorized by VISA Inc. to issue VISA cards
- ◆ Parties cache relevant credentials locally, search out others dynamically

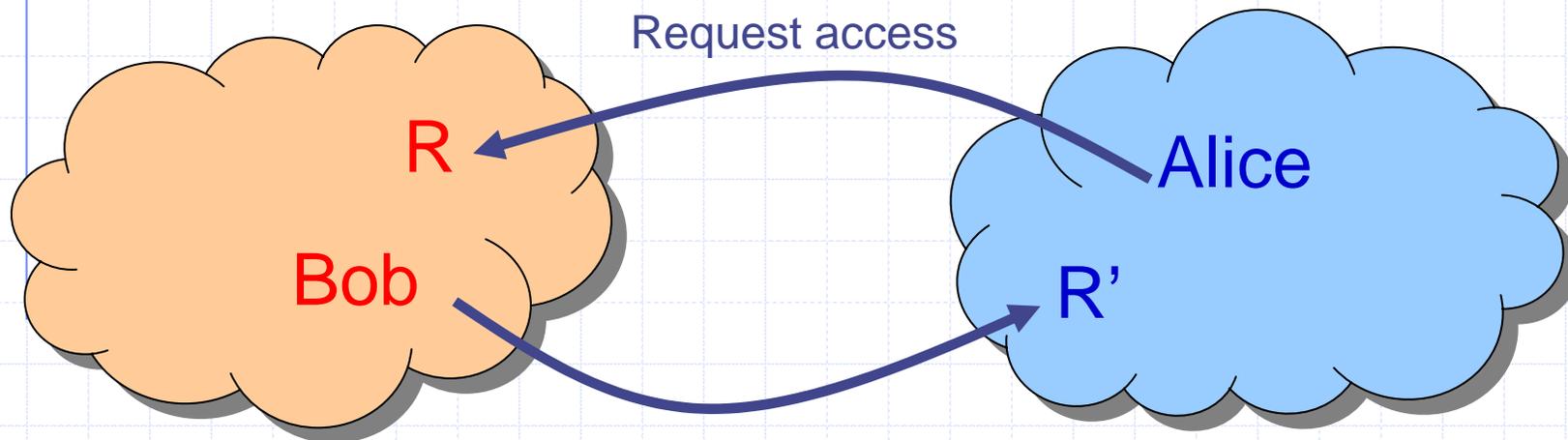
# Trust Negotiation E-business Example



# Talk outline

- ◆ Why do we need automated trust negotiation?
- ◆ **Theoretical and practical issues raised by automated trust negotiation**
- ◆ Example results

# The underlying challenge



Fully automated

# Policy and credential capture and interpretation

*Access control policies play a central role!*

- ◆ Expressive policy languages
- ◆ Tools to help people write, update, and analyze policies
- ◆ Standard schemas/ontologies for popular types of credentials
- ◆ Efficient policy compliance checkers
- ◆ Protection as strong as for any resource

# Needed language features

- ◆ Well-defined semantics
- ◆ Monotonicity
- ◆ Everything relational algebra can do, plus transitive closure
- ◆ Support for delegation
- ◆ References to the local environment and external functions (e.g., time of day, current user)
- ◆ Explicit specification of authentication requirements

# Trust negotiation architectures

- ◆ Trusted third parties that are not vulnerable to attack
- ◆ Direct peer-to-peer
  - With disclosure of credentials/policies
  - Zero knowledge
  - Hidden credentials/OSBE

# Trust negotiation strategies

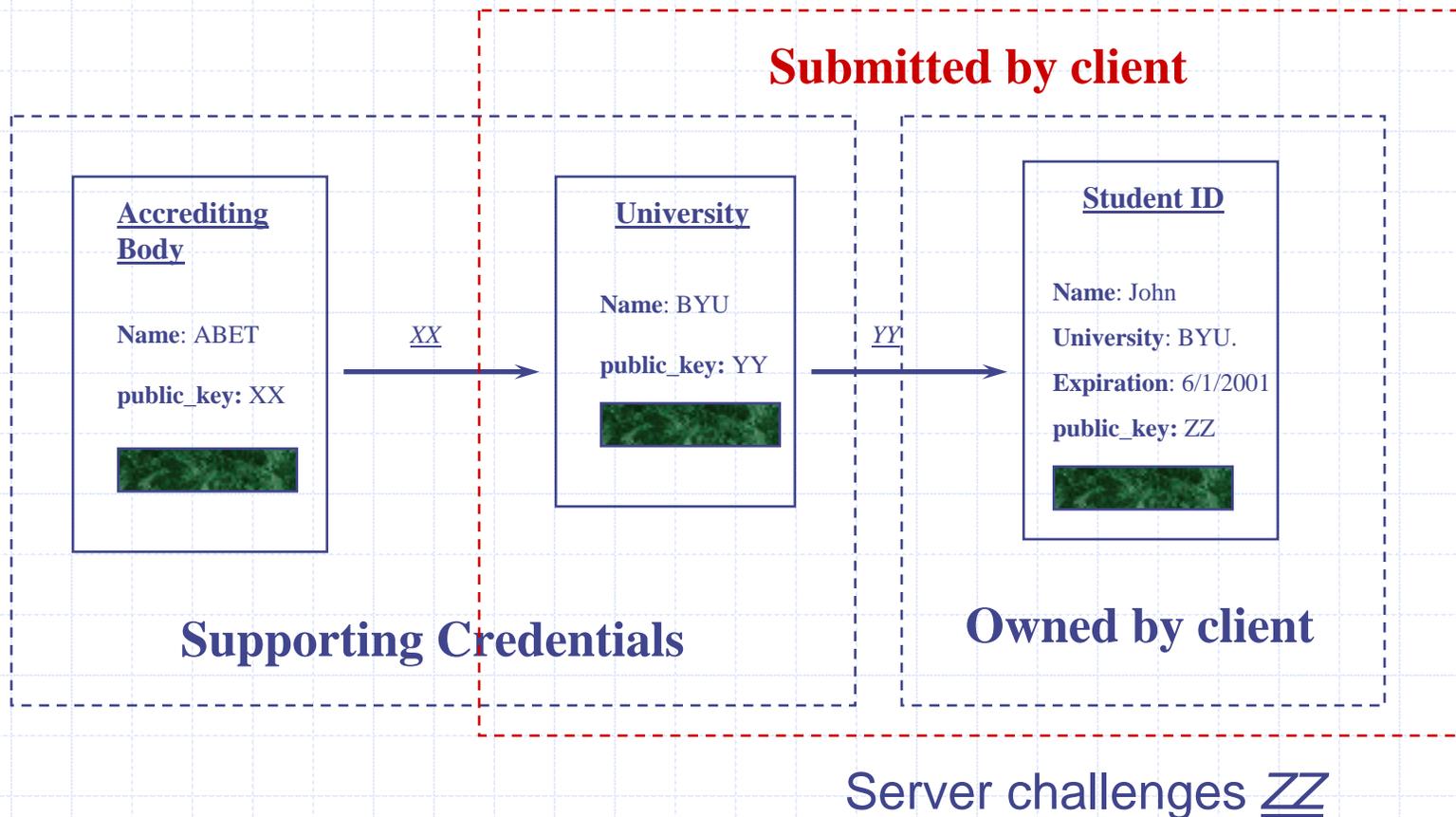
- ◆ Relevant for approaches that disclose credentials and/or policies
- ◆ Out of all the credentials and policies that I *could* disclose next, which should I actually disclose next?
  - Willing to show contents of my purse
  - But is there a need to know?
- ◆ Autonomy? Interoperability?

# Obtaining and storing credentials

- ◆ How do I get them?
- ◆ Where do I keep them, to keep them private?
- ◆ How can I quickly find credentials I haven't cached already, during a negotiation? (credential chain discovery/n-party trust negotiation)

# Credential Chains - Web of Trust

Establishing server trust that the client is a student at an accredited university



# Scalability and deployment

- ◆ How can we implement trust negotiation in a modular, scalable, and reusable manner that will support ubiquitous trust negotiation?
- ◆ How can trust negotiation be included in today's popular communication protocols (SOAP, IPsec, TLS, etc.), in a backwardly compatible manner?

# Vulnerabilities

- ◆ What kinds of attacks is trust negotiation vulnerable to?
- ◆ How can we mitigate the danger?
- ◆ What parts of the process/system must be trusted, and to what degree?
- ◆ What privacy guarantees can we give?

# Privacy guarantees

- ◆ Can outsiders eavesdrop on negotiations?
- ◆ Can I disclose just part of a credential?
- ◆ Can there be a concept of “need to know”?
- ◆ What can be inferred about my credentials *without* my disclosing them? (leaks)

# Managing multiple identities

- ◆ Support for many identities has many benefits for issuers and owners, today and in the future
- ◆ How to prove I possess several identities, while preventing or penalizing collusion?
- ◆ How to make my identities unlinkable?

# Talk outline

- ◆ Why do we need automated trust negotiation?
- ◆ Theoretical and practical issues raised by automated trust negotiation
- ◆ Example results
  - **Overview of the TrustBuilder project**
  - Focus on systems work (as time permits)

# TrustBuilder faculty and close collaborators

## ◆ Theory

- **M. Winslett, UIUC**
- T. Yu, NCSU
- N. Li, Purdue
- W. Winsborough, GMU

## ◆ Systems

- **K. Seamons, BYU**

## ◆ Applications

- W. Nejdl, U. Hannover

## ◆ Funding

- DARPA Dynamic Coalitions Program
- NSF (ITRs on TN, disaster response)
- Industry (Zone Labs, Dallas Semiconductor, Network Associates Laboratories)

# Our major efforts

## ◆ Policy languages

- RT (constraint datalog), policy analysis tools/computability, finding credentials at run time, preventing leaks/attacks during negotiation, support for sensitive access control policies

## ◆ Negotiation protocols & strategies

- Range of possible strategies, autonomy and interoperability

## ◆ Testbed implementations

- HTTPS, TLS, content-triggered TN, hand-held TN, PeerTrust, ...

# Policy languages: the RT family

- ◆ Versions to support delegation, credentials with internal structure, resources with internal structure, etc.
- ◆ Semantics based on Datalog
- ◆ Li, Winsborough

# Finding credentials at run time

- ◆ The “credential chain discovery problem”: addressed by introducing a typing system for credentials and a search mechanism
  - Li & Winsborough
- ◆ “N-party trust negotiation”: a peer to peer approach
  - Nejdil, Winslett, soon Bertino

# Policy analysis algorithms & tools

- ◆ New computability results for analysis of RT policies
  - Li & Winsborough
  - Planned continuation to turn computability results into algorithms and tools

# Leaks during trust negotiation

- ◆ Behavior during a negotiation gives strong clues about what credentials you might have (even with a zero-knowledge approach)
- ◆ An attacker can provoke even more leaks
  - Game theory (Winslett)
- ◆ Proposed remedies
  - Acknowledgement policies (Li & Winsborough)
  - Hidden credentials/OSBE (Seamons/Li & Winsborough)
  - Probabilistic approaches (Winslett)
  - Policy filtering (Yu, Winslett)
  - And more ...

# Support for sensitive policies

- ◆ Acknowledgement policies (to some degree)
- ◆ UniPro
  - Treats policies as first class named resources that can be protected like any other resources
  - Yu, Winslett

# Protocols & strategies, interoperability and autonomy

◆ Goal: allow autonomy while supporting interoperation

◆ Approach

- Simple protocols for permissible message exchanges
- Large sets of strategies with proven interoperability (completeness)
- Extensions to n-party trust negotiation
- Yu, Winslett, soon Bertino

# Systems issues

- ◆ The TrustBuilder prototype
- ◆ Extensions to TLS, SMTP, HTTPS, IPSec, etc., to support trust negotiation
- ◆ Surrogate TN (for wireless devices)
- ◆ Content-triggered TN (for pushing sensitive information)
- ◆ Seamons

# Main applications

## ◆ Educational consortia

- PeerTrust (n-party trust negotiation)
- NejdI, Winslett

## ◆ Disaster management

- In LA
- Cross-disciplinary: police, firefighters, government officials, computer scientists, sociologists

# Talk outline

- ◆ Why do we need automated trust negotiation?
- ◆ Theoretical and practical issues raised by automated trust negotiation
- ◆ Example results
  - Overview of the TrustBuilder project
  - **Focus on systems work** (as time permits)

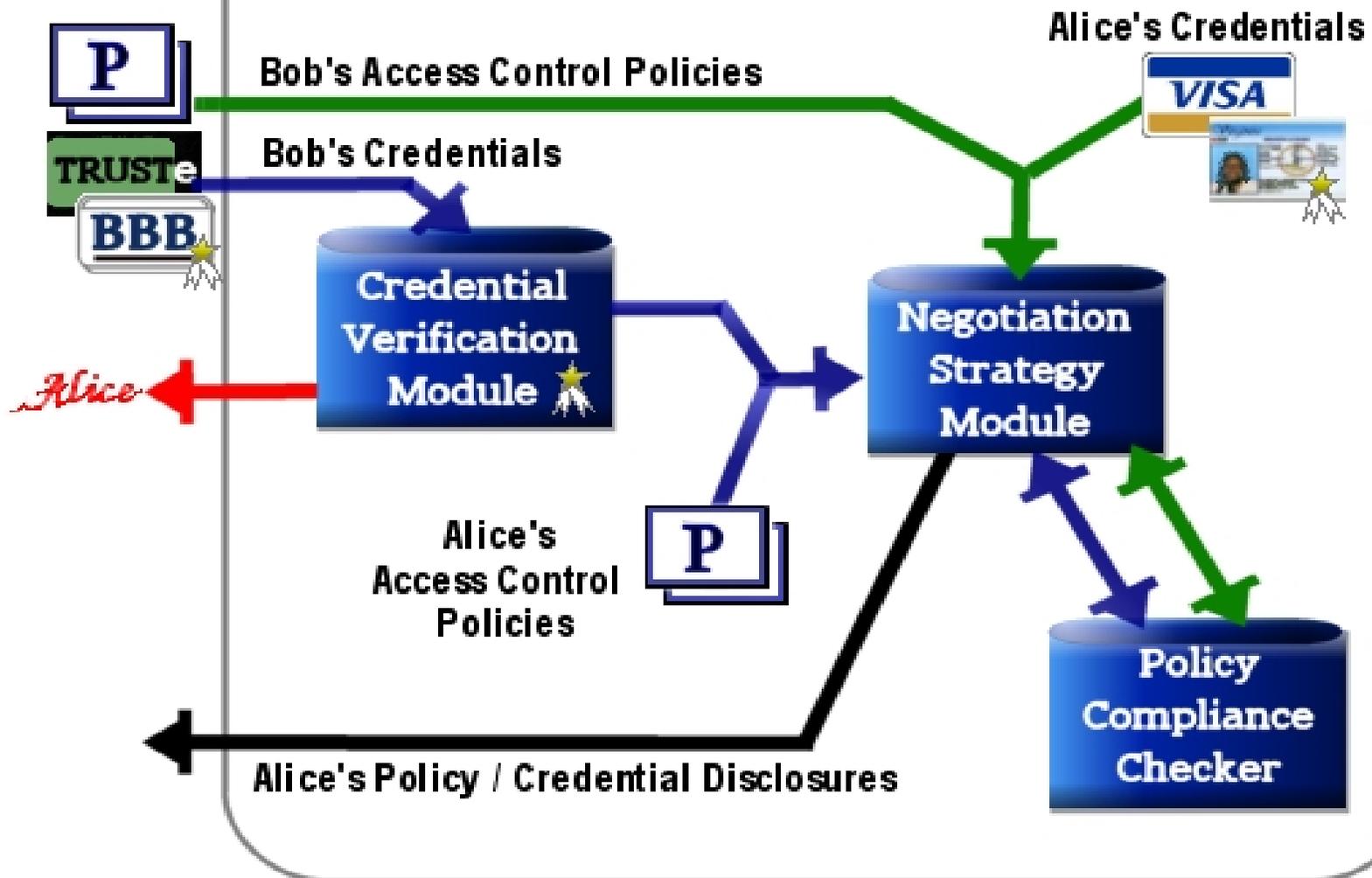
# TrustBuilder systems work at BYU

## ◆ Goals

- Ubiquitous trust negotiation facilities, to meet security needs at all levels
  - Scalable, modular, reliable implementations of those facilities
- ## ◆ Overall strategy: deploy TrustBuilder in every popular communications protocol
- HTTPS, TLS, SMTP, ssh, more on the way
  - As security agents, in Java



# TrustBuilder Security Agent



# TrustBuilder in HTTPS

- ◆ Client, server establish normal TLS/SSL session
- ◆ Then trust negotiation messages are passed back and forth in HTTPS headers
- ◆ Use of HTTPS protects against eavesdropping
- ◆ Convenient when authorization module is implemented at the application level (with respect to the web server)

# Negotiating trust in TLS

- ◆ Transport Layer Security (TLS) = IETF version of SSL 3.0
- ◆ TLS-level trust negotiation facilities are useful when
  - web server/client have proxies that can negotiate trust for them
  - web server/client know how to do trust negotiation directly
  - Other protocols built on top of TLS (e.g., SOAP) need to negotiate trust, but lack the facilities
  - TLS security (confidentiality, authentication of identity) is too limited for a particular web server/client

# Limitations of TLS authentication

- ◆ Certificates are exchanged in plain text
- ◆ Client and server each disclose only one certificate chain
- ◆ Server can specify a list of trusted certifying authorities; client cannot
- ◆ Server always discloses its certificate first
- ◆ Server certificate ownership is not yet established when the client discloses its certificate

# Trust Negotiation in TLS (TNT)

- ◆ Extends the TLS handshake protocol (where TLS does its authentication)
- ◆ Leverages existing and proposed features of the TLS handshake protocol
  - Client hello and server hello extensions
  - TLS rehandshake
  - Session resumption (avoids expensive computation of a new session key)
- ◆ Retains compatibility with TLS
- ◆ Implementation extends PureTLS (free, Java)

# Handshakes in TLS / TLS + RSA key exchange algorithm

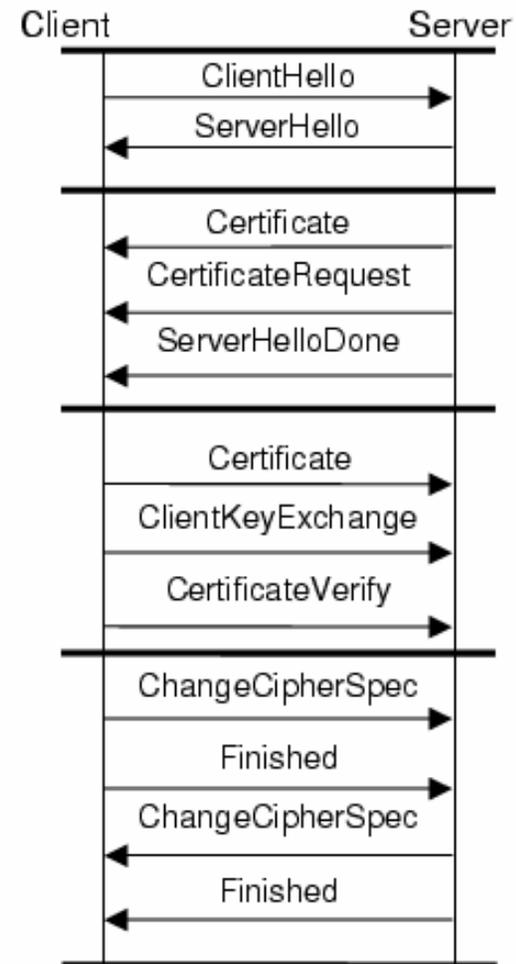
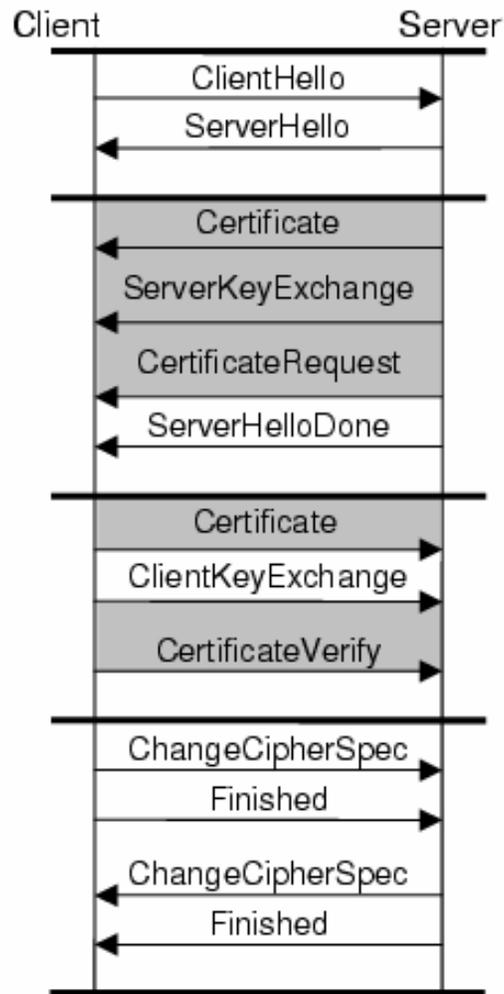


Figure 2 The TLS handshake protocol for

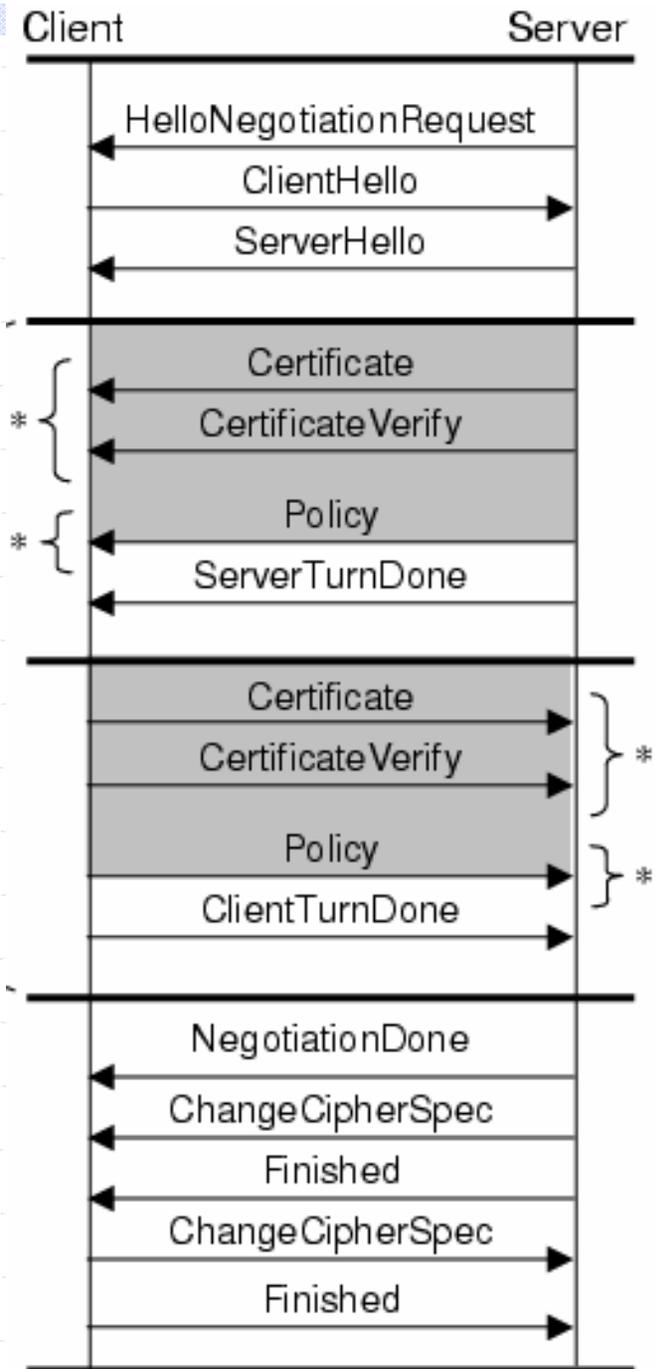
# TLS+RSA Hello Parameters

TNT adds NegotiationStrategyFamily parameter for ClientHello and ServerHello

Parameters	ClientHello	ServerHello
Version	Highest SSL/TLS version supported by client	Lower of the client-suggested version and highest server-supported version
Random	Client-generated random structure, used as a nonce	Server-generated random structure
SessionID	Variable-length session identifier. A zero value indicates a new session. A non-zero value refers to an earlier session the client wishes to resume.	If client sends a zero value, server returns a new session ID, otherwise returns the old session ID supplied by the client.
CipherSuite	List of cryptographic algorithm combinations the client supports, in decreasing order of preference.	Single cipher suite selected from the list supplied by the client.
Compression Method	List of the compression methods supported by the client	Compression method selected by the server.

# TNT Protocol

- ◆ New parameters:
  - HelloNegotiationRequest
  - ServerTurnDone
  - ClientTurnDone
  - NegotiationDone
  - Policy (opaque struct, specific to the negotiation strategy family)
  - Never part of initial handshake
  - Server doesn't know URL/POST data yet
- ◆ Rehandshakes normally used to upgrade cypher suite, change master secret



# TNT enhances TLS authentication

- ◆ Negotiation occurs on an encrypted channel
- ◆ Client and server can exchange multiple certificate chains
- ◆ Either the client or server can disclose certificates first
- ◆ Client and server can exchange multiple policies
- ◆ Client and server demonstrate certificate ownership as certificates are disclosed

# Many network protocols disclose sensitive content

ISRL

Internet Security Research Lab  
BRIGHAM YOUNG  
UNIVERSITY

Protocol	Potential Sensitive Data in Protocol
HTTP	form data, headers, cookies, URLs
SMTP	email messages, attachments
FTP	transferred files
SOAP	method parameters and names
NNTP	uploaded news posting
CORBA	method parameters and names

# Pushing sensitive content

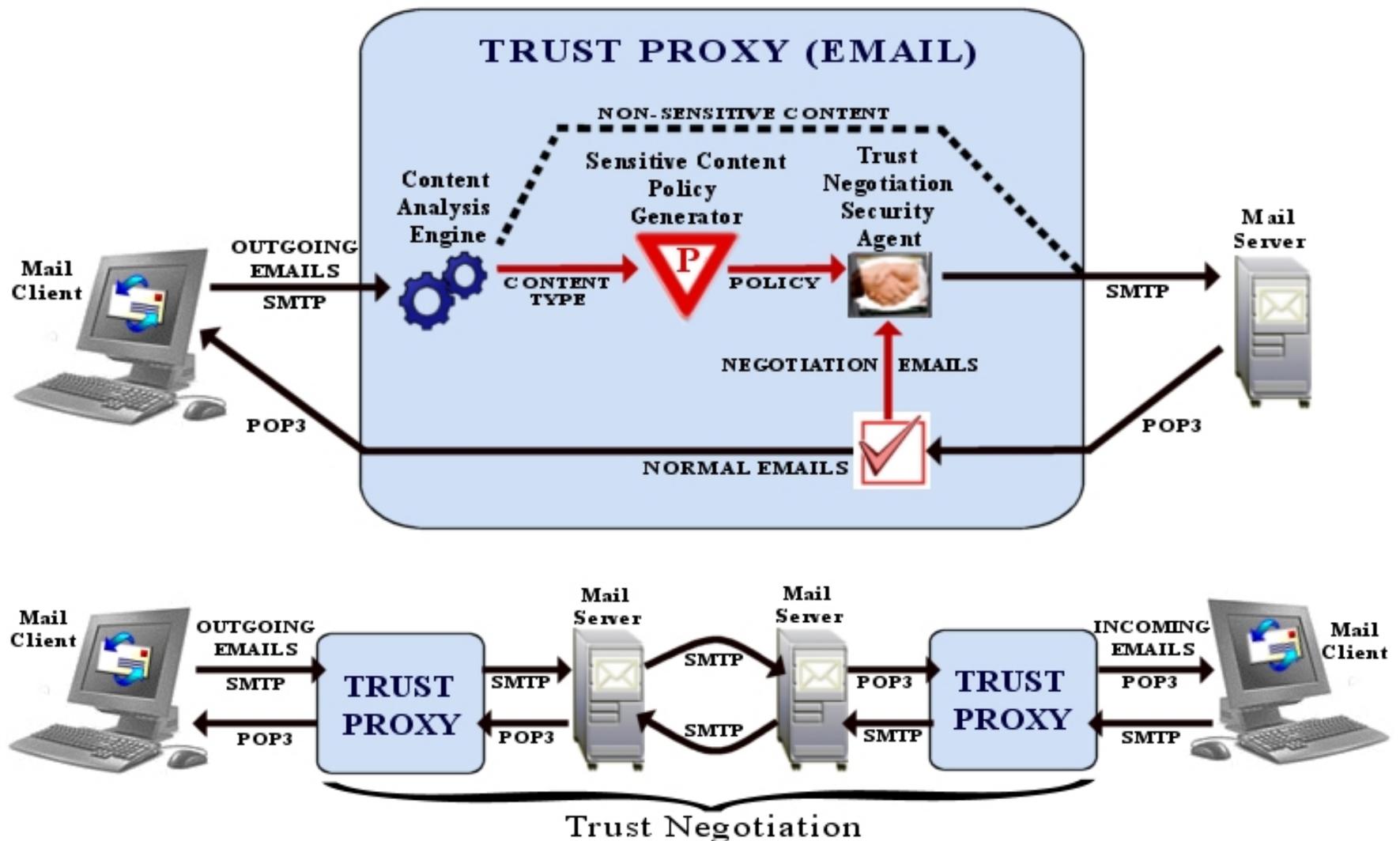
## ◆ Problem:

- These protocols disclose sensitive information to strangers without verifying that the recipient is authorized to receive it
- Sensitive content is frequently generated dynamically, making it difficult to associate access control policies with the content in advance

## ◆ Solution:

- A (non-malicious) attempt to transmit sensitive data generates appropriate policies dynamically and initiates “content-triggered trust negotiation”

# SMTP content-triggered TN



# Other example uses

## ◆ Typo pirates:

- [www.paypa1.com](http://www.paypa1.com) vs. [www.paypal.com](http://www.paypal.com)

## ◆ HTTP URL login

- <http://www.trustedsite.com/~.../@hacker.org>

Deceptive login name

Actual URL

# ◆ IE address bar URL spoofing flaw

(announced Dec. 10, 2003 by Sam Greenhalgh)  
(patch available Feb. 2, 2004 from Microsoft)

- `http://microsoft.com[null character]@hacker.org`

causes browser to display

- `http://microsoft.com`

# Phishing attacks

## ◆ Definition:

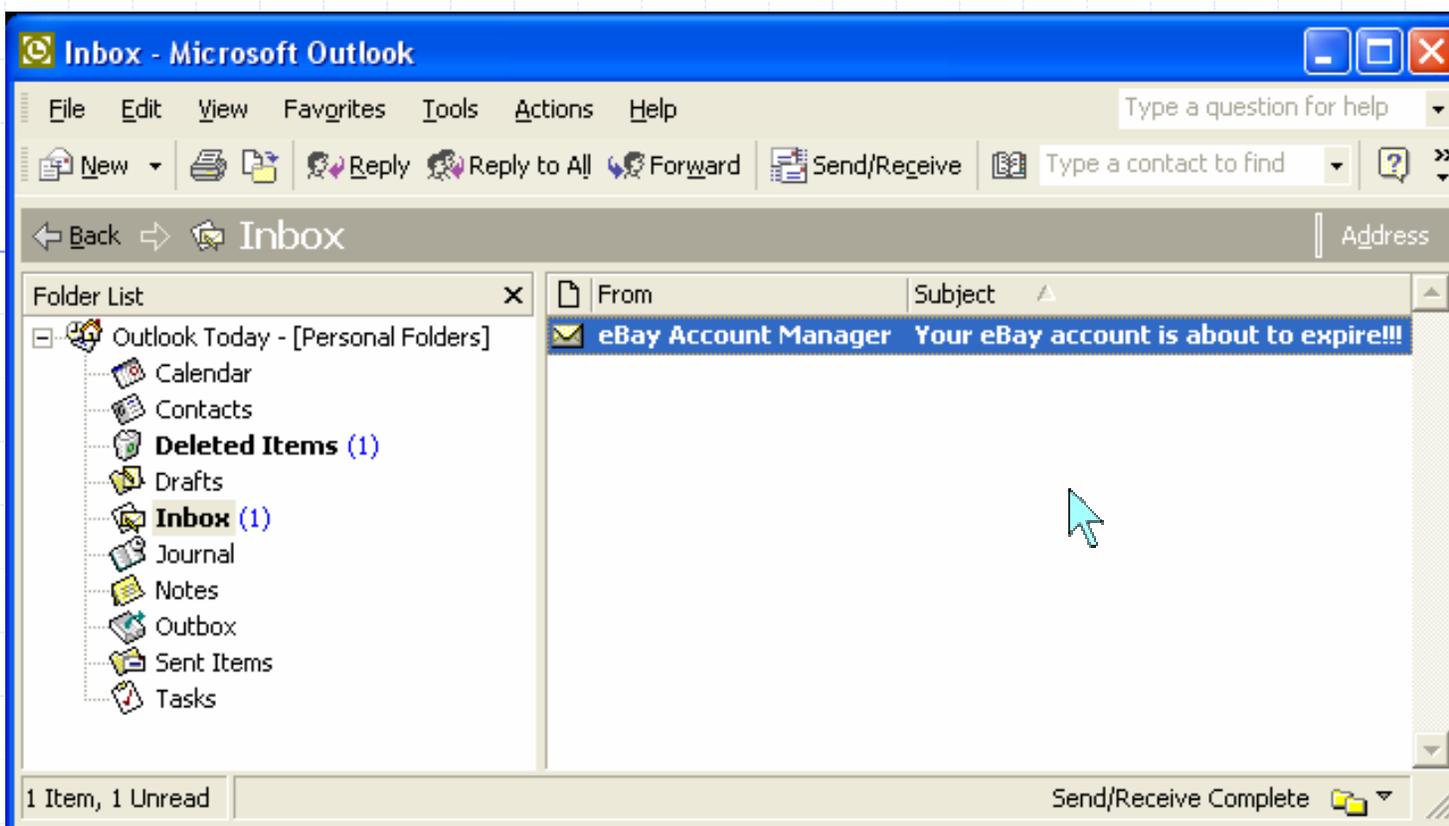
“The mass distribution of e-mail messages with return addresses, links, and branding which appear to come from legitimate companies, but which are designed to fool the recipients into divulging personal authentication data”

([www.antiphishing.org](http://www.antiphishing.org))

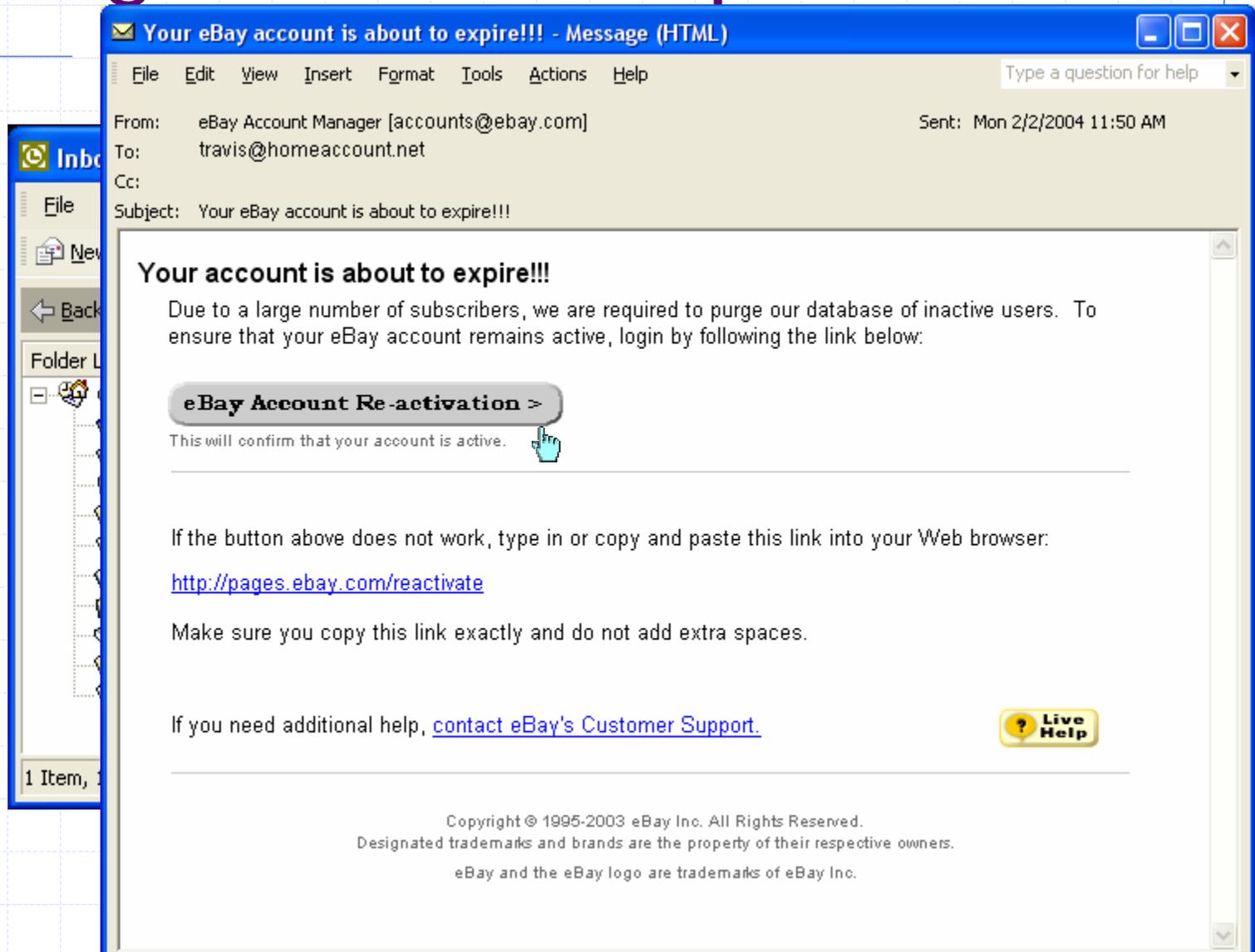
◆ “Up to 20% of recipients may respond to [the phishing attack], resulting in financial losses, identity theft, and other fraudulent activity.”

([www.antiphishing.org](http://www.antiphishing.org))

# Phishing attack example



# Phishing attack example





Possible URL spoofing attack:  
http://pages.ebay.com/reactivate[null]@steal\_your\_identity.com

New to eBay?

If you want to sign in, you'll need to register first.

Registration is fast and free.

Register >

or

Already an eBay user?

View all your bidding and selling activities in one location.

eBay User ID

travis2004

[Forgot](#) your User ID?

Password

\*\*\*\*\*

[Forgot](#) your password?

Sign In >

[Keep me signed in](#) on this computer unless



[Account protection tips](#) | [Secure sign in \(SSL\)](#)



