Distributed Authorization by Multiparty Trust Negotiation

Charles C. Zhang and Marianne Winslett, ESORICS 2008

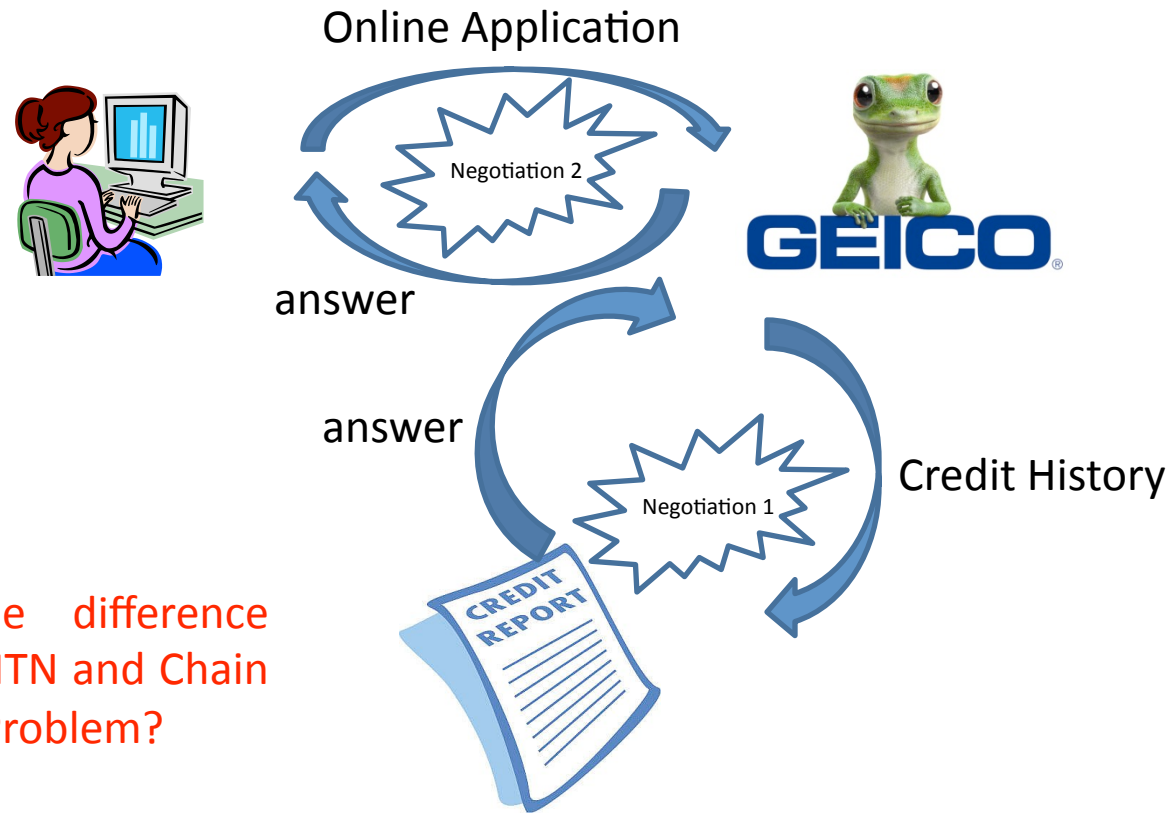
Yue Zhang

yzhang@sis.pitt.edu

Sep 28th, 2009

Motivation

- Sometimes trust negotiation among more than two parties are needed, and can not be solved using several two-party negotiations



Discussion:
What's the difference between MTN and Chain Discovery Problem?

Solution

- A multiparty trust negotiation framework
- Contributions:
  - A language: DARCL
  - A protocol: diffusion protocl
  - Two strategies: Eager and Cautious
- xxx

Key Idea

- Allow the policy writer to specify the source and destination of a credential – to control the negotiation flow

  - Geico↑Geico.approve(Alice)↓Alice←CreditU↑CreditU.goodCred(Alice)↓Geico ∧ ⋯

DARCL

$$rule ::= disclosure \leftarrow disclosure \ \wedge \ \cdots \wedge \ disclosure$$
$$disclosure ::= peer \uparrow credential \downarrow peer$$
$$credential ::= peer.credential\_name(term, \ \ldots, \ term)$$
$$term ::= peer \mid value$$
$$peer ::= variable \mid peer\_name$$
$$value ::= variable \mid \text{string}$$
$$variable ::= x \mid y \mid \ldots$$
$$peer\_name ::= Alice \mid Bob \mid \ldots$$
$$credential\_name ::= \text{string}$$

Example Policy

- EM can issue a visa to Alice only if : (1) Alice shows her Canada passport to EM; (2) DFS shows to EM that Alice has passed background check done by DFS; (3) Alice shows to EM that she is willing to allow DFS release background check result to EM

EM:

$$EM \uparrow EM.visa(x) \downarrow x \leftarrow x \uparrow Canada.passport(x) \downarrow EM \wedge x \uparrow x.OkToRelease(DFS, EM) \downarrow EM \wedge DFS \uparrow DFS.clear(x) \downarrow EM$$

Example Policy

- Alice is willing to show her passport to EM if: (1) EM shows to Alice that it is an official Embassy; (2) Alice has her passport by hand

$$Alice \uparrow Canada.passport(Alice) \downarrow x \leftarrow x \uparrow MG.officialEmbassy(x) \downarrow Alice \wedge Canada.passport(Alice)$$

Discussion:
The policies in their framework seems too specific and ad-hoc. How does Alice know to specify such policy regarding to visa application when she joins the network?

Inference Rule

Local Inference Rules A peer A can use the following local derivation rules.

- **Instantiation.** From a rule r in A's knowledge base, derive an instance of r by replacing all occurrences of the same variable in r with another variable or literal.
- **Knowledge.** From $B \uparrow d \downarrow A$, derive $A \uparrow d \downarrow A$.
- **Modus ponens.** From a rule $d_0 \leftarrow d_1 \wedge \cdots \wedge d_n$ and facts d_i, $1 \leq i \leq n$ in A's knowledge base, derive d_0.

Global Inference Rule

- From $A \uparrow d \downarrow B$ in A's knowledge base, where d is ground, derive $A \uparrow d \downarrow B$ in B's knowledge base.

In practice, no one is interested in discover all such knowledge from Global Inference Rule. One party is usually only interested on the knowledge that is directly related to the service – ensured by proper MTN protocol and strategies

Safe Disclosure Sequence

- A sequence of disclosure is safe if each disclosure in it is safe when it takes place
- A disclosure is safe if it is unlocked when it takes place
- A disclosure d=peer1↑C↓peer2 is unlocked if (1) d is already in peer1's knowledge base; or (2) d can be derived in peer1's knowledge base using local inference rules
- The goal of trust negotiation protocol and strategy is to find such safe disclosure sequence!

Diffusion Protocol

```
PARTICIPATEINMTN () {
  while willing to participate do
    if the incoming message queue is empty then
      Wait a short period of time for new messages
    if there is a new message in the incoming message queue then
      Choose such a message m and remove it from the queue
      Record m's receipt time as the current time
      ProcessMessage(m)
}

/* message handler */
PROCESSMESSAGE (m) {
  /* M_received and M_sent store received and sent messages respectively */
  M_received = M_received ∪ {m}
  If m is a disclosure, add m to the local knowledge base L
  Apply the local negotiation strategy with parameters M_received, M_sent, and L,
  which returns a list of messages M
  /* send the messages to their intended recipients */
  if M is not empty then
    for every message k in M do
      Send k to its specified recipient
      Record k's sending time as the current time
      M_sent = M_sent ∪ {k}
}
```

Receive message

Process message

Negotiation

Send message

Basic Eager Strategy

1. BASICEAGERSTRATEGY ($M_{received}$, M_{sent}, L) {
2. Let P_{this} be the current peer
3. m = the latest message in $M_{received}$
4. Q_{sent} = set of disclosures P_{this} requested from others
5. $Q_{received}$ = set of disclosures others requested from P_{this}
6. $Q_{new} = \emptyset$
7. D_{sent} = set of disclosures P_{this} sent to others
8. $D_{received}$ = set of disclosures P_{this} received from others
9. $D_{new} = \emptyset$
10.
11. **if** m is a disclosure d **then**
12. /* Calculate new disclosures D_{new} that P_{this} will send to other parties */
13. $D_{unlocked}$ = all disclosures unlocked by d and other disclosures in $D_{received}$
14. $D_{new} = D_{unlocked} \cap Q_{received} - D_{sent}$
15. **else if** m is a request for disclosure d **then**
16. **if** d is already unlocked **then**
17. D_{new}=$\{d\}$
18. **else**
19. /* Calculate new disclosures Q_{new} that P_{this} will request from others */
20. $D_{relevant}$ = all relevant remote disclosures for d
21. $Q_{new} = D_{relevant} - D_{received} - Q_{sent}$
22.
23. Return the list of messages composed of disclosures in D_{new} and requests for disclosures in Q_{new}
24. }

Even BES considers the relevant credentials only !!!

Full Eager Strategy

- Use Ack. message to guarantee that negotiation will ends if not succeed
  - i.e. prevent deadlock such that several parties requests disclosures from each other
- Hard to understand without consulting reference [27] and proofs in reference [26]
  - skip…

Cautious Strategy

```
1.  CautiousStrategy (M_received, M_sent, L) {
2.    Let P_this be the current peer
3.    m = the latest message in M_received
4.
5.    if m is a request for a disclosure d then
6.      e = d
7.    else
8.      m must be a disclosure d or a denial !d.
9.      Let ?e be the latest request in M_received that has not been denied or disclosed,
        and for which d is relevant.
10.     if no such ?e exists then
11.       /* The originating request of the negotiation must be for d. If m is a dis-
             closure, the MTN has succeeded; otherwise, m is a denial message, and
             the MTN has failed. */
12.       Return ∅
13.   if e is already unlocked then
14.     Return {e}
15.   Let S be the set containing all remote disclosures f such that (1) f is relevant to
      e, (2) f has not been received by P_this, (3) if P_this has requested f, that request
      has been denied; and (4) P_this has not requested f since it received ?e
16.   if S is empty then
17.     /* There are no more disclosures that P_this can request to unlock e */
18.     Return { !e }
19.   Pick one disclosure g from S
20.   Return {?g}
21. }
```

e is either directly requested or a previous request that *d* is relevant for

In either case, the next step is to answer e

Cautious!!!

Strengths

- Models Multiparty Negotiation
- DARCL is novel and allows to specify source and destination of a disclosure
- The two strategies are well-designed: even the BES only discloses relevant credentials

Weaknesses

- ***Discussion…***

Weaknesses

- Policy defined by source and destination are too specific and ad-hoc in nature
- Parties in MTN have to choose the same strategy
  - otherwise their strategies are not complete
- How to choose one disclosure in CS?