Modelling and Analysing an Identity Federation Protocol: Federated Network Providers Scenario

Maurice ter Beek (FMT, ISTI–CNR, Pisa, Italy)

joint work with

Marinella Petrocchi (IIT–CNR, Pisa, Italy)

Corrado Moiso (Telecom Italia, Torino, Italy)

YR-SOC 2007, Leicester, UK
Outline of the talk

- Setting
- Identity federation protocols
- Telecom Italia’s network protocol for identity federation
- Modelling and analysis
  - Analysis approach
  - Crypto-CCS specification language
  - Formalisation of two scenarios of the network protocol
  - Analyses and results of a man-in-the-middle attack
- Conclusions and future work
Setting

• Formal modelling and analysis of security protocols is an active branch of computer security

• Many techniques proved successful (based on process algebras, authentication logic, type systems, etc.)

• We formally specify three user scenarios of a network protocol for identity federation proposed by Telecom Italia, at the same time adding primitives to assure basic security properties

• We then model check our specifications to test their correctness
Identity federation protocols

• Growing interest in defining telecommunication protocols that allow a user to access all services belonging to the same *circle of trust* with a (cross-domain) *single sign-on*

• Process of *identity federation*: federating an entity’s identity and allowing access to services without explicitly presenting one’s credentials time and again

• *Liberty Alliance*: consortium formed to define processes supporting the federation of identities

• Specifications make use of the XML-based *Security Assertion Markup Language* SAML
Security features

- Limit access to authenticated and authorized users
- Preserve privacy of users:
  - protect sensitive information (e.g. network addresses)
  - guarantee identities without explicitly discovering them
  - only disclose information related to the specific service for which access is requested (e.g. destination preferences if the service is a travel agency)
- (Optional) Grant users anonymous access to services (e.g. for temporary federations)
Federating identities example

- ABC airlines and XYZ car rental company decide to create a circle of trust
- Mary has accounts on both ABC’s and XYZ’s web sites
- She logs into ABC’s web site – "You may share (or federate) your ABC online identity with members of our affinity group, which includes XYZ"
- Mary likes the idea, so she gives her permission
- Mary goes to XYZ – "We see you’re logged into ABC’s web site. Would you like to link your XYZ online identity with your ABC online identity?" OK!

⇒ In the future, when Mary goes to either ABC’s or XYZ’s web site, she only needs to log into one to be automatically logged into the other.
Federated identity architecture

A circle of trust is a group of providers that have joined together to exchange authentication information.

Principal
• Customer
• Employee
• Company
• ...

Service Providers
• Web content
• Portal
• Merchant
• ...

Identity Provider
• Authentication
• Federation
• Profile
• ...

The principal has a defined local identity with more than one provider, and has the option to federate them.

The identity provider is the center of the authentication infrastructure. It is a trusted entity that maintains core attributes regarding the principal.

Service providers in the authentication domain offer complimentary services.
Some main features

- Authentication is delegated to an identity provider, allowing single sign-ons.
- A user token is a sequence of characters that identifies the user to each pair of parties in the circle of trust.
- User tokens are opaque, i.e. have meaning only for the two parties that federate their users’ identities.
- Problem: handle identity and authentication information of end users that access services on convergent networks through multiple telecommunication channels (e.g. ADSL, GPRS/UMTS, SMS).
The network protocol proposed by Telecom Italia @ ICIN’06

- is an identity federation protocol
- permits users to access services through different access networks (e.g., fixed and mobile)
- gives the network provider the role of identity provider, based on the idea that providers are in a privileged position to pass user information obtained within their security domain to the application level

⇒ Services thus rely on the authentication information provided by the access network
Token injector mechanism

- intercepts HTTP access requests
- (generates) and injects tokens
- forwards them to the applications
1. HTTP Request http://www.SP.com/register.html

2. Would you like to federate?
   - NO
   - 3. Local registration

4. Request of registration+federation

5. Verify authentication of Client on basis of IP address

6. a. IdP/TI generates opaque−id
   b. IdP/TI creates SAML Assertion with <AuthnStatement>

7. c. SAML Assertion may also contain <AttributeStatement>
   d. IdP/TI inserts SAML Assertion in SAML <Response>

   "Inject" SAML <Response> in the Request

8. HTTP Request (POST) http://www.SP.com/registerTravel.jsp + SAML <Response>

9. SP receives, in SAML <Response>, also the opaque−id

10. The following two situations may occur:
    Case 1. SP needs no further info and the UA directly accesses the service (step 15)
    Case 2. SP needs specific profile info from the service, which must be provided by the UA, via a "form"

11. HTTP Response (200−OK,"form")

12. UA fills in the "form"

13. HTTP Request (POST)

14. SP stores the received info

15. HTTP Response (200−OK,access.jsp)
Multiple access networks
1. Request of registration+federation
2. Search repository for token associated to U

   YES

   3. Token found?

   NO

   4. a. Verify authentication of Client on basis of IP address
   b. IdP/TI generates token (opaque-id)

   5. Send token

   Store token

6. IdP/TI creates SAML Assertion with <AuthnStatement>

7. c. SAML Assertion may also contain <AttributeStatement>
   d. IdP/TI inserts SAML Assertion in SAML <Response>

   "Inject" SAML <Response>
in the Request

8. HTTP Request (POST) http://www.SP.com/registerTravel.jsp + SAML <Response>

9. SP receives, in SAML <Response>, also the token

10. Case 1. SP needs no further info and the U
    directly accesses the service (step 15)
10. Case 2. SP needs specific profile info from the service,
    which must be provided by the U, via a "form"

11. HTTP Response (200−OK, "form")

12. U fills in the "form"

13. HTTP Request (POST) + SAML <Response>

14. SP stores the received info

15. HTTP Response (200−OK, access.jsp)
Analysis approach

• We specify the protocol in the process algebra Crypto-CCS, which is CCS plus some cryptographic primitives
• We specify the properties to be verified by logic formulae
• We add a Dolev-Yao-like intruder to the specification, whose behaviour is implicitly defined by the semantics of the language
• We verify a property by monitoring the intruder’s knowledge, which is the set of messages the intruder initially knows plus those received during computation
Crypto-CCS

- Set of processes communicating via message passing
- Inference system models possible operation on messages

\[ r = \frac{m_1 \cdots m_n}{m_0} \]

\[ S := S_1 \parallel S_2 | A \]
\[ A := 0 | p.A | [m_1 \cdots m_n \vdash_r x]A; A_1 \]
\[ p := c!m | c?x \]

- compound systems
- sequential agents
- prefix constructs
Informal semantics of Crypto-CCS

$c!m$ send message $m$ over channel $c$
$c?x$ receive message $m$ over channel $c$
0 do nothing
$p.A$ perform $p$ and then behave as $A$

$[m_1 \cdots m_n \vdash_r x] A; A_1$ inference construct

$S_1 \parallel S_2$ parallel composition plus synchronization

Example: $[m \quad pk_y^{-1} \vdash_{sign} x] A; 0$

A process that uses rule $sign$ to obtain a digitally signed message from plaintext message $m$ and private key $pk_y^{-1}$ and then behaves as $A$, or otherwise does nothing
An example inference system for public-key cryptography

\[
\begin{align*}
\frac{x}{y} & \quad \frac{y}{x} \quad \text{(pair)} \\
\frac{\text{Pair}(x, y)}{x} & \quad \text{(1st)} \\
\frac{\text{Pair}(x, y)}{y} & \quad \text{(2nd)} \\
\frac{x}{y} & \quad \frac{p_{k_y}^{-1} x}{\{x\} p_{k_y}^{-1}} \quad \frac{p_{k_y}}{x} \quad \text{(sign)} \\
\frac{x}{y} & \quad \frac{x}{\{x\} p_{k_y}^{-1}} \quad \text{(ver)} \\
\frac{x}{y} & \quad \frac{\text{KEY}}{\{x\} \text{KEY}} \quad \frac{\text{KEY}}{x} \quad \text{(enc)} \\
\frac{x}{y} & \quad \frac{x}{\text{KEY}} \quad \text{(dec)} \\
\frac{x}{y} & \quad \frac{\text{check}}{x} 
\end{align*}
\]
### Federated registration

There are three steps:

1. **user** \( U \) asks identity provider \( IdP \) and service provider \( SP \) to federate
   
   \[ c_0 \quad U \leftrightarrow IdP : r \]

2. **request** \( r \) intercepted by \( IdP \)

   \[ c_1 \quad IdP \leftrightarrow SP : \{ r, \text{SAML assertion} \}^{K_{IdP}^{-1}} \]

3. **\( SP \) grants/denies access to** \( U \)

   \[ c_2 \quad SP \leftrightarrow U : \{ \text{ok/ko} \}^{K_{SP}^{-1}} \]

1. **user** \( U \) asks identity provider \( IdP \) and service provider \( SP \) to federate
   
   \[ \quad \Rightarrow \text{authenticate } U \]

2. **request** \( r \) intercepted by \( IdP \)

   \[ \quad \Rightarrow \text{generate token } id_U \]

   \[ \quad \Rightarrow \text{assemble } \text{SAML assertion} \]

3. **\( SP \) grants/denies access to** \( U \)
A SAML assertion declares “Subj is authenticated”

\[ \{ \text{Subj, AuthStat, AttrStat} \}_{\text{KEY}} \]  (encrypted SAML assertion)

Subj token \( id_U \), univocally identifying \( U \)

AuthStat authentication statement, asserting \( U \) was authenticated (and the mechanism by which)

AttrStat attribute list of \( U \) plus nonce \( n_{IdP}^U \) to avoid replay attacks

\[ \{ r, \text{SAML assertion} \}_{K_{IdP}^{-1}} \]  (signed by \( IdP \) for authenticity)
receive SAML assertion + request
    verify signature,
    extract encryption,
    decrypt,
    extract pair: token + AuthStat,
    extract nonce,
    extract token,
    extract AuthStat,
    test correctness AuthStat,
    test freshness nonce,
    build pair to store,
    store token + nonce pair,
    prepare signature to
    grant access and stop
Federated network providers

$c_{MF} \quad FO \leftrightarrow MO$ assumed secure: share secret key $KEY_{FM}$

<table>
<thead>
<tr>
<th>$c_0$</th>
<th>$U \leftrightarrow MO : r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{MF}$</td>
<td>$MO \leftrightarrow FO : {id_U, U}<em>{KEY</em>{FM}}$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$MO \leftrightarrow SP : {r, SAML\ \text{assertion}}<em>{K</em>{MO}^{-1}}$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$SP \leftrightarrow U : {ok/ko}<em>{K</em>{SP}^{-1}}$</td>
</tr>
</tbody>
</table>

We slightly enrich network protocol presented @ ICIN’06:

When $FO/MO$ receives $r$ from $U$, search repository for $id_U$
- If found, then retrieve it and continue as usual
- Else, generate $id_U$ and send it to federated provider, where stored for other interactions between $U$ and $SP$
Crypto-CCS specification – MO

\[ MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \defs c_0?x_r.MO_1(x_r, n_U^{MO}, id_U, KEY_{FM}) \]

\[ MO_1(x_r, n_U^{MO}, id_U, KEY_{FM}) \defs [id_U \ U \vdash_{pair} (id_U, U)] \]

receive request

create pair,

encrypt pair,

send token to FO,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,

create pair,
A man-in-the-middle attack

Can intruder $X$ intercept (modify) a conversation between $MO$ and $SP$, without the latter being aware of this?

PROPERTY

“whenever $SP$ concludes the protocol apparently with $MO$, it was indeed the latter that executed the protocol”

Use two special actions in our Crypto-CCS specification:

- $commit(a,b)$: $a$ indeed finished the protocol with $b$
- $run(b,a)$: $a$ indeed started the protocol with $b$
Property

Does a computation exists such that:

- \textit{SP} is convinced to have finished talking with \textit{MO}, while in reality \textit{MO} never started talking with \textit{SP}
- \textit{FO} is convinced to have finished talking with \textit{MO}, while in reality \textit{MO} never started talking with \textit{FO}

\[\text{commit}(SP, MO) \land (\neg \text{run}(MO, SP))\]

\text{OR}

\[\text{commit}(FO, MO) \land (\neg \text{run}(MO, FO))\]
Input model checker

PaMoChSA v1.0 developed at IIT–CNR

- Specification file: mitm-2.exp
- Logic formula: 
  \[(commit(SP,MO) \text{ AND } (\text{NOT}\ run(MO,SP))) \text{ OR } (commit(FO,MO) \text{ AND } (\text{NOT}\ run(MO,FO)))\]
- Initial knowledge: \(\{pk_X, pk_X^{-1}, pk_{MO}, pk_{FO}, pk_{SP}\}\)
- Result: **No attack found**

(analogously for federated registration)
PaMoChSA’s graphical interface

You start a new elaboration, with
OPTIONS:
No Hide Channels,
Generation of Random Values,
Stored States

RESULT
No attack found
Conclusions

- We advocate the use of formal methods in the design phase of protocols so as to obtain well-defined protocols guaranteed to satisfy certain desirable properties.

- The results of our initial analyses strengthen our confidence in our formal specifications.

- In particular, these results lead us to believe that we correctly inserted digital signatures, encryption and nonces into the network protocol as originally proposed by Telecom Italia.
Future work

- Extend our analyses by considering:
  - more user scenarios
  - more security issues (e.g. unsubscription & anonymity)
- Presented paper at AICT’07 (3rd Advanced International Conference on Telecommunications, IEEE Computer Society) that covers the federated registration scenario
- Deal with quantitative extensions of formal methods and tools (such as probabilistic specification languages and stochastic model checkers)