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

Maurice H. ter Beek[1], Corrado Moiso[2], and Marinella Petrocchi[3]

[1] ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
[2] Telecom Italia, Via Reiss Romoli 274, 10148 Torino, Italy
[3] IIT–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

Abstract. We continue our work on modelling and analysing security issues of an identity federation protocol for convergent networks. This protocol was proposed by Telecom Italia as a solution to allow end users access to services on the web through different access networks, without explicitly providing any credentials, while the service providers can trust the user's identity information provided by the access networks and access some user data. As an intermediate step towards a full-blown formal security analysis of this protocol, we specify one specific user scenario in the process algebra Crypto-CCS and verify its vulnerability w.r.t. a man-in-the-middle attack with the model checker PaMoChSA.

1 Introduction

We continue work initiated in [1] by formally specifying one specific user scenario of the identity federation protocol for convergent networks introduced in [2] and by analysing the possibility of a man-in-the-middle attack. We use Crypto-CCS, a CCS-like process algebra with cryptographic primitives [3, 4], in combination with the Partial Model Checking Security Analyser PaMoChSA [5] developed by the Security group of IIT–CNR.

The protocol proposed in [2] permits end users to access services through different access networks (*e.g.* mobile and fixed ones), without explicitly providing any credentials, while the application level can trust the identity and authentication information provided by the access networks. As a result, service providers (SPs) identify a user by using the authentication procedure performed by the network provider. After identity federation, a single sign-on suffices for a user to access all services belonging to the same "circle of trust" of SPs while keeping personal data private. This is an advantage over introducing (and remembering) one's credentials time and again. The protocol may thus grant users anonymous access to services and at the same time allow the application level to limit access to authorised users. Moreover, without knowing who the user is, SPs can still obtain her/his location or age or charge her/his account.

In Section 2 we sketch our scenario. We then describe our analysis approach in Section 3, after which we formally specify our user scenario in Section 4. A small security analysis is presented in Section 5. Section 6 contains our conclusions.

* This work has been partially funded by the EU project SENSORIA (IST-2005-016004).

2 The Case of Federated Network Providers

The protocol of [2] uses a token injector mechanism to translate the identity and authentication information provided by a secure access network to the Internet application level (of a lower security level). The token injector thus plays the identity provider role described in the Liberty Alliance specifications [6].

We consider a user scenario with two network operators, a Mobile Operator MO and a Fixed Operator FO, both of which have implemented the token injector mechanism and have established a trusted relation with a SP, for instance a travel site. The user is a customer of both operators, has an active registration on the travel site and has already federated her/his account on the travel site with her/his profile on each of the operators (*i.e.* two federations have been activated: one between the SP and the MO accounts and one between the SP and the FO accounts). During the process of federation, the token injector generates an opaque-id (or pseudonym) for the user and sends it to the SP. This opaque-id is then stored on the repositories associated to the token injector and to the SP and it is exchanged in all communication between the operators and the SP, so as to identify the user in a secure way. Consequently, the user can access the travel site by mobile phone (GPRS) or by PC (ADSL) without introducing credentials. Instead, the network authentication is forwarded to the travel site, which identifies the user. Even though the SP does not know the user's mobile phone number, the opaque-id enables it to use the operators service (as SMS gateway) to nevertheless exchange or request information about the user. The architecture of this federated network providers scenario is sketched in Figure 1.

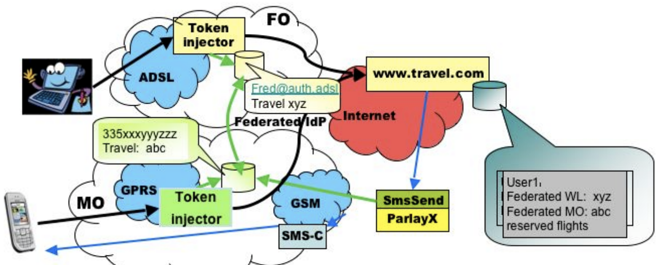

Fig. 1. Federated network providers scenario in convergent networks.

3 Modelling and Analysis Approach

We adopt the approach of [4] based on the observation that a protocol under analysis can be described as an open system in which some component has an unspecified behaviour (not fixed in advance). One then assumes that, regardless of this behaviour, the system works properly (*i.e.* satisfies a certain property). In our scenario, one can imagine a hostile adversary trying to achieve some kind of advantage w.r.t. the honest participants. Such an adversary is added to the specification of the protocol in the next section as a component with a behaviour defined only implicitly by the semantics of the specification language.

We assume the adversary to act in Dolev-Yao fashion [7] by using a set of message manipulating rules that model cryptographic functions. A message encrypted with the public key of one of the participants cannot be decrypted by anyone but the person who knows the corresponding private key. As is common, we adopt a black-box view of cryptography by assuming all cryptographic primitives involved in the network protocol to be perfect. To analyse whether a system works properly, at a certain point the adversary's knowledge is checked against a security property. If the intruder has come to know information it was not supposed to know, then the analysis has thus revealed an attack w.r.t. that property, *i.e.* a sequence of actions of the adversary that invalidates the property.

First, we fix some notation related to the security primitives we deal with. Sending message *msg* from sender A to receiver B over the ith communication channel c_i is denoted by $c_i\ A \mapsto B : msg$. We use the following security primitives:

$pk_i,\ pk_i^{-1}$	*public and private key of agent i*
$\{\_\}_{pk_i^{-1}}$	*message signed by agent i*
$\{\_\}_{pk_i}$	*message encrypted by public key of agent i*
$\{\_\}_{KEY}$	*message encrypted by symmetric key KEY*
n_j^i	*nonce related to j generated by i*

A *nonce* is a parameter that varies with time, *e.g.* a special marker intended to prevent the unauthorised replay or reproduction of a message.

A model defined in Crypto-CCS consists of a set of sequential agents able to communicate by exchanging messages. Inference systems model the possible operations on messages and therefore consist of a set of rules of the form $r = \frac{m_1 \cdots m_n}{m_0}$, with premises $m_1, \ldots, m_n$ and conclusion m_0. An instance of the application of rule r to closed messages $m_1, \ldots, m_n$ is denoted $m_1 \cdots m_n \vdash_r m_0$.

The control part of the language consists of compound systems and its terms are generated by the grammar (only constructs used below are presented):

$S := S_1 \parallel S_2 \mid A$	*compound systems*
$A := \mathbf{0} \mid p.A \mid [m_1 \cdots m_n \vdash_r x]A; A_1$	*sequential agents*
$p := c!m \mid c?x$	*prefix constructs*

where $m_1, \ldots, m_n, m$ are closed messages or variables, x is a variable and c is a channel. Informally, the Crypto-CCS semantics used below are:

– $c!m$: a message m sent over channel c;
– $c?x$: a message m received over channel c which replaces variable x;
– $\mathbf{0}$: a process that does nothing;
– $p.A$: a process that can perform an action p and then behave as A;
– $[m_1 \cdots m_n \vdash_r x]A; A_1$ (the inference construct): if (by applying an instance of rule r with premises $m_1, \ldots, m_n$) a message m can be inferred, then the process behaves as A (where m replaces x), otherwise it behaves as A_1.
– $S_1 \parallel S_2$: the parallel composition of S_1 and S_2, *i.e.* $S_1 \parallel S_2$ performs an action if either S_1 or S_2 does. It may perform a synchronisation or internal action, denoted by τ, whenever S_1 and S_2 can perform two complementary send and receive actions over the same channel.

The language is completely parameteric w.r.t. the inference system used. The inference system that is used below to model our scenario is shown in Figure 2.

$$\frac{x \quad y}{Pair(x,y)}\ (pair) \qquad \frac{x \quad pk_y^{-1}}{\{x\}_{pk_y^{-1}}}\ (sign) \qquad \frac{x \quad KEY}{\{x\}_{KEY}}\ (enc)$$

$$\frac{Pair(x,y)}{x}\ (1st) \qquad \frac{\{x\}_{KEY} \quad KEY}{x}\ (dec)$$

$$\frac{Pair(x,y)}{y}\ (2nd) \qquad \frac{\{x\}_{pk_y^{-1}} \quad pk_y}{x}\ (ver) \qquad \frac{x}{x}\ (check)$$

Fig. 2. Inference system for the federated network providers scenario.

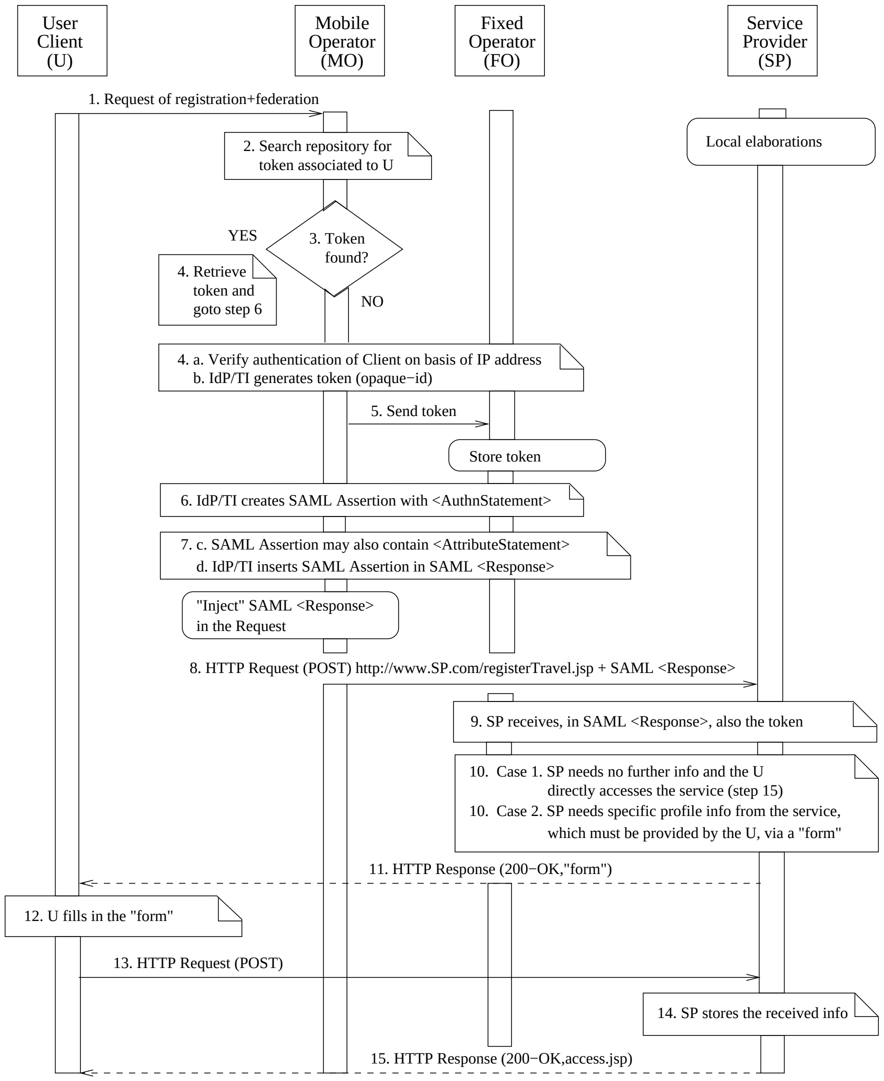
User Client (U)
Mobile Operator (MO)
Fixed Operator (FO)
Service Provider (SP)
1. Request of registration+federation
2. Search repository for token associated to U
Local elaborations
YES
3. Token found?
4. Retrieve token and goto step 6
NO
4. a. Verify authentication of Client on basis of IP address
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)
14. SP stores the received info
15. HTTP Response (200−OK,access.jsp)

Fig. 3. Message sequence chart of federated network providers.

Rule (*pair*) builds the pair of two messages x and y. Rules (*1st*) and (*2nd*) return the components of a pair. Rule (*sign*) digitally signs a message x by applying the secret key pk_y^{-1} of agent y. Rule (*ver*) verifies a digital signature $\{x\}_{pk_y^{-1}}$ by applying the public key pk_y of signer y. Rule (*enc*) encrypts a message x by applying the symmetric key KEY. Rule (*dec*) decrypts a message $\{x\}_{KEY}$ by applying the symmetric key KEY. Finally, rule (*check*) performs checks on the correctness of authentication statements and on the freshness of nonces.

4 Specification of Federated Network Providers Scenario

The federated network providers scenario that we formalise involves a user U, a service provider SP and two federated network providers: a fixed operator FO and a mobile operator MO. Our starting point is when U initiates the process of federated registration with SP through MO. From [2] we inherit the assumption that all communication between FO and MO is secure: we consider them to share a secret key KEY_{FM}. The process is lined out in detail in Figure 3.

SAML [8] is an XML standard to exchange information on authentication and authorisation data between security domains intended to implement mechanisms for single sign-ons. A *SAML assertion* declares a subject authenticated by a particular means at a particular time. For our purposes, it contains a field *Subj* with the token id_U identifying U, a field $AuthStat$ with an authentication statement asserting that U was authenticated (and the mechanism by which s/he was), and a field $AttrStat = \langle attr\,list, n_U^{IdP}\rangle$ with a list of attributes of U related to her/his service accesses and a nonce to avoid replay attacks.

To avoid dealing twice with the same process of token generation, we propose a formalisation that slightly enriches the procedure presented in [2, 6]: as soon as one of the two network providers receives a request from U, it searches its repository for a token already associated to U. If this token is found, then it is retrieved and the procedure continues as in the federated registration scenario. Otherwise it is generated and immediately sent to the other federated network provider, where it is stored for subsequent interactions between U and SP:

$$\begin{array}{lll}
c_0 & U \mapsto MO : & r \\
c_{MF} & MO \mapsto FO : & \{id_U, U\}_{KEY_{FM}} \\
c_1 & MO \mapsto SP : & \{r, SAML\ assertion\}_{K_{MO}^{-1}} \\
c_2 & SP \mapsto U \ : & \{ok/ko\}_{K_{SP}^{-1}}
\end{array}$$

Next we specify this federated network providers scenario in Crypto-CCS. This specification is more expressive than the one in standard notation given above, because all operations and security checks on the various messages are explicitly modelled. Each process is parameterised by the terms or variables it has in its knowledge (from the beginning or because it received them earlier).

$U_0(r) \doteq c_0!r.c_2?x_{sign}.$ *send request, receive signature,*
  $[x_{sign} \quad K_{SP} \vdash_{ver} x_{acc}].\mathbf{0}$ *verify signature and stop*
$SP_0(0) \doteq c_1?x_m.SP_1(x_m)$ *receive SAML assertion + request and goto next state*

$SP_1(x_m) \doteq [x_m \quad k_{IdP} \vdash_{ver} x_p]$	*verify signature,*
$\quad [x_p \vdash_{2nd} x_{enc}]$	*extract encryption,*
$\quad [x_{enc} \quad KEY \vdash_{dec} x_{dec}]$	*decrypt,*
$\quad [x_{dec} \vdash_{1st} x_{pair}]$	*extract pair: token + AuthStat,*
$\quad [x_{dec} \vdash_{2nd} x_{n_U^{IdP}}]$	*extract nonce,*
$\quad [x_{pair} \vdash_{1st} x_{id_U}]$	*extract token,*
$\quad [x_{pair} \vdash_{2nd} x_{auth}]$	*extract AuthStat,*
$\quad [x_{auth} \vdash_{check} x_{auth}]$	*test correctness AuthStat,*
$\quad [x_{n_U^{IdP}} \vdash_{check} x_{n_U^{IdP}}]$	*test freshness nonce,*
$\quad [x_{id_U} \quad x_{n_U^{IdP}} \vdash_{pair} (x_{id_U}, x_{n_U^{IdP}})]$	*build pair to store,*
$\quad c_S!(x_{id_U}, x_{n_U^{IdP}})$	*store token + nonce pair,*
$\quad [access \quad k_{SP}^{-1} \vdash_{sign} x_{sign}]$	*prepare signature to*
$\quad c_2!x_{sign}.\mathbf{0}$	*grant access and stop*
$MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \doteq c_0?x_r.$	*receive request and*
$\quad MO_1(x_r, n_U^{MO}, id_U, KEY_{FM})$	*goto next state*
$MO_1(x_r, n_U^{MO}, id_U, KEY_{FM}) \doteq [id_U \quad U \vdash_{pair} (id_U, U)]$	*create pair,*
$\quad [(id_U, U) \quad KEY_{FM} \vdash_{enc} \{(id_U, U)\}_{KEY_{FM}}]$	*encrypt pair,*
$\quad c_{MF}!\{(id_U, U)\}_{KEY_{FM}}.$	*send token to FO,*
$\quad [id_U \quad auth \vdash_{pair} (id_U, auth)]$	*create pair,*
$\quad [(id_U, auth) \quad n_U^{MO} \vdash_{pair} ((id_U, auth), n_U^{MO})]$	*create pair,*
$\quad [((id_U, auth), n_U^{MO}) \quad KEY \vdash_{enc} \{((id_U, auth), n_U^{MO})\}_{KEY}]$	*encrypt pair,*
$\quad [x_r \ \{((id_U, auth), n_U^{MO})\}_{KEY} \vdash_{pair} (x_r, \{((id_U, auth), n_U^{MO})\}_{KEY})]$	*create pair,*
$\quad [(x_r, \{((id_U, auth), n_U^{MO})\}_{KEY}) \quad k_{MO}^{-1} \vdash_{sign} x_{sign}]$	*sign pair,*
$\quad c_1!x_{sign}.\mathbf{0}$	*send SAML assertion + request and stop*
$FO_0(KEY_{FM}) \doteq c_{MF}?x_{enc}$	*receive encryption,*
$\quad [x_{enc} \quad KEY_{FM} \vdash_{dec} x_{dec}]$	*retrieve decryption,*
$\quad c_S!x_{dec}.\mathbf{0}$	*store token + identity pair and stop*

For the sake of readability, we did not fully spell out the digital signatures (*i.e.* we just applied the private key to the message to be signed). Moreover, we assumed a direct communication channel between *SP* and *U* in order to grant/deny access, while in reality all communication passes through *IdP*. The whole process is described by $U_0(r) \parallel MO_0(0, n_U^{MO}, id_U, KEY_{FM}) \parallel FO_0(KEY_{FM}) \parallel SP_0(0)$.

5 Verification of a Security Property

As an intermediate step towards a full-blown security analysis of the protocol, we verified the vulnerability of our federated network providers scenario w.r.t. a man-in-the-middle attack. Such an attack is an adversary's attempt to intercept and modify messages between two trusted participants, in such a way that neither participant is able to find out that their communication channel has been compromised. We used model checking to verify whether the specification of the insecure channel between *MO* and *SP* can withstand such an attack. In [1] we already verified that the insecure channel between *IdP* and *SP* can.

The analysis boils down to verifying the following property: whenever *SP* concludes the protocol apparently with *MO*, it was indeed *MO* that executed it. To this aim, we introduced two special actions in the specification: *commit*(*SP,MO*) and *run*(*MO,SP*). The former represents the fact that *SP* has indeed terminated the protocol with *MO*, while the latter represents the fact that *MO* indeed started communicating with *SP*. We then translated the property into requiring *run*(*MO,SP*) to always precede *commit*(*SP,MO*). We did the same to test for possible misbehaviour or interceptions of communications between *MO* and *FO*, by introducing the actions *commit*(*FO,MO*) and *run*(*MO,FO*) in the specification.

We used the model checker PaMoChSA v1.0 [5] to verify this. We considered an adversary X and set its initial knowledge to the set of public messages that it knows at the start of the protocol, *i.e.* the public keys of *MO*, *FO* and *SP* and its own public and private key denoted by pk_X and pk_X^{-1}. With as input the specification, the logic formula (*commit*(*SP,MO*) AND (NOT *run*(*MO,SP*))) OR (*commit*(*FO,MO*) AND (NOT *run*(*MO,FO*))) and the intruder's initial knowledge $\{pk_X, pk_X^{-1}, pk_{MO}, pk_{FO}, pk_{SP}\}$, the result of the analysis was **No attack found**.

To verify the logic formula specified above, the tool set out to find a run of the protocol with the following characteristic: At the end of the run, the adversary either knows message *commit*(*SP,MO*), but it does not know message *run*(*MO,SP*) (*i.e. SP* is convinced to have finished talking with *MO*, while in reality *MO* has never started talking with *SP*), or it knows message *commit*(*FO,MO*), but it does not know message *run*(*MO,FO*) (*i.e. FO* is convinced to have finished talking with *MO*, while in reality *MO* has never started talking with *FO*). Since the tool did not find such a run we conclude that, at the conceptual level, the network protocol is correct w.r.t. the analysed security property.

6 Conclusion

The result of the security analysis presented above, together with the one presented in [1], strengthens our confidence in the formal specifications of the scenarios presented in these two papers. In particular, it leads us to believe that we correctly inserted digital signatures, encryption and nonces into the protocol.

References

1. ter Beek, M.H., Moiso, C., Petrocchi, M.: Towards Security Analyses of an Identity Federation Protocol for Web Services in Convergent Networks. In: Proc. AICT'07, IEEE Computer Society (2007)
2. Bonifati, M., De Lutiis, P., Moiso, C., Morello, E., Sarchi, L.: Identity Federation for Services in Convergent Networks. In: Proc. ICIN'06. (2006) 109–114
3. Focardi, R., Martinelli, F.: A uniform approach for the definition of security properties. In: Proc. FM'99. Volume 1708 of LNCS., Springer (1999) 794–813
4. Martinelli, F.: Analysis of security protocols as open systems. Theoretical Computer Science **290**(1) (2003) 1057–1106
5. IIT–CNR: Partial Model Checking Security Analyzer PaMoChSA v1.0 (2007)
6. T. Wason et al.: Liberty ID-FF Architecture Overview v1.2 (2005)
7. Dolev, D., Yao, A.: On the Security of Public Key Protocols. IEEE Transactions on Information Theory **29**(2) (1983) 198–208
8. OASIS Security Services: Security Assertion Markup Language SAML v2.0 (2005)