An intruder model for trust negotiation

Philippe Balbiani, Yannick Chevalier, Marwa El Houri

An Intruder Model for Trust Negotiation

Philippe Balbiani, Yannick Chevalier and Marwa El Houri
CNRS-Université de Toulouse
Institut de recherche en informatique de Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 9, France

***Abstract*—In a distributed environment, and more specially in service oriented architectures, the entities interacting one with another rely on credentials to decide whether an action they are told to perform is permitted. These credentials are exchanged within trust negotiation sessions during which the participating entities build up trust by communicating certificates to trusted peers.**

Dolev and Yao have introduced a notion of symbolic intruder to represent the capacities of a malicious agent trying to attack a cryptographically secured communication protocol. We present in this paper an adaptation of that intruder that retains the same deductive capabilities but is specialized for the analysis of the exchanges during a trust negotiation session. In particular this permits us to analyze the security of a distributed access control policy *w.r.t.* a malicious insider.

I. Introduction

In a distributed environment, entities communicate with one another in order to execute complex tasks. This is the case for example in the modeling of business processes. The communication consists in sending and receiving credentials that are used to authenticate and regulate access to an entity's functionalities. In real life all entities are not necessarily honest, and thus communication is not necessarily secured.

In this paper, we consider that entities negotiate credentials one with another. We define entities as an abstraction of stateful services with a repository and a set of trust negotiation rules. The repository contains the set of credentials known to the entity and the trust negotiation rules define the negotiation strategy of the entity concerning the sending and receiving of credentials from other entities. We introduce channels that represent the reliability of the point-to-point medium on which the credentials are transported. In this context, we consider the problem of the existence of a malicious entity in the environment that can intervene during a trust negotiation session. As in [5] we have taken the pessimistic approach of assuming that a dishonest entity can control all the traffic on unsecured channels, *i.e.* that it can inspect, block or re-order legitimate messages, and can itself inject new messages with a forged header so that they appear to originate from the entity itself or any other honest one. These new messages are built from pieces of messages previously read, or from the initial set of objects in the intruder entity. Finally the intruder can compose, decompose, encrypt and decrypt messages in case he knows the keys.

Our goal in this paper is to analyze the robustness, *w.r.t.* standard confidentiality and authentication goals, of a secured trust negotiation setting. We consider a trust negotiation construct with secure channels and cryptography, our objective being to assess whether an active malicious insider may acquire undue knowledge or impersonate a honest entity.

Related Works: We have built upon a large collection of works on cryptographic protocols. Beside the seminal paper of Dolev and Yao [5], one could cite different papers taking into account channels either on their own [10] or in conjunction with a symbolic intruder [1], [11]. Our notion of *authenticity* originates from [9], while the notion of confidentiality is the classic *weak secrecy* one. The attack on Google's implementation of SAML Single Sign-On (SSO) was presented in [2], and we have merely translated it into a more natural trust negotiation setting. Though we have tried to make this paper self-contained, we have preferred to hide some formal details *e.g.* on rewriting and equational theories that may not interest the reader. We refer to [4], [12] for a gentle introduction to the main concepts.

The framework we consider is based on *Automated Trust Negotiation* [13], from which we have borrowed the eager strategy. Considering secured transport protocols or cryptographic primitives is not new [7], [14], [8] but these works do not consider an active malicious insider.

Outline: We present the basics of the language in Section II, and our modeling of secure channels in Section III. We introduce our intruder model in Section IV, and the adaptation of the trust negotiation algorithm to take it into account in Section V. We present in Section VI the security properties we analyze. In Section VII we present a modelling for Google's SAML SSO implementation and conclude in Section VIII.

II. Syntax

A. Basic concepts

In this section we define the syntax that shall be used to represent a trust negotiation infrastructure. We use the notions of *objects* and *values* to represent the elements of our framework. We let Val be a countable set of values (with typical members denoted $v, v', \ldots$), Att be a finite set of attributes (with typical members denoted $a,\ a',\ \ldots$) and $\mathcal{K}$ be a finite set of public and private keys (with typical

members denoted $k,\ k', k^{-1}, k'^{-1},\ \ldots$). We assume that $Att \cap \mathcal{K} = \emptyset$. We let $VarVal$ be a countable infinite set of *variables for values* (with typical members denoted $x,\ y,\ \ldots$).

An *object* O is a partial function $O : Att \to Val$ and we denote by $\mathcal{O}$ the set of all objects. Given an object O, we denote $dom(O)$ its domain.

Given two objects O_1 and O_2 we call the *update of* O_1 *by* O_2, and denote $O_1 \leftarrow\!\!+ O_2$ the object O of domain $dom(O_1) \cup dom(O_2)$ such that for $x \in dom(O)$ we have $O(x) = O_2(x)$ if $x \in dom(O_2)$ and $O(x) = O_1(x)$ otherwise.

Also, we define the domain restriction operation $\setminus$ that, given an object O and a set of attributes A, computes $O \setminus A$, the restriction of O to $dom(O) \setminus A$.

We write $O.a$ to denote $O(a)$ and we use the notation $O.a = \perp$ to express that a is not in the domain of O. And we define the *empty object* ε such that $dom(\varepsilon) = \emptyset$.

We let $VarObj$ be a countably infinite set of variables called *variables for objects* (with typical members denoted $X,\ Y,\ \ldots$).

Signed objects: In order to simplify notations we also consider in this paper only the case of cryptographic operations with public and private keys. We model both the encryption and signature operations with a single binary symbol f.

The set of *signed objects* is denoted $\varphi(\mathcal{O})$ and is the smallest set that contains $\mathcal{O} \cup VarObj$ and such that, if $O \in \varphi(\mathcal{O})$ and $k, k^{-1} \in \mathcal{K}$ then:

- $f(O, k)$ is in $\varphi(\mathcal{O})$ and denotes either the encryption of O with the encryption key k or the verification of the signature O with the validation key k;
- $f(O, k^{-1})$ is in $\varphi(\mathcal{O})$ and denotes either the signature of O with the decryption key k^{-1} or the decryption of O with signature key k^{-1}.

For soundness of analysis we assume that in the model we consider there are disjoint sets of key pairs for encryption and signature, and that the nature of an operation (*e.g.* encryption or validation) is clear given the key employed. The distinction between these two kinds of key pairs is however irrelevant in terms of possible symbolic operations, hence our unique set $\mathcal{K}$ of keys.

In order to model the relative properties of decryption *w.r.t.* encryption and validation *w.r.t.* signature we consider $\varphi(\mathcal{O})$ *modulo* the following two equations:

$$\begin{cases} f(f(O, k), k^{-1}) &= O \\ f(f(O, k^{-1}), k) &= O \end{cases}$$

It is straightforward to see that these equations define a convergent rewriting system on $\varphi(\mathcal{O})$, and thus that signed objects can be put in a *normal form* in which none of the above equations can be applied from left to right. In the rest of this paper, we consider that all signed objects without variables are in normal form.

A signed object in normal form without variables is *clear* iff it is not of the form $f(T, k)$ or $f(T, k^{-1})$ for some signed object without variables T. The notation $O.a$ is not defined for unclear objects.

A *term* is denoted t and is either a value, a variable for value, or an expression $X.a$ with $X \in \varphi(\mathcal{O})$.

Interpretation: An interpretation function for variables is a function I that associates to every variable $X \in VarObj$ (resp. $x \in VarVal$) an object $O \in \mathcal{O}$ (resp. a value $v \in Val$). It is extended on signed objects and terms as follows. We let $I(O) = O$ if $O \in \mathcal{O}$ and $I(v) = v$ if $v \in Val$. If $I(X) \in \mathcal{O}$ we define $I(X.a) = I(X).a$, and $I(X.a)$ is undefined otherwise. Finally we let $I(f(T, k)) = f(I(T), k)$ and $I(f(T, k^{-1})) = f(I(T), k^{-1})$.

Entity and state: An *entity* e_t is a set of trust negotiation rules (defined below). We denote by $\mathcal{E}$ a finite set of entities. To each entity e_t we associate a value $t \in Val$ which is its unique identifier and a *security state* s_t made up of a *repository* denoted by Rep_t, *i.e.* a set of objects, and an interpretation function for variables. The set of security states for entities in $\mathcal{E}$ is denoted by $\mathcal{S}$.

B. Trust negotiation policy

A trust negotiation policy consists of a set of trust negotiation rules. A trust negotiation rule gives the conditions to be satisfied in order to send (*put*) an object during a trust negotiation session. It is of the form:

$$put(T, t_1, t_2) \leftarrow body$$

where T is a signed object, t_1 (resp. t_2) is a term representing the apparent sender (resp. intended receiver). For honest entities the apparent sender is the entity itself, which is not always the case when one considers a malicious entity.

The predicate $put(T, t_1, t_2)$ allows the disclosure of the signed object T to an entity t_2 (as being sent by t_1) whenever the conditions in the body of the rule are satisfied.

Body of rules: The syntax of the body of the trust negotiation rules is defined by:

$$body{:=}\top \mid Test \mid body \wedge body \mid body \vee body$$
$$Test{:=}has(T) \mid get(T, t) \mid t_1 = t_2 \mid t_1 \neq t_2$$

where T is a signed object and t, t_1, t_2 are terms, with the intended meaning:

- $has(T)$ is true if the signed object T is in the repository of the entity;
- $t_1 = t_2$ (resp. $t_1 \neq t_2$) is true if t_1 and t_2 are equal (resp. not equal);
- $get(T, t)$ is true if the signed object T can be received during trust negotiation from an entity who seems to be e_t.

Note that while the predicate $put(T, t_1, t_2)$ represents the permission for the *sending entity* to communicate a

certificate T, the predicate $get(T, t)$ denotes the capacity of the *receiving entity* to accept the communicated certificate T. The trust negotiation semantics will be explicitly defined in Section V.

Trust policy of honest vs. malicious entities: In a perfect world, entities only send messages using their own identity (i.e. the apparent sender is the same as the real sender). However the presence of a malicious entity induces forged trust negotiation objects sent by the intruder impersonating a honest entity. Accordingly, we impose that a trust negotiation rule:

$$put(T, i, j) \leftarrow body$$

can only belong to an entity e_k if $k = i$ or e_k is malicious. We will introduce in Section III further restrictions according to the type of the communication channels.

Example II.1. In our running example there are three honest entities e_c, e_{sp} and e_{idp} denoting respectively a client, a service provider and an identity provider. A fourth entity e_{int} is malicious. To denote that the client is willing to forward certificates from the identity provider to the service provider, e_c contains the rule:

$$put(X, c, sp) \leftarrow get(X, idp) \wedge X.\textbf{respondent} = idp$$

We note that in the above example, the client has no way to ensure that the object X was actually issued by the identity provider. To this end we present in the next section a modeling of the properties provided by such channels. This will allow us to express secure communication on the one hand, and model the behavior of the intruder in the presence of secure communication on the other hand.

III. Secure channels

We assume that the exchange of objects within a trust negotiation session is done via communication protocols between the communicating entities. In this paper we abstract away the actual protocols and consider only the security properties they guarantee, among authenticity, confidentiality, and unblockability. One can refer to [1] and [11] for the formalization of cryptographic protocols into properties of communication channels. In this section we model these aspects with respect to the objects exchange during the trust negotiation mechanism and to the intruder.

A. Security properties of channels

Suppose two entities e_s and e_r want to exchange objects in a trust negotiation session. We wish to specify security properties for such a communication depending on the communication channels used by these entities in order to send/receive an object O. Formally, a channel is confidential if its output is exclusively accessible to a specific receiver, it is authentic if its input is exclusively accessible to a specific sender, and it is *unblockable* if the intruder cannot prevent a message sent on this channel to reach its destination. Otherwise the channel is said to be *blockable*.

For example, if an entity e_r receives the object O on an *authentic* channel then e_r will know who is the real sender of the object O. On the other hand if e_s sends O on a *confidential* channel then e_s will know that only the intended receiver will receive the object. Also two entities living on the same computer system will employ unblockable channels if the intruder does not have access to that computer system.

B. Extending the syntax

We extend the definition of the put and get predicates presented in the previous section to take into account the different kinds of communication by indexing it with a set of tags $A \subseteq \{auth, conf, unblock\}$.

- If $auth \in A$, then $put_A(T, t_1, t_2)$ denotes that the object T is sent on behalf of the apparent sender e_{t_1} to the entity e_{t_2} through a communication channel satisfying the authenticity property for the sending entity. Note that in such a case e_{t_1} is also the real sender of T.
- If $conf \in A$ then $put_A(T, t_1, t_2)$ denotes that the object T is sent on behalf of the apparent sender e_{t_1} to the entity e_{t_2} through a communication channel satisfying the confidentiality property for e_{t_2}. In such a case e_{t_2} is the unique receiver of that object.
- If $unblock \in A$ then $put_A(T, t_1, t_2)$ denotes that the object T is sent on behalf of the apparent sender e_{t_1} to the entity e_{t_2} through a communication channel on which objects cannot be blocked or delayed by the intruder.

The predicate $get_A(T, t)$ is defined in the same manner. Note that the *indexing* of the put and get predicates by the set A of tags allows the specification of the communication types in our framework, as will be explained in Section V.

We define an *available object* to be a 5-tuple (O, rs, s, r, A) where $O \in \varphi(\mathcal{O})$ is a signed object, $rs, s, r \in Val$ are values denoting the real sender, the apparent sender, and the intended receiver respectively and A is a set of tags from $\{auth, conf, unblock\}$. For example, if $auth \in A$ then (M, s, s, r, A) denotes an available object sent on an authentic communication channel between e_s and e_r.

Example III.1. In Google's implementation of SAML SSO, the certificates are sent by the client over a confidential channel to the service provider, and they are accepted by the client only on a confidential and authentic channel from the identity provider. To reflect this policy the rule of Ex. II.1 is now written:

$$put_{\{conf\}}(X, c, sp) \leftarrow \\ get_{\{auth,conf\}}(X, idp) \wedge X.\textbf{respondent} = idp$$

In the following section we present the actions that can be done by the intruder on objects that he knows or gained access of.

IV. Intruder model

In this section we describe the behavior of the intruder with respect to signed objects. That is, we assume that the intruder can encrypt, decrypt, sign or create objects depending on his knowledge on the one hand, and the security of a *known* object on the other hand. Note that a *known* object is an object that the intruder either has in his repository at the beginning of the trust negotiation session, or has acquired (legitimately or by listening to the network) during the previous rounds of negotiation.

A. Entities and keys

We assume the existence of a secure public key infrastructure, and as a consequence that each entity $e_i \in \mathcal{E}$ possesses couples of a public and private key denoted by k_i and k_i^{-1}, respectively. Recall that if an object O is protected by the public key k_i of e_i, we say the object is *encrypted for entity* e_i and denote it by $f(O, k_i)$. Similarly if an object O is protected by the private key k_i^{-1} of e_i, we say the object is *signed by* e_i and denote it by $f(O, k_i^{-1})$. In the rest of this section we model the behavior of the intruder with respect to the cryptographic primitives protecting the objects.

B. Trust negotiation policy of the intruder

In this section we present the possible actions of the intruder on objects available to him. From now on we assume int is the identifier of a malicious entity, and that honest entities have their identifier in the set $\{1, \ldots, n\}$.

Emission of a signed object: An object X is *available* to the intruder if the formula $get_A(X, l) \vee has(X)$ is true. We assume that the intruder may send any available signed object, and model this assumption with all the rules Send(A, A', i) (see Fig. 1) with $A, A' \subseteq \{auth, conf, unblock\}$ such that if $auth \in A'$ then i is the constant int. Otherwise i denotes a variable. An important special case is the sending by the intruder of objects to himself, as these self-communications permit us to embed the least fixpoint computation of the intruder's knowledge into the trust negotiation mechanism.

Cryptographic operations: Given our assumption on the existence of a PKI, we suppose that a key is available to the intruder if it is contained in its repository. Such keys can be employed to encrypt or sign objects. Since the intruder sends to himself all the available objects, these cryptographic operations are modeled by the set of rules Apply (see Fig. 1).

Management of known values: We assume that the intruder knows all the attributes. However for security analysis it is essential that the intruder does not know all the possible values. Let us now describe how values are managed.

- **Creation of a new object:** the intruder may create new, empty objects at will, as reflected by the rule Empty of Fig. 1;
- **Attribute extraction:** for each attribute name $a \in Att$ we have a rule Extract(a) that extracts the value of this attribute in a received clear object. We employ the attribute kn (knowledge) to store these values into objects;
- **Attribute update:** the intruder can update each attribute a of a clear object by a value he knows. This is modeled for all $a \in Att$ by a rule Update(a) described in Fig. 1;
- **Attribute deletion:** finally, and for any attribute $a \in Att$ the intruder may delete an attribute a from a clear object that he possesses. This is modeled by the rules delete(a) in Fig. 1.

V. Trust negotiation semantics

So far, we presented the modeling of communication by expressing the security properties of communication channels (see Sect. VI) and cryptography (see Sect. IV). In this section we give the trust negotiation semantics for both honest entities and the intruder. Note that while honest entities are assumed to abide by their trust negotiation policy, the intruder can receive objects destined to others in a non-confidential communication and can send objects to others pretending to be someone else in a non-authentic communication.

A. How entities receive available objects

With the presence of an intruder, the real sender and real receiver of a negotiated object may differ from the intended one. We illustrate this by the evaluation semantics for the $put(T, t_1, t_2)$ and $get(T, t)$ predicates in the trust negotiation policy of the honest entities on the one hand and the intruder on the other hand.

Trust negotiation session: We define a trust negotiation session to be a sequence of trust negotiation rounds. A trust negotiation session terminates when there are no more objects that can be negotiated, in which case the least fixed point is established. In this paper we model eager trust negotiation. That is we assume honest entities send all the objects that satisfy their trust negotiation policy during a trust negotiation session. However we assume that the intruder can perform a non-deterministic choice on the objects that he actually sends from the set of objects that satisfy his trust negotiation policy.

Adequate available objects: We have assumed that honest entities did not impersonate other entities nor listened to communications directed to others. Informally we say that an available object abiding by these rules is adequate. Formally let $\mathcal{A} = (O, x, y, z, A)$ be an available object and assume $i \in \{1, \ldots, n, int\}$. We say that $\mathcal{A}$ is *adequate* for

$$
\begin{array}{rrcl}
\text{Send}(A, A', i): & put_{A'}(X, i, x) & \leftarrow & get_A(X, y) \vee has(X) \\
\text{Apply}: & put_{\emptyset}(f(X, k), int, int) & \leftarrow & get_{\emptyset}(X, int) \wedge has(Y) \wedge Y.key = k \\
\text{Empty}: & put_{\emptyset}(\epsilon, int, int) & \leftarrow & \top \\
\text{Extract}(a): & put_{\emptyset}(X, int, int) & \leftarrow & get_{\emptyset}(Y, int) \wedge Y.a = x \wedge X = \epsilon \leftarrow\!\!+ (kn, x) \\
\text{Update}(a): & put_{\emptyset}(X, int, int) & \leftarrow & get_{\emptyset}(Y, int) \wedge get_{\emptyset}(Z, int) \wedge \ Z.kn = x \\
 & & & \wedge \ X = Y \leftarrow\!\!+ (a, x) \\
\text{Delete}(a): & put_{\emptyset}(X, int, int) & \leftarrow & get_A(Y, l) \wedge X = Y \setminus \{a\}
\end{array}
$$

Figure 1. Trust Negotiation rules for the intruder

i and denote $ad(\mathcal{A}, i)$ iff:

$$
\begin{cases}
x \in \{y, int\} \& (auth \in A \Rightarrow x = y) & \text{(send restrictions)} \\
i \in \{z, int\} \ \& (conf \in A \Rightarrow z = i) & \text{(receive restrictions)}
\end{cases}
$$

Let us explain the role of the conditions:

- $x \in \{y, int\}$ and $i \in \{z, int\}$ enforce respectively the facts that only the intruder sends objects as someone else or listens to objects sent to others;
- $auth \in A \Rightarrow x = y$ enforces the fact that the real and apparent senders must be equal for an available object transported on an authentic channel;
- $conf \in A \Rightarrow y = i$ enforces the fact that an available object transported on a confidential channel can only be listened by the intended receiver.

Trust negotiation round: A trust negotiation round is an $(n+1)$-tuple of sets of *available objects*

$$\Sigma_k = (\Sigma^k_{\mathcal{S},e_1} \dots \Sigma^k_{\mathcal{S},e_n}, \Sigma^k_{\mathcal{S},e_{int}})$$

where $\Sigma^k_{\mathcal{S},e_j}$ denotes the set of *adequate available objects* for the entity e_j at the beginning of round k. The tuples $(\Sigma_k)_{k \geq 0}$ are defined inductively as follows:

- **base case:** $\Sigma^0_{\mathcal{S},e_1} = \dots = \Sigma^0_{\mathcal{S},e_n} = \Sigma^0_{\mathcal{S},e_{int}} = \emptyset$;
- **induction:** Let $i \in \{1, \dots, n, int\}$ be an entity, $k \geq 0$. Suppose by induction that $\Sigma^k_{\mathcal{S},e_i}$ has already been defined. Furthermore assume that the set $\Omega^k_{\mathcal{S},e_i}$ of available objects acquired by i during the k-th trust negotiation step is defined. We define $\Sigma^{k+1}_{\mathcal{S},e_i} = \Sigma^k_{\mathcal{S},e_i} \cup \Omega^k_{\mathcal{S},e_i}$.

In the remaining of this section we present how $\Omega^k_{\mathcal{S},e_i}$ is computed given Σ_k.

B. What entities can send

Let $\Sigma^k_{\mathcal{S},e_i}$ denote the set of objects available for an entity e_i at the beginning of round k. We say that a signed object $O \in \varphi(\mathcal{O})$ can be sent by entity e_i to entity e_j at step k, and write $\mathcal{S}, e_i \models_k put_A(O, i, j)$, iff $put_A(O, i, j) \leftarrow body$ is a ground instance of a rule in the trust negotiation policy of an entity e_l such that:

$$\mathcal{S}, e_l \models_k body$$

To evaluate this formula we say $\mathcal{S}, e_i \models_k body$ iff in the rule instance:

- $body$ is $has(O')$ and $O' \in Rep_i$
- $body$ is $t = t$
- $body$ is $t \neq t'$ and t and t' are different
- $body$ is $get_{A'}(O', j)$ and there exists $y, z \in \{1, \dots, n, int\}$ and A'' such that $(O', y, j, z, A'') \in \Sigma^k_{\mathcal{S},e_i}$ and $A' \subseteq A''$ and $ad((O', y, j, z, A''), i)$.
- $body$ is $body_1 \wedge body_2$ and $\mathcal{S}, e_i \models_k body_1$ and $\mathcal{S}, e_i \models_k body_2$
- $body$ is $body_1 \vee body_2$ and $\mathcal{S}, e_i \models_k body_1$ or $\mathcal{S}, e_i \models_k body_2$.

The adequacy criterion in the fourth point implies in particular that honest entities only receive objects sent to them ($i = z$) and send objects as being themselves ($y = j$). Also, if i is the intruder and $i \neq z$ then the communication channel must be non-confidential and if $y \neq j$ the communication channel must be non-authentic.

C. Computing the set of available objects

The set of available objects for entities computed at round k depends on the set of objects sent by honest entities, the set of objects sent by the intruder as himself or as another entity and the set of objects blocked by the intruder.

Let $\Pi^k_{\mathcal{S},e_i}$ be the maximal set of objects that are available to e_i taking the assumption that the intruder sends everything he can and blocks nothing. By definition we have:

$$
\begin{array}{rcl}
\Pi^k_{\mathcal{S},e_i} & = & \{(O, x, j, y, A) : \mathcal{S}, e_x \models_k put_A(O, j, y) \\
 & & \text{with } x, j, y \in \{1, \dots, n, int\}, \text{and} \\
 & & ad((O, x, j, y, A), i)\}
\end{array}
$$

Let $\Lambda^k_{\mathcal{S},e_i}$ be the minimal set of objects that are available to e_i taking the assumption that the intruder does not send

anything and blocks all objects that can be blocked. By definition we have:

$$\Lambda^k_{\mathcal{S},e_i} = \{(O,j,j,i,A) \quad : \quad \mathcal{S}, e_x \models_k put_A(O,j,y), \\ j \neq int \text{ and } unblock \in A\}$$

We define the set of available objects for an entity e_i at step k to be a set $\Omega^k_{\mathcal{S},e_i}$ chosen non-deterministically by the intruder and such that:

$$\Lambda^k_{\mathcal{S},e_i} \subseteq \Omega^k_{\mathcal{S},e_i} \subseteq \Pi^k_{\mathcal{S},e_i}$$

We note (but do not prove) that a complete strategy (*w.r.t.* the search of an attack) for the intruder is to always choose *at least* all the messages intended to himself.

VI. Security requirements

In this section we define some security requirements that should be satisfied by the model during a trust negotiation session. Recall that a trust negotiation session is the process of computing a least fixed point with respect to (i) the set of trust negotiation rules for each entity in the model and (ii) the non-deterministic choice of actions performed by the intruder, as presented in section V. In this section we present different aspects of confidentiality and authenticity properties.

A. Confidentiality

We distinguish two different kinds of confidentiality, the usual one for which a fixed group of entities shall not be able to obtain a clear object, and a *containment* one that expresses that an entity may possess an object O only if it has been sent to this entity by another honest entity.

Confidentiality: A first notion of confidentiality consists in imposing that an object O must be unknown to a coalition G of entities. This means that, when considering the union $\mathcal{K}_G$ of the sets of keys known by entities in G, none of the entities in G can gain access to a signed object T that can, by performing certain encryption or decryption operations with keys in $\mathcal{K}_G$, retrieve the object O. In order to formalize this notion, let $\mathcal{K}_i$ be the set of available keys for entity e_i. We say that an object O is confidential for the group G if and only if

$$\forall T, \forall k_1, \ldots, k_n \in \mathcal{K}_G, \\ f(f(\ldots f(T,k_1),\ldots),k_n) = O \\ \Rightarrow \forall A, \forall i \in G, \forall m > 0, (\mathcal{E}, \mathcal{S}, e_j \not\models_m get_A(T,i))$$

Containment: An object O is *contained* if no entity can gain access to this object unless it is the exclusive respondent. Note that while this is always the case for a honest entity, such a property is necessary to check whether an intruder can intercept such a message or construct it using his own knowledge. We say an object O preserves containment if and only if

$$\forall A, \forall i, \forall m > 0, \\ (\mathcal{E}, \mathcal{S}, e_{int} \models_m get_A(O,i) \\ \Rightarrow \exists m' \leq m, \exists j \neq int (\mathcal{E}, \mathcal{S}, e_j \models_{m'} put_A(O,j,int)))$$

B. Authenticity

Among the ones proposed in [9] we consider two types of authenticity.

Strong authenticity: for an object O is expressed by the fact that at no time can the intruder send the object O as being someone else. This property is formally expressed as follows:

$$\forall A, \forall i, j, l \in \{1, \ldots, n, int\}, l \neq i, \forall m > 0, \\ (\mathcal{E}, \mathcal{S}, e_l \not\models_m put_A(O,i,j))$$

Weak authenticity: for an object O is expressed by the fact that if an entity e_i receives the object O apparently from entity e_j then the entity e_j previously sent the object O to the entity e_i.

We say an object O preserves weak authenticity if and only if

$$\forall A, \forall i, j \in \{1, \ldots, n, int\}, k > 0, \\ (\mathcal{E}, \mathcal{S}, e_i \models_k get_A(O,j)) \\ \Rightarrow \exists m < k, (\mathcal{E}, \mathcal{S}, e_j \models_m put_A(O,j,i))$$

VII. Attack on Google's implementation of SAML SSO

In [2] the authors describe an attack on SAML SSO[1] protocol that allows a dishonest service provider to impersonate a user at another service provider in the case of Google Applications. In this scenario we assume the presence of a client BOB, an identity provider IDP and an intruder acting as a service provider SP.

In this section we give an instance of the model defined throughout this paper that presents a confidentiality attack. Let $\mathcal{E} = \{e_{bob}, e_{google}, e_{idp}, e_i\}$ where e_{bob} is the client entity, and e_{google} is a service provider, then the trust negotiation policies of these three entities are defined as follows:

In e_{bob}:

$bob2i_1(X)$: $put_{\{conf\}}(X, bob, i) \leftarrow X.\textbf{subject} = bob \wedge X.\textbf{respondent} = i \wedge X.\textbf{resource} = uri$

$bob2i_2(X)$: $put_{\{conf\}}(X, bob, i) \leftarrow get_{\{auth,conf\}}(X, idp) \wedge X.\textbf{respondent} = i$

$bob2idp(X)$: $put_{\{conf\}}(X, bob, idp) \leftarrow get_{\{auth\}}(X, i) \wedge X.\textbf{respondent} = idp$

In e_{google}:

$google2i(bob)_1(X, Y)$:
$put_{\{auth\}}(X, google, bob) \leftarrow get_{\{conf\}}(Y, \wedge Y.\textbf{subject} = bob \wedge Y.\textbf{resource} = calendar \wedge Y.\textbf{respondent} = google \wedge X.\textbf{subject} = google \wedge X.\textbf{requestId} = id_{google} \wedge X.\textbf{respondent} = idp$

$google2i(bob)_2(X, Y, Y')$:
$put_{\{auth\}}(X, google, bob) \leftarrow get_{\{conf\}}(f(Y, k^{-1}_{idp}), bob) \wedge get_{\{conf\}}(Y'), bob) \wedge Y.\textbf{certifier} = idp \wedge Y.\textbf{auth-status} = is\text{-}authentic \wedge Y'.\textbf{resource} = calendar \wedge X.\textbf{data} = secret$

[1] SSO protocols enable companies to establish a federated environment in which a client signs-in once and then has access to resources and services offered by the different companies in the federation

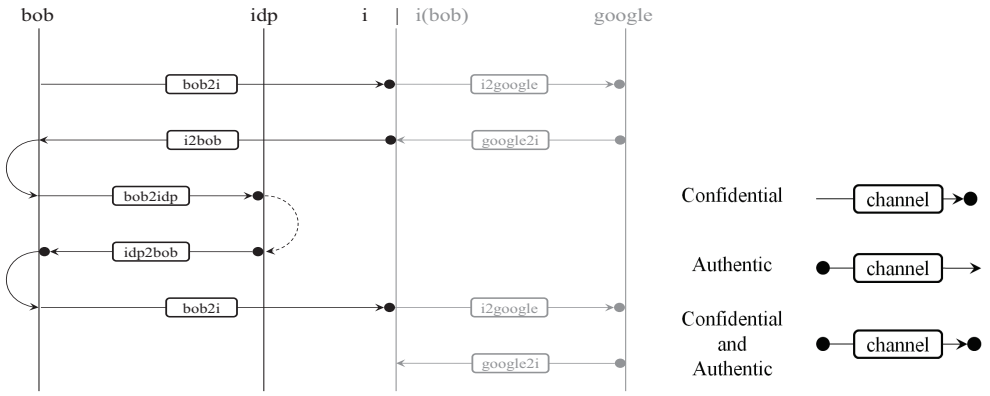

Figure 2. The communication of messages in the attack on SAML SSO for Google application

In e_{idp}:

$idp2idp(X,Y):$
$put_{\{auth,conf\}}(X, \mathit{idp}, \mathit{idp}) \leftarrow get_{\{conf\}}(Y, \mathit{bob})$
$\wedge Y.\textbf{subject} = i \wedge Y.\textbf{requestId} = id_i$
$\wedge Y.\textbf{respondent} = \mathit{idp} \wedge X.\textbf{subject} = \mathit{bob}$
$\wedge X.\textbf{auth-status} = \mathit{is\text{-}authentic} \wedge X.\textbf{certifier} = \mathit{idp}$
$\wedge X.\textbf{respondent} = Y.\textbf{subject}$

$idp2bob_1(X):$
$put_{\{auth,conf\}}(f(X, k_{idp}^{-1}), \mathit{idp}, \mathit{bob}) \leftarrow$
$get_{\{auth,conf\}}(X, \mathit{idp}) \wedge X.\textbf{subject} = \mathit{bob}$

$idp2bob_2(X,Y):$
$put_{\{auth,conf\}}(X, \mathit{idp}, \mathit{bob}) \leftarrow get_{\{conf\}}(Y, \mathit{bob})$
$\wedge Y.\textbf{subject} = i \wedge Y.\textbf{requestId} = id_i \wedge Y.\textbf{respondent}$
$= \mathit{idp} \wedge X.\textbf{resource} = Y.\textbf{resource} \wedge X.\textbf{respondent}$
$= Y.\textbf{subject}$

The rules of the intruder e_i are as described in Section IV. We assume that the intruder has in his repository the certificates:

$(kn, \mathit{i}), (kn, \mathit{c}), (kn, \mathit{sp}), (kn, \mathit{idp}), (kn, \mathit{bob}),$
$(kn, \mathit{google}), (kn, \mathit{calendar}), (kn, \mathit{uri})$

Confidentiality attack: The property saying that the object

$$O := (kn, \mathit{secret})$$

is confidential for the intruder i is violated. In fact e_{bob} communicates with the intruder as being a service provider, however, the intruder acts as e_{bob} to access the service provider e_{google}.

In the above scenario, and with the help of Fig 2, we present an instance of the above communication protocol that violates the confidentiality property. To this end we describe the communication between the entities and the actions performed by the intruder (according to the intruder's capacities and trust negotiation policy) in order for the confidentiality attack to take place.

Suppose bob has in his repository a certificate

$$O := \{(\textbf{subject}, \mathit{bob}), (\textbf{respondent}, \mathit{i}), (\textbf{resource}, \mathit{uri})\}$$

then the rule $bob2i_1(O)$ is satisfied and O is sent to the intruder e_i (acting as a service provider). The intruder can use the rule $Send(\{auth\}, \emptyset, i)$ to send the object O to himself, then rules $Update(resource)$ and $Update(respondent)$ on object O to replace the value uri by the value $calendar$ and the value i by the value $google$ that exist in its repository, and thus constructs a new object

$$O_1 := \{(\textbf{subject}, \mathit{bob}), (\textbf{respondent}, \mathit{google}), (\textbf{resource}, \mathit{calendar})\}.$$

The intruder then use the rule $Send(\emptyset, \{conf\}, bob)$ to sent the newly constructed object O_1 to the entity e_{google} as being bob. In entity e_{google}, the rule $googletoi(bob)_1(O_2, O_1)$ is satisfied, with

$$O_2 := \{(\textbf{subject}, \mathit{google}), (\textbf{requestId}, id_{google}), (\textbf{respondent}, \mathit{idp})\}.$$

Since the object O_2 is sent on an authentic channel, it can be intercepted by e_i and blocked from e_{bob}. e_i can then use the rules $Update(subject)$ and $Update(requestId)$ on O_2 to construct the object

$$O_3 := \{(\textbf{subject}, \mathit{i}), (\textbf{requestId}, id_i), (\textbf{respondent}, \mathit{idp})\}$$

and send it to bob on an authentic channel. The rule $bob2idp(O_3)$ is then satisfied in e_{bob} and consequently the rules $idp2idp(O_4, O_3)$ and $idp2bob_1(O_4)$ are satisfied in e_{idp}, for

$$O_4 := \{(\textbf{subject}, \mathit{bob}), (\textbf{auth-status}, \mathit{is\text{-}authentic}), (\textbf{certifier}, \mathit{idp}), (\textbf{respondent}, \mathit{i})\},$$

and the rule $idp2bob_2(O_5, O_3)$ is satisfied for

$$O_5 := \{(\textbf{resource}, \mathit{uri}), (\textbf{respondent}, \mathit{i})\}.$$

Thus the rules $bob2i_2(O_4)$ and $bob2i_2(O_5)$ are satisfied. Then e_i forward the signed object O_4 to google by the rule

$Send(\{conf\}, \{conf\}, bob)$ and modifies the object O_5 by rules $Update(resource)$ and $Update(respondent)$ to get the object

$$O_6 := \{(\textbf{resource}, \textit{calendar}), (\textbf{respondent}, \textit{google})\}$$

and use the rule $Send(\{conf\}, \{conf\}, bob)$ on O_6 to send it to google as being bob. Then the rule $google2i(bob)_2(SECRET, O_5, O_6)$ is satisfied for the object $SECRET := (\textbf{data}, \textit{secret})$. As this object is sent on a non-confidential channel intended to e_{bob}, it can be intercepted by the intruder. This message is clear and contains the value *secret*. The intruder can then extract this value and compute the secret object.

VIII. Conclusion

In this paper we defined an intruder model as an entity in our attribute based logical framework whose behavior mimics that of a Dolev-Yao intruder by extending the syntax and evaluation semantics of our security rules to take into account the presence of various types of communication channels along with the presence of basic cryptographic primitives allowing encryption and decryption through the use of secret and public keys. We note that in this setting there is *a priori* an unbounded number of possible different available objects because of the cryptographic operations. However we conjecture that we can apply the result of [6] to obtain a decision procedure for the security properties we consider. The utility of this work lies in the fact that during the trust negotiation session, certificates are exchanged among entities depending on a security policy (a set of trust negotiation rules) which is defined locally by each corresponding entity. Thus it is interesting to test, in the presence of different entities with different sets of trust negotiation rules, whether the interaction among these entities satisfies a global security properties expressed by authentication or confidentiality constraints on the whole process. It is worth to mention that the work presented in this paper comes as an extension of the work presented in [3]. For future work we plan to define access control related constraints such as separation of duty constraints on the level of the business process.

Acknowledgements

The work presented in this paper was partially supported by the FP7-ICT-2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu).

References

[1] Alessandro Armando, Roberto Carbone, and Luca Compagna. Ltl model checking for security protocols. In *CSF '07: Proceedings of the 20th IEEE Computer Security Foundations Symposium*, pages 385–396, Washington, DC, USA, 2007. IEEE Computer Society.

[2] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra. Formal analysis of saml 2.0 web browser single sign-on: breaking the saml-based single sign-on for google apps. In *FMSE '08: Proceedings of the 6th ACM workshop on Formal methods in security engineering*, pages 1–10, New York, NY, USA, 2008. ACM.

[3] Philippe Balbiani, Yannick Chevalier, and Marwa El-Houri. A Logical Framework for Reasoning about Policies with Trust Negotiations and Workflows in a Distributed Environment. In *Proceedings of the 4th International Conference on Risks and Security of Internet and Systems*, pages 3–11, Toulouse, France, 2009. IEEE.

[4] Nachum Dershowitz. A taste of rewrite systems. In Peter E. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning*, volume 693 of *Lecture Notes in Computer Science*, pages 199–228. Springer, 1993.

[5] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[6] Shimon Even and Oded Goldreich. On the security of multi-party ping-pong protocols. In *FOCS*, pages 34–39. IEEE, 1983.

[7] Adam Hess, Jared Jacobson, Hyrum Mills, Ryan Wamsley, Kent E. Seamons, and Bryan Smith. Advanced client/server authentication in tls. In *NDSS*. The Internet Society, 2002.

[8] Jiangtao Li, Ninghui Li, and William H. Winsborough. Automated trust negotiation using cryptographic credentials. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 46–57. ACM, 2005.

[9] Gavin Lowe. A hierarchy of authentication specifications. In *CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations*, page 31, Washington, DC, USA, 1997. IEEE Computer Society.

[10] Ueli M. Maurer and Pierre E. Schmid. A calculus for secure channel establishment in open networks. In *ESORICS '94: Proceedings of the Third European Symposium on Research in Computer Security*, pages 175–192, London, UK, 1994. Springer-Verlag.

[11] Sebastian Mödersheim and Luca Viganò. Secure Pseudonymous Channels. In *Proceedings of Esorics'09*, LNCS 5789, pages 337–354. Springer-Verlag, 2009. Extended version: Technical Report RZ3724, IBM Zurich Research Lab, 2009, domino.research.ibm.com/library/cyberdig.nsf.

[12] David A. Plaisted. *Equational Reasoning and Term Rewriting Systems*, volume 1, chapter 5, pages 273–364. Oxford University Press, 1993.

[13] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, Jan 2000.

[14] Marianne Winslett, Ting Yu, Kent E. Seamons, Adam Hess, Jared Jacobson, Ryan Jarvis, Bryan Smith, and Lina Yu. Negotiating trust on the web. *IEEE Internet Computing*, 6(6):30–37, 2002.