Model Driven Security Analysis of IDaaS Protocols

Apurva Kumar

IBM Research – India, 4-Block C, Institutional Area, Vasant Kunj, New Delhi, India 110070
kapurva@in.ibm.com

Abstract. Offloading user management functions like authentication and authorization to identity providers is a key enabler for cloud computing based services. Protocols used to provide identity as a service (IDaaS) are the foundation of security for many business transactions on the web and need to be thoroughly analyzed. While analysis of cryptographic protocols has been an active research area over the past three decades, the techniques have not been adapted to analyze security for complex web interactions. In this paper, we identify gaps in the area and propose means to address them. We extend an important belief logic (the so-called BAN logic) used for analyzing security in authentication protocols to support new concepts that are specific to browser based IDaaS protocols. We also address the problem of automating belief based security analysis through a UML based model driven approach which can be easily integrated with existing software engineering tools. We demonstrate benefits of the extended logic and model driven approach by analyzing two of the most commonly used IDaaS protocols.

Keywords: Security Protocol Analysis, Identity Management, Model Driven Security, Identity as a Service.

1 Introduction

Service providers on the Web are rapidly moving towards a model of focusing on their core business and relying on partners for functions that are needed to support the business. A major enabler for this trend is the existence of protocols that provide user management services over the cloud. We refer to these as IDaaS protocols. While initially limited to providing authentication (single sign-on), user management services are now being used for identity federation, authorization, sharing of user attributes and content. Transactions on the web involving multiple providers often depend on IDaaS protocols for their security needs and thus it becomes important that these protocols be thoroughly examined for the security guarantees they provide.

Important protocols in this space are Security Assertion Markup Language (SAML) [2], OpenID [3] and more recently OAuth [4], but use of proprietary solutions to address specific business to business collaboration requirements is not uncommon. Currently, there is no common set of tools and techniques that can be used to analyze security in these protocols and it requires a lot of skill to perform the analysis. It is often when a protocol starts getting commonly used that it receives enough attention for a thorough analysis, usually once an incident has been reported.

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 312–327, 2011.

We advocate need for analysis tools for standards based and custom identity services that can be incorporated in the toolkit of a common solution designer so that the implications of using a security mechanism in a solution are clearly understood.

Analysis of cryptographic protocols has been an active area of research in the past three decades. We identify challenges in extending these approaches for web based IDaaS protocols. A major challenge is the need to support users without identifying keys (public or shared) which is a common situation in web transactions. Another challenge is the need to support transport layer security (e.g. SSL/TLS), which forms part of most web transactions as a primary construct (much like public and shared key encryption in existing techniques). Finally, unlike typical cryptographic protocols, user interactions play an important role in these protocols. Users perform actions like submitting a request, signing in, accepting terms, clicking a link etc. When identities are not global, establishing that a user has previously performed an action is often more important than knowing its identity. Need for representing user actions in security protocols was motivated in [17].

An important set of approaches for security protocol analysis are based on the Burrows, Abadi, Needham (BAN) [1] logic which is used to express and reason about beliefs for secure communication. It defines a logic based on statements about keys, messages, principals and has inference rules which can be used to generate beliefs at participants based on messages exchanged. We extend the belief logic in fundamental ways to support concepts like users communicating over secure channel, authenticating using passwords and performing actions.

We also propose a Unified Modeling Language (UML) based, model driven approach for automating analysis of protocols that are analyzed using belief logics. The approach allows us to analyze security in a wide range of protocols and transactions using domain specific models as input. We show how representation of protocol concepts and properties in the model can be used to simplify modeling and analysis. Using UML as the basis for automating security protocol analysis ensures that our approach can be easily integrated with the software development process. We demonstrate the use of extended belief logic and model driven analysis by analyzing two of the most well-known IDaaS protocols: SAML browser single sign-on protocol and the OAuth [4] protocol.

The rest of the paper is organized as follows: Section 2 covers comparison with related work. Section 3 provides an overview of BAN logic. Section 4 describes the extended logic proposed in this paper along with an analysis of SAML using the logic. Section 5 describes the model driven analysis approach. In Section 6 we describe analysis of the OAuth protocol. In Section 7, we discuss important conclusions from our work.

2 Related Work

Approaches for security protocol analysis can be broadly classified under two categories. *Inference construction* approaches attempt to use inference in specialized logics to establish required beliefs at the protocol participants. The logic of authentication described in [1] was one of the first successful attempts at representing and reasoning about security properties of protocols. In [6], minor improvements to the logic's syntax and inference rules are suggested to remove some ambiguity. Authors

of [7] introduced the concept of 'recognizability'. Logic in [5] introduces the concept of possession along with belief and uses it to support constructs like 'not originated here'. In [8] authors attempt to consolidate good features from earlier belief logic approaches. These logics have the advantage of being usually decidable and efficiently computable. There have been efforts to automate verification for these logics. In [9], a transformation of BAN logic and inference rules to first order formula is performed and theorem prover SETHEO is used for finding proofs. In [10], the authors attempt to embed BAN logic in EVES theorem prover.

Attack construction approaches on the other hand do not try to establish beliefs at the participants but use model-checking techniques to construct attacks. The states and transitions used for modeling the protocol include modeling the structure of the message passing over the channel and a model of the intruder. The intruder is usually based on a Dolev-Yao model [11], and is allowed to perform any sequence of operations such as data interception, concatenation, deconcatenation, encryption, decryption etc. These complexities result in such approaches suffering from state-space explosion problem. Few works that are representative of this class of approaches are mentioned below. The first such approach was introduced in [11] but the class of protocols studied in this work was very limited. In [12] the author has modeled an extension of Dolev-Yao model in a specialized PROLOG based model-checker. Other approaches in this area include the use of FDR model checker for CSP [13] and use of SAT based model-checking techniques to solve a simplified version of the protocol insecurity problem [14]. We note that protocol modeling is generally quite complex in these approaches and even the analysis stage often depends upon user inputs at various stages of the state exploration process.

In our work we prefer to use the inference construction based approach for several reasons. Firstly, attack construction approaches do not actually determine what the protocol achieves but only whether a particular bad state (e.g. a secret being discovered by intruder) is reachable. While this may at times be sufficient for authentication problems, it is hard to specify goals of generic web transactions in this manner. Secondly, the higher abstraction level of inference based approaches allows us to focus on the protocol flow rather than insecurities at the message level. For web transactions, message level security is increasingly being addressed through transport level security. Finally, browser-based protocols are harder to analyze in attack construction based approaches that explicitly model the intruder, since it requires extending the intruder model to incorporate browser-based attacks.

Another relevant work is the analysis of SAML in [15] but the author's do not propose a generic framework for analysis of similar protocols.

3 Overview of Logic of Authentication

BAN is a logic of belief of honest principals participating in a security protocol. Since our approach is an extension of the belief logic described in [1], we provide a brief overview of BAN logic in this section.

BAN statements. A formula in BAN logic is constructed using operators from Table 1. P and Q range over principals. The two statements about keys represent atomic statements. In all other statements X represents a BAN formula constructed using one or more BAN operators.

Table 1. Operators used in BAN logic. X represents a statement in BAN logic.

Notation	Meaning	Notation	Meaning
$P \mid\equiv X$	P believes X	$P \overset{K}{\leftrightarrow} Q$	Shared key K
$P \triangleleft X$	P sees X	$\mapsto_K Q$	Q has public key K
$P \mid\sim X$	P said X	$\{X\}_K$	X encrypted by K
$P \mid\Rightarrow X$	P controls X	$\sharp X$	X is fresh

The expression $\sharp X$ means that the message X is fresh and has not been used before the current run of the protocol. This is especially true for a *nonce*, a sequence number or timestamp generated with this specific purpose. Nonces are used in protocols to defeat replay attacks from previous executions of the protocol.

Inference Rules. There is a set of inference rules for deriving new beliefs from old ones. E.g. the *message-origin* inference rule below states that if P knows that K is a secret key between itself and Q and it sees a message X encrypted by K, then P is entitled to believe that Q said X. A similar inference rule about public keys is also provided where K^{-1} represents the private key corresponding to public key K.

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P, P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X} \qquad \frac{P \mid\equiv \mapsto_K Q, P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X} \tag{R1}$$

A *nonce-verification* rule (R2) states that, in addition if the message is known to be fresh, then P believes that Q must still believe X. Further, the *jurisdiction rule* (R3) states that, if in addition, P also believes that Q is an authority on the subject of X (i.e. Q controls X), then P is entitled to believe X himself.

$$\frac{P \mid\equiv Q \mid\sim X, P \mid\equiv \sharp X}{P \mid\equiv Q \mid\equiv X} \tag{R2} \qquad \frac{P \mid\equiv Q \mid\equiv X, P \mid\equiv Q \mid\Rightarrow X}{P \mid\equiv X} \tag{R3}$$

Idealization. Each message exchanged in the protocol is idealized into a BAN formula representing meaning of the message including any facts that the sending of the message implies. Consider for example, the second message in the Needham Schroeder protocol [1] in which a server S sends a response to an initiator A containing a session key K_{ab}, along with a message for another principal B encrypted using B's key containing the same session key and A's identity. In typical Alice-Bob notation used in literature this can be expressed as:

$$S \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$$

where N_a is a nonce value. K_{as} and K_{bs} represent keys shared between A and S, B and S respectively. The message is idealized as follows:

$$S \rightarrow A : \{N_a, (A \overset{K_{ab}}{\leftrightarrow} B), \sharp(A \overset{K_{ab}}{\leftrightarrow} B), \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}\}_{K_{as}}$$

The idealization makes explicit that the server says that K_{ab} is a shared key for communication between *A* and *B* and also that it is fresh (due to the presence of the nonce).

Analysis. Protocol analysis in inference construction approaches involves two main tasks: (*i*) identification of an initial set of beliefs i.e. assumptions at each principal. (*ii*) message-by-message manual reasoning based on combining formula (idealized messages) that a principle sees and what it knows using inference rules of the logic.

4 Extending Belief Logic

Our goal is to extend BAN for analysis of generic browser based web transactions. This requires us to extend the logic of authentication in the following ways.

Supporting Principals without Identifying Keys. Existing techniques for security protocol analysis require principals to possess identifying keys. However, it is common for users to authenticate to websites using passwords over secure connections. A *secure channel* in this paper refers to a transport layer security mechanism e.g. SSL, TLS that provides server authentication, confidentiality and integrity in message exchanges. We introduce a new sort (type) in the many sorted BAN logic called *user* which represents the client side of a secure connection.

Support for Passing Domain Specific Information. There is often need for exchanging protocol specific information. The information could represent a role of a participant, a set of authorizations required from another site etc. We allow this by providing protocol specific parameters to be defined.

Support for User Actions and Secrets. While principals make statements signed using identifying keys, users interact with principals (usually servers) over secure channels. We define the concept of an action and allow it to be associated with a user. We allow secrets to be associated with actions in order to identify a user that performed an action (possibly at a different place or time). Actions are defined as *Aname* (t_1,t_2..t_n) where t_i represents a sort (type) in the logic or a parameter name. A specific action is *SignIn* (*Principal*) which represents user signing in as a principal.

The new concepts are represented using the notation described in Table 2.

Table 2. Additional operators used in our extended belief logic. *Aname* ranges over action names, while *Pname* ranges over parameter names. Secure channel associated with a user is identified by a suffix. New operators are allowed in formula *X* in inference rules R1-R3.

Notation	Meaning	Notation	Meaning
$P \overset{\Delta}{\longleftrightarrow} U_C$	*C is a secure channel between U_c and P.*	$X \rightsquigarrow Aname$	*Secret X is associated with action Aname.*
$[\![X]\!]_C$	*Formula X exchanged over secure channel C.*	$U_C \ni X$	*U_c possesses secret X.*
$U_C \triangleright Aname((t_1..t_n))$	*U_C performs action Aname*	*Pname = val*	*Parameter Pname has value 'val'.*

4.1 Reasoning about Users, Actions and Secrets

The following new inference rules have been defined to reason about users, their actions and associated secrets.

Table 3. Inference rules added to the logic. The description assumes that P is connected to U_c over a secure channel C.

Inference Rule (R4, R5, R6)	Description
$\dfrac{P \models (P \overset{\Delta}{\longleftrightarrow} U_C), P \lhd [\![Aname]\!]_C}{P \models (U_C \rhd Aname)}$	*If P sees an action on the channel C it believes U_c performed the action.*
$\dfrac{P \models (S \rightsquigarrow Aname), P \models (P \overset{\Delta}{\longleftrightarrow} U_C), P \lhd [\![S]\!]_C}{P \models (U_C \rhd Aname)}$	*If P sees a secret associated with an action on channel C, it believes U_c performed the action.*
$\dfrac{P \models (P \overset{\Delta}{\longleftrightarrow} U_c), P \models U_c \rhd SignIn(Q), P \lhd [\![Aname]\!]_C}{P \models (Q \rhd Aname)}$	*If P believes that U_c has already signed in as Q, then any further actions on C are associated with Q.*

4.2 Example: Analysis of SAML Web Single Sign-On

We assume that all communication between client and servers is protected by a secure channel. We introduce new parameters identifying roles of identity provider, *Prov* and identity consumer, *Cons* and service, *Service*. We introduce the action *Request* (*Service*) and use the action *SignIn* (*Principal*) defined in Section 4. We include only those messages in the idealized version with target as *SP* (assertion consumer, *C*) or IdP (assertion provider, P) since we are interested in beliefs only at these participants:

$$\begin{aligned}
&\text{Message 1} \quad U_{C1} \rightarrow C: \quad [\![Request(S)]\!]_{C1} \\
&\text{Message 3} \quad U_{C2} \rightarrow P: \quad [\![\{N_{cp}, prov = P, cons = C\}_{K_C^{-1}}]\!]_{C2} \\
&\text{Message 5} \quad U_{C2} \rightarrow P: \quad [\![N_{cp}, SignIn(Q)]\!]_{C2} \\
&\text{Message 7} \quad U_{C3} \rightarrow C: \quad [\![\{N_{cp}, T, T \rightsquigarrow SignIn(Q), \\
&\qquad\qquad prov = P, cons = C\}_{K_P^{-1}}]\!]_{C3}
\end{aligned}$$

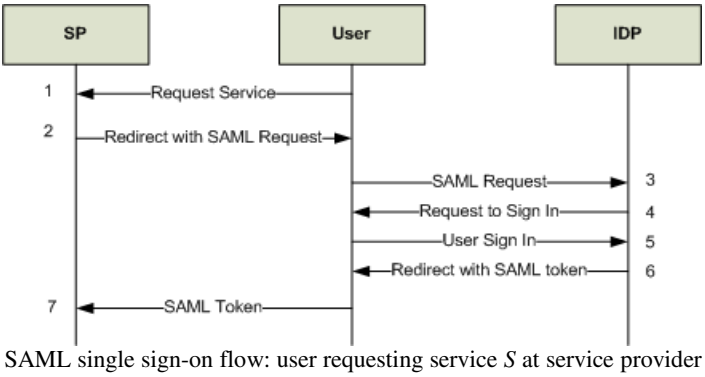


Fig. 1. The SAML single sign-on flow: user requesting service *S* at service provider (SP) site is redirected to Identity Provider (IdP) with SAML request. After authenticating the user, IdP redirects user back to SP site with a signed SAML token having the asserted identity (*Q*).

The SAML request contains a request identifier and a timestamp which we combine as a single nonce value N_{cp} in Message 3. Message 7 is the SAML response with token T, association of the token with the sign-in action performed by the user. We now perform a BAN style analysis of the protocol. We start with the assumptions:

$$C \models C \overset{\triangle}{\leftrightarrow} U_{C1} \qquad P \models P \overset{\triangle}{\leftrightarrow} U_{C2}$$
$$C \models C \overset{\triangle}{\leftrightarrow} U_{C3}$$
$$C \models \sharp(N_{cp}) \qquad P \models \sharp(N_{cp})$$
$$C \models \mapsto_{K_P} P \qquad P \models \mapsto_{K_C} C$$
$$C \models (P \models\!\Rightarrow SignIn(X)) \qquad P \models (C \models\!\Rightarrow prov)$$
$$C \models (P \models\!\Rightarrow cons)$$

The first three assumptions are about secure channels. The next say that both C and P believe the nonce N_{cp} to be fresh. The next two statements are about C and P having knowledge of the others public key. The last assumption is about C trusting P on the subject of the *SignIn* action. The analysis is described in Table 4. The inference in the last row says that C now believes that U_{c3} has signed in as Q at P.

Table 4. SAML analysis for idealized messages. Receiver of message indicated in parentheses

Msg.	Reasoning	Rules	Inference
1 (C)	*Combining message with assumption about channel C1.*	R4	$C \models (U_{c1} \triangleright Request(S))$
3 (P)	*Combining message with assumption about K_c and N_{cp}.*	R1, R2	$P \models C \models (prov = P, cons = C)$
3 (P)	*Using assumption about C having control over prov.*	R3	$P \models (prov = P)$
5 (P)	*Combining message with assumption about channel C2.*	R4	$P \models (U_{c2} \triangleright SiginIn(Q))$
7 (C)	*Combining message with assumption about K_p and N_{cp}*	R1, R2	$P \models T \rightsquigarrow SiginIn(Q)$, $P \models (prov = P), P \models (cons = C)$
7 (C)	*Using assumption about P's control over SignIn(X).*	R3	$C \models (T \rightsquigarrow SiginIn(Q))$
7 (C)	*Using the fact that T is observed on secure channel C3.*	R5	$C \models (U_{C3} \triangleright SiginIn(Q))$

5 Model Driven Security Protocol Analysis

5.1 Automated Analysis of Belief Logics

Analysis of security protocols using a belief logic like BAN is often simple enough to be carried out manually and message-by-message in a way similar to the SAML example. Despite this there have been efforts to automate verification for these logics [9, 10]. We consider automation to be even more important for our logic which is not restricted to authentication problem and is a bit more complex than BAN. Existing

approaches like [9] perform mapping of BAN formula and inference rules to first-order logic. The approach essentially transforms all the assumptions, messages and inference rules into a single first order formula and feeds it to the theorem prover SETHEO for proving implication of a required theorem. There are two problems with such approaches. Firstly, combining formulae representing all the messages as described above is unsafe because it results in loss of relative timing and it becomes possible to use an inference rule to resolve a message based on information that became available only later in the protocol run. Secondly, existing theorem provers are not user-friendly enough to be incorporated in the software development process. For these reasons, we propose a model driven approach for automated analysis of protocols modeled using belief logics like BAN and its extensions.

5.2 Overview

Reasoning performed in belief logics like BAN is not very complex. Given a message, the applicable inference rules and the sequence in which they apply is often deterministic (e.g. R1 results in expressions that can be used with R2 which in turn leads to expressions usable in R3). We thus follow a forward chaining approach to discover all possible beliefs. We store beliefs at a principal using a model representing sorts (types) in the logic and relationships between them. The algorithm we use for combining messages with existing beliefs is aware of this model and makes queries only to retrieve relevant facts e.g. if it sees a message encrypted with K, it queries to find owners of K.

The entities and relationships in the model are not fixed and can be specific to a domain. This makes our approach model driven as the same analysis program works with different protocol specific models without any change. The analysis program is only impacted when new inference rules are added. Entities in the model correspond to all entities that appear in a protocol description while relationships correspond to beliefs established through a protocol execution. We call this the *domain model* of the protocol. An instance of this model is used at each participant to represent its current set of beliefs. This is termed as the *belief graph* of the participant.

The analysis program is given an idealized representation of the protocol to be examined, a domain model of the protocol and a set of assumptions. The program uses unification algorithms that we discuss in Section 5.5, which process an idealized formula by combining it with known facts available from the belief graph of the message recipient on the basis of inference rules of the logic, to generate beliefs. The beliefs are added to the belief graph of the participant. Once all the protocol messages have been processed by the program, the belief graph at each protocol participant represents its final set of beliefs.

5.3 Benefits of Model Driven Analysis

Ease of Use. The model driven analysis approach is based on UML which makes it easy to integrate with existing CASE tools. Viewing belief graphs generated by the

analysis program provides tremendous insight into the design of the protocol and often provides hints on resolving security issues.

Simplifying Idealization. A protocol parameter can be mapped to a high level protocol specific concept in the model. This reduces complexity in each idealized expression where the concept is referred.

Representing Protocol Properties. Some properties of the protocol are important for proving goals even though they are never communicated to the participants through protocol messages. Use of idealization to represent such properties is both arbitrary and error prone. We solve this problem by representing such properties in the model itself by specifying constraints, thus removing the need for them to be artificially communicated through idealized messages.

5.4 Modeling of Extended Belief Logic

Message handling algorithms that perform reasoning have to interact with the belief graph to query existing beliefs and to add new ones. This requires statements representing belief in the logic to be mapped to the model. Table 5 shows how atomic sentences of the logic are modeled in the graphical representation. Each important sort (type) of the logic is mapped to a UML class (stereotyped entity) and statements are mapped to associations. The important classes are `Principal`, `SharedKey`, `AsymKey`, `Token`, `User`, `Request`, `Action`. `Token` is used to represent both secrets and nonces, `AsymKey` to represent public/private keys, while `Request` is used to represent protocol sessions. `Action` is an abstract class and is the super class for specific actions like `SignIn`.

Table 5. Mapping of statements of the logic to graphical UML syntax. Multiplicity at association ends is used to limit ownership of entities like keys. Trust types in parentheses.

Statement	Graphical Representation	Statement	Graphical Representation
$P \xleftrightarrow{K} Q$ *(SkeyTrust)*	«entity» **Principal** — +subjects (2 — 0..*) — «entity» **SharedKey**	$\mapsto_K Q$ *(PkeyTrust)*	«entity» **Principal** — +subject (1 — 0..1) — «entity» **AsymKey** +type
$\sharp X$ *(NonceTrust)*	«entity» **Request** — +freshvals (1 — 0..*) — «entity» **Token** +val	$U \ni X$ *(HolderTrust)*	«entity» **User** +channel — +holder (0..* — 0..*) — «entity» **Token** +val
$U \triangleright Action$, $X \rightsquigarrow Action$ *(Action-Trust)*	«entity» **Action** — +performedBy (* — 1) — «entity» **User** +channel; +secret (0..*) — «entity» **Token** +val	$U \triangleright SignIn$ $X \rightsquigarrow SignIn$ *(SigninTrust)*	«entity» **SignIn** — +performedBy (* — 1) — «entity» **User** +channel; +authSub (0..* — *) — «entity» **Principal**; «entity» **Token** +val — +secret (*)
$P \xleftrightarrow{\Delta} U_c$ *(SecChannel Trust)*	«entity» **User** +channel		

Note that statements that include operators $\lhd, \{\ \}_K, [\![\ ]\!]_C$ appear only in premises of the inference rules of the logic are not required to be modeled as we are only interested in beliefs established at the participants. Jurisdiction statements are not mapped to the model. Instead each statement in Table 5 is associated with a trust type. An N x N table (where N is the number of participants) is used to represent trust between participants. An element of the table, TRUST [P,Q] is a set of trust types with each type t such that $P \models Q \models\Rightarrow t$. The table is provided as one of the assumptions to the analysis program.

5.5 Unification Algorithms

Reasoning involves unifying (combining) the message and one or more existing beliefs with the inference rules of the logic to obtain new beliefs. We now describe the message handling algorithms which process an idealized message, perform reasoning to obtain beliefs and store the beliefs in a participant's belief graph.

Handling Encrypted Formula. The algorithm HANDLEENCRFORMULA takes an encrypted statement of the logic, and the name of the principal that receives the formula as its input. Any encrypted sub-formulae are removed (routine EXTRACTENCRFORMULA) and processed using a recursive call. The rest of the algorithm corresponds to applying the nonce-verification and message-origin rules. Any tokens which represent fresh values or secrets as per the belief graph are extracted (GETTOKEN). If no nonces are found, then the message is not fresh and no beliefs can be established. Otherwise, the origin of the assertion is identified using the encryption key for the message. In the event the encryption key is not known, the message is rejected. If the key is the public key owned by the principal itself, then any secrets in the message are used to identify the origin. Once the source subject is identified, ADDBELIEF is called for each sub-formula.

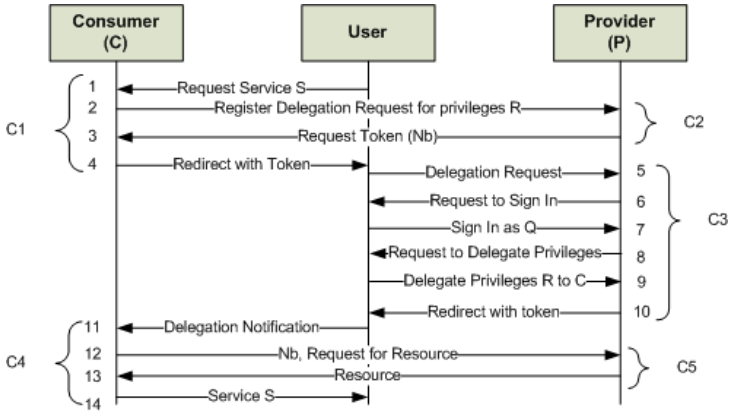
Handling Secure Channel Messages. The algorithm HANDLESECUREMSG takes a statement, a secure channel identifier for the channel on which it is received and the principal where it is received as the input. It is determined whether the channel is known to the principal (routine GETCHANNEL). If the principal is acting as a client, then the other end of the channel is the subject principal and an attempt is made to add each sub-formula using ADDBELIEF. If the principal is acting as a server, the other end represents a user and the formula could either be an action or a secret. If it is an action, the ADDACTION routine is called. In case of secret any associated actions are inferred to be performed by the user and added to the participant's belief graph. The ADDBELIEF routine adds beliefs (including beliefs about other participants beliefs e.g. $A \models\equiv B \models\equiv$) to a participant belief graph applying jurisdiction rule in the process.

```
procedure HANDLEENCRFORMULA(stmt, principal)
    local variables: m, formulae,
    subject ← null, candidates ←{ }, nonces←{ }, secrets←{ }
    while (m ← EXTRACTENCRFORMULA(stmt)) do
        HANDLEENCRFORMULA(m, principal)
    GETTOKENS(stmt, principal, nonces, secrets)
    if (nonces = { }) then return
    key ← KEY[stmt], formulae ← FORM[stmt], type← KEYTYPE [stmt]
    candidates ← GETPRINCIPALS(key)
    if (candidates = { }) return
    if (candidates = {principal} and SIZE[secrets]=1) then
        candidates ← GETPRINCIPALS(secrets)
    if (candidates = {principal, Q}) then subject ← Q
    else if (candidates = {Q}, type = Private) then subject ← Q
    if (subject = null) then return
    ADDBELIEF(formulae, subject, principal) return

procedure HANDLESECUREMSG (stmt, cid, principal)
    local variables: m, formulae, subject, channel ←null, action←null
    while (m ← EXTRACTENCRFORMULA(stmt)) do
        HANDLEENCRFORMULA(m, principal)
    channel ← GETCHANNEL(cid, principal), subject ← CLIENT[channel]
    if (channel = null) then return
    formulae ← FORM[stmt]
    if (subject= principal) then
        ADDBELIEF(formulae, SERVER[channel], principal) return
    for each (m ∈ formulae) do
        if (ACTION?(m)) then aname = ACTIONNAME[m]
            if (aname = SignIn) then
                ADDSIGNINACTION(subject, AUTHNAME[m], principal)
            else ADDACTION(subject, m, principal)
        else if (SECRET?(m)) then action = GETACTION(m)
            if (action = null) then ADDSECRET(subject, m, principal)
            else ADDACTION(subject, m, principal)
    return

procedure ADDBELIEF(formulae, subject, principal)
    for each (f ∈ formulae) do
        ADDSTATEMENT(f, subject, principal)
        if (TRUSTTYPE[f] ∈ TRUST[principal, subject])
            ADDSTATEMENT(f, principal, principal)
    return
```

6 OAuth Protocol Analysis

The OAuth protocol [4] provides a web based workflow that allows a user to temporarily delegate privileges of his account at a provider to a third party without sharing his login credentials. Privileges could mean access to pictures, contacts, blogs etc. OAuth is the primary protocol used by Google, Facebook and Twitter to allow third party access to their users' content. In this paper, we use OAuth to refer to the original version (1.0) of the protocol.

We show the concrete and idealized protocols next to each other in Fig. 2. Message 2 represents registration of delegation request by C at P. The message signed by C,



Message 1	$U \rightarrow C : Access\ Service\ S$	$\rightleftharpoons$	$U_{C1} \rightarrow C : \ [\![Request(S)]\!]_{C1}$
Message 2	$C \rightarrow P : \{R, N_c, url_c\}K_c^{-1}$	$\rightleftharpoons$	$C_{C2} \rightarrow P : \ [\![\{N_c, Privileges = R, Callback = url_c\}K_c^{-1}]\!]_{C2}$
Message 3	$P \rightarrow C : N_b$	$\rightleftharpoons$	$P \rightarrow C_{C2} : \ [\![OAuthTkn = N_p]\!]_{C2}$
Message 4	$C \rightarrow U : N_b, url_p$		
Message 5	$U \rightarrow P : N_b$	$\rightleftharpoons$	$U_{C3} \rightarrow P : [\![N_p]\!]_{C3}$
Message 6	$P \rightarrow U : Login\ request$		
Message 7	$U \rightarrow P : Q,\ password$	$\rightleftharpoons$	$U_{C3} \rightarrow P : [\![SiginIn(Q)]\!]_{C3}$
Message 8	$P \rightarrow U : Request\ for\ delegation$		
Message 9	$U \rightarrow P : Delegate\ R\ to\ C$	$\rightleftharpoons$	$U_{C3} \rightarrow P : [\![Delegate(R, C)]\!]_{C3}$
Message 10	$P \rightarrow U : N_b, url_c$		
Message 11	$U \rightarrow C : N_b$	$\rightleftharpoons$	$U_{C4} \rightarrow C : [\![N_p]\!]_{C4}$
Message 12	$C \rightarrow P : N_b,\ Access\ Content$		
Message 13	$P \rightarrow C : Content$	$\rightleftharpoons$	$P \rightarrow C : \ [\![x \triangleright Delegate(R, C)]\!]_{C5}$
Message 14	$C \rightarrow U : Provide\ Service\ S$		

Fig. 2. OAuth protocol flow. *Steps 1-4*: User requests service S from C which requires access privileges R to the user account at P. C registers delegation request with P and gets returned a OAuth token. C redirects user to P with the token. *Steps 5-10*: User is requested to sign in and delegate access and then redirected back to C with the token. *Steps 11-14*: C uses the token to get user content from P and provides service S to the user. C1-C5 represents secure channels.

contains list of privileges to be delegated, a callback URL (url_c) and a nonce N_c, identifying the request. In response P returns the request token, N_b.

An execution of the protocol can be uniquely identified as $OAuth(C, url_c, S, R, N_c, N_b, Q)$. We name the protocol parameters as *Consumer*, *Callback*, *Service*, *Privileges*, *Nonce_c*, *OAuthTkn, AuthPrincipal*. We only use parameters when there is no statement in the logic which can succinctly represent a message. In the idealized protocol we use only the following three parameters: *Callback*, *Privileges* and *OAuthTkn*. The user performs three actions in the protocol: request for service in message 1, signing in as Q in message 7 and delegating privileges to C in message 9. We define the following action types, *Request*(*Service*), *SignIn*(*Principal*) and *Delegate* (*Privileges*, *Principal*). Instances of these action types are *Request*(*S*), *SignIn*(*Q*) and *Delegate*(*R*, *C*).

Since we are interested only in beliefs established at principals C and P, we ignore messages with destination as user during idealization. The idealization of message 13 is interesting. P's returning requested user content to C implies that some user x has performed the delegation.

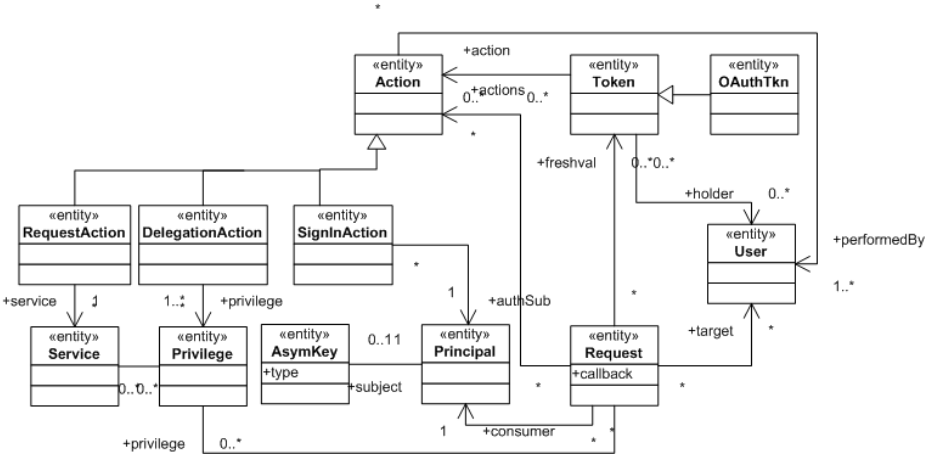


Fig. 3. Domain model for OAuth. Idealization is greatly simplified using parameters and actions that map to the domain model.

The domain model of the protocol is shown as a UML class diagram in Fig. 3. Entities `Principal`, `User`, `Action`, `Token`, `AsymKey` are from the mapping described in Section 5.4. `Service`, `Privileges` and `OAuthTkn` (sub-class of `Token`) are protocol specific entities. Classes representing the three specific user actions are defined and associated with other entities like `Service`, `Privilege`, `Principal`. The `target` relationship is used to identify the `User` which is provided the service in Message 14. The constraints on the model are described in Table 6. Property 1 in Table 6 is a property of the `OAuthTkn`. We can see how mapping parameter *OAuthTkn* to class `OAuthTkn` simplifies idealization by replacing two statements: $\sharp N_b$, $N_b \rightsquigarrow Request(S)$. Property 2 is an example of how protocol properties which are useful for proving goals but do not appear in an idealized message can be represented in the model.

Table 6. Constraints representing properties and goals of the protocol. Object constraint Language (OCL) expressions are used to specify constraints.

Type	OCL Expression	Meaning
Definition	**def** *delegator* : `User` = self.*actions*-> **select**(OclAsType(`DelegationAction`)). *performedBy*	Identify user who performed delegation as *delegator*.
Property 1	**context** `OAuthTkn` *self.isFresh* = *true* **and** self.action-> **select**(OclAsType(`RequestAction`))	An OAuth Token is fresh as well a secret associated with Request action.
Property 2	**context** `Request` *delegator* =self.*freshvals*-> **select**(OCLAsType(`OAuthTkn`)).*holder*	User who delegates privileges also holds OAuth token.
Goal	**context** `Request` self.*target* = *delegator*	Service is given to *delegator.*

Fig. 4 shows the belief graph at *C* after the automated analysis program has processed all idealized message using HANDLESECUREMSG. We can see that the goal constraint in Table 6 is not satisfied since user U_{C4} (which is the target) is not associated with the delegation action. Inspecting Fig. 4, it is not hard to find an assumption, which if true, can satisfy the goal. User *x,* associated with delegation action, is also holder of OAuth token as are users U_{C1} and U_{C4}. If one assumes that U_{C1} which originally received the token does not share the token then all user instances shown in Fig. 4 can be merged and we can immediately see that the target and delegator links point to the same user instance. However, this is clearly an unreasonable assumption to make for an unauthenticated user.

A security issue [16] was reported with the original version of OAuth which was fixed in the next release of the protocol. The attack works as follows: The attacker can perform steps 1-4 (see Fig. 2) of the protocol with the Consumer, send a link with the request token to the victim (who has an account with the provider) which then performs steps 5-10 and then rejoin the protocol from step 11 onwards. This quite clearly exploits the invalid assumption identified above.

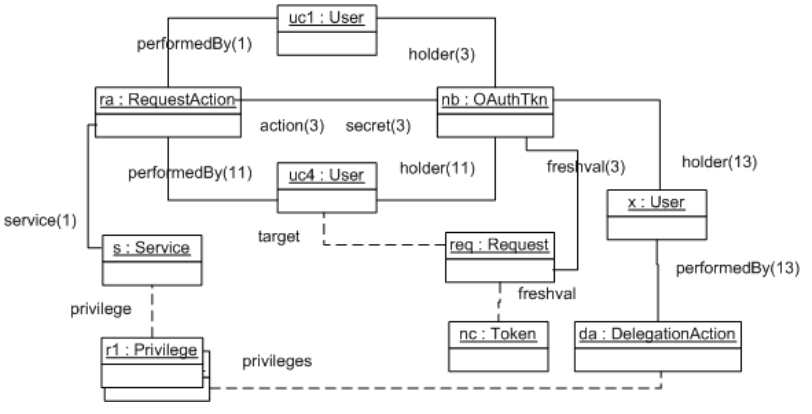


Fig. 4. Belief graph generated by the analysis program for Consumer C. Objects names are derived from idealized expressions. Broken lines represent assumptions. For other links the message number that resulted in the belief is identified in parentheses.

7 Conclusion

Protocols that provide user management services to third parties are central to the cloud computing model and analyzing their security properties is extremely important. We identified some unique aspects of these protocols that do not allow analysis methods for cryptographic protocols to be applied. We find that existing inference construction based approaches are the closest to satisfying the need. We extend an important belief logic that has been used successfully for analyzing numerous cryptographic protocols for analysis of web based identity services. We provide algorithms to automate analysis for belief based analysis approaches. We demonstrate the extended logic and model driven approach through analysis of two of the most well known browser based IDaaS protocols. UML has particularly wide acceptance in the software industry and supported by all major CASE tools. Use of UML for automating security analysis allows third party developers to appreciate implications of using standard or custom identity services in their solutions.

References

1. Burrows, M., Abadi, M., Needham, R.: A Logic of Authentication. ACM Transactions on Computer Systems (TOCS) 8(1), 18–36 (1990)
2. OASIS SAML Specifications. SAML v2.0, Core, `http://saml.xml.org/saml-specifications`
3. OpenID 2.0 Specifications, `http://openid.net/specs/openid-authentication-2_0.html`
4. The OAuth 1.0 Protocol. IETF RFC: 5849, `http://www.rfc-editor.org/rfc/rfc5849.txt`
5. Gong, L., Needham, R., Yahalom, R.: Reasoning about Belief in Cryptographic Protocols. In: Proceedings 1990 IEEE Symposium on Research in Security and Privacy (1990)
6. Abadi, M., Tuttle, M.R.: A semantics for a logic of authentication. In: Proceedings of the ACM Symposium of Principles of Distributed Computing (1991)
7. Kessler, V., Wedel, G.: AUTLOG: An advanced logic of authentication. In: Proceedings of Computer Security Foundation Workshop VII, pp. 90–99 (1994)
8. Syverson, P., van Oorschot, P.: On unifying some cryptographic protocol logics. In: Proceedings of the Symposium on Security and Privacy, Oakland, CA, pp. 14–28 (1994)
9. Schumann, J.: Automatic Verification of Cryptographic Protocols with SETHEO. In: McCune, W. (ed.) CADE 1997. LNCS, vol. 1249, pp. 831–836. Springer, Heidelberg (1997)
10. Craigen, D., Saaltink, M.: Using EVES to analyze authentication protocols. Technical Report TR-96-5508-05, ORA Canada (1996)
11. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inform. Theory IT-29, 198–208 (1983)
12. Meadows, C.: Applying formal methods to the analysis of a key management protocol. Journal of Computer Security 1, 5–53 (1992)
13. Lowe, G.: Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)

14. Armando, A., et al.: An Optimized Intruder Model for SAT-based Model-Checking of Security Protocols. Elec. Notes in Theoret. Comp. Sci. 125(1) (March 2005)
15. Groß, T.: Security analysis of the SAML single sign-on browser/artifact profile. In: Proceedings of 19th ACSAC 2003, pp 298–307. IEEE Computer Society Press (2003)
16. Hammer-Lahav, E.: Explaining the OAuth Session Fixation Attack, http://hueniverse.com/2009/04/explaining-the-oauth-session-fixation-attack/
17. Kumar, A.: Integrated Security Context Management of Web Components and Services in Federated Identity Environments. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 565–571. Springer, Heidelberg (2008)