Formal Analysis of the CORBA CSIv2 Security Protocol

Polar Humenn, Susan Older, Shiu-Kai Chin
Systems Assurance Institute, Syracuse University, Syracuse, New York 13244, USA

Abstract

In this paper we look at reasoning about access control for distributed resources over CORBA. We analyze the CORBA CSIv2 protocol, which is the internationally accepted standard security protocol for CORBA distributed requests. We also introduce a logic for authentication and access control that supports reasoning about the principals in a distributed system, the statements they make and their delegations. Using this logic we formally reason about the CSIv2 protocol, semantics, and decisions of granting access to CORBA requests.

1 Introduction

Distributed security is an intricate property that is achieved only by a combination of sufficiently strong cryptographic algorithms, protocols, and appropriate trust assumptions about authorities. For example, if a service provider Carol receives a request purportedly from Alice through a broker Bob, Carol must determine whether or not to grant the request. To make that decision, she must determine whether Bob and Alice are who they claim to be, whether Bob is authorized to make requests on Alice's behalf, and whether Bob and Alice are authorized to make the specific request. Assuring that these decisions are made correctly and subsequently form a secure system is difficult without the use of rigorous and formal analytical techniques.

The Common Object Request Broker Architecture (CORBA) and its Common Secure Interoperability Version 2 Protocol (CSIv2) give us part of that picture. CORBA is an open standard that supports component-based software design and services at the middleware level [COR98]. CSIv2 is an accepted CORBA standard in support of authentication of client identity [CSI01]. The CSIv2 protocol was designed specifically to augment existing authentication technology (e.g., SSL) with authorization and delegation information that can be used to distinguish brokers from the clients who originate requests. A formal way to analyze the CSIv2 standard with respect to access control is needed.

To reason about delegations and access-control decisions, we introduce a specialized modal logic that originally appeared in a a series of papers by Abadi and colleagues [LABW92,ABLP93,WABL94]. This logic supports reasoning about the statements made by principals (i.e., the actors in a system), their beliefs, their delegations, and their authorizations.

The rest of this paper is organized as follows. Section 2 describes the CORBA CSIv2 protocol. Section 3 describes the syntax and semantics of Abadi and colleagues calculus [LABW92,ABLP93,WABL94] that is used to reason about principals and their beliefs. In Section 4, we illustrate how that logic can be used to describe aspects of a Web Server brokered CSIv2 request. Finally, our conclusions are presented in Section 5.

2 The CSIv2 Protocol

The protocol we describe in this paper is the Common Object Request Broker Architecture (CORBA) Common Secure Interoperability Version 2 Protocol (CSIv2). The CSIv2 protocol is a network communications protocol designed for passing on CORBA requests from clients (service requesters) to targets (service providers) with security information. The protocol augments existing identity authentication technology (e.g. SSL) with identity delegation and authorization information.

We describe the CSIv2 protocol in terms of *principals*, who are the subjects of the system: they make requests, grant privileges, and delegate or endorse other principals to act on their behalf. When a service provider, Carol, receives a request, she must determine who is actually making the request: the principal who transmitted it to her may be making the request on his own behalf or on behalf of another principal. Furthermore, the provider must also determine if that principal is authorized to make that request.

Although the CSIv2 protocol can accommodate a number of different transport and authentication protocols, we focus on two characteristic cases: TCP/IP (where the transport's client identity is not authenticated) and SSL with a client certificate (where the transport's client identity is authenticated). In this paper, without loss of generality, we assume that Carol receives all requests over TCP/IP or SSL.

When Carol receives the request over TCP/IP, the transport principal is *Default*: Carol does not authenticate this principal, but she assumes a default entity sends her requests. This scenario is appropriate for situations in which the network is secure (e.g., private networks) and for which Carol assumes that the request arrived through a pathway she trusts. When, instead, Carol receives the request over SSL, she authenticates the transport principal (who we now call Tony) using a public-key certificate and sets up a protected pipe. This protected pipe enables authentication, integrity, and possibly confidentiality protection in which all requests delivered to Carol are cryptographically signed by Tony. We use the verb, **says**, to denote that a cryptographic signature is effectively an utterance of the request by the service requester, who is Tony. For example, when a request r arrives over this SSL transport, Carol interprets the request as follows:

Tony **says** *r*

CSIv2 allows a request to have a more complex nature, which may contain other principals. There are potentially four principals who participate in any given request: (1) the target service provider (*Carol*), (2) a transport principal (*Default* or *Tony*) who delivers the request to *Carol*, (4) a client to be authenticated (*Clyde*), and (4) a principal (*Alex* or *Anon*) whose name appears in an identity token.

To understand the purpose of the named principals Clyde, Alex, and Anon, it is helpful to first understand the structure of CSIv2 requests. These requests may include a Security Attribute Service (SAS) data structure, which holds userid/password pairs and identity tokens. The SAS data structure can be viewed as a three-tuple $\langle CA, IA, SA\rangle$, with the following components:

- *CA* (if present) contains client authentication information, such as userid and passwords using the General Security Services Userid Password (GSSUP) client authentication mechanism, which is also defined in CSIv2.
- *IA* is an identity assertion, which (if present) identifies the client of the request.
- *SA* contains any applicable security attributes, which indicate various privileges that the relevant principals may have.

The authors of the CSIv2 protocol have not yet standardized on authorization information. Therefore, we focus on the aspects of the CSIv2 protocol that only deal with identity and identity delegation, which uses the *CA* and *IA* components along with the transport identity. There are six possible pairs $\langle CA, IA\rangle$ to consider. The *CA* component may either be empty (in which case no principal is associated with *CA*), it may contain a userid/password pair $\langle Clyde, PW\rangle$ (in which case the principal *Clyde* is associated with *CA*). There are three possibilities for the *IA* component: it may be empty, it may contain an identity token for a particular principal *Alex,* or it may contain a special value, *Anon,* indicating the anonymous principal.

When the target provider Carol receives a request *r* with an associated SAS data structure, she always interprets it relative to the same ordering on principals: the transport principal (either Default or Tony) first, followed by (if present) the principal associated with *CA* (Clyde), followed by (if present) the principal associated with *IA* (Alex). Thus, if an Tony (using SSL) delivers a request *r* that contains a SAS structure indicating Clyde as the *CA* principal and Alex as the *IA* principal, then Carol interprets this request as:

(Tony **says** (Clyde **says** (Alex **says** *r*))), and she considers Alex to be the invocation principal (i.e., client). In contrast, a TCP/IP delivered request *r* that identifies Clyde as the *CA* principal and contains no *IA* component is interpreted as

(Default **says** (Clyde **says** *r*)),

and Clyde is the invocation principal. Table 1 enumerates the twelve cases for requests that are made in association with the SAS data structure over the two transports.

Transport	***Transport Principal***	***CA Principal***	***IA Principal***	***Invocation Principal***	***Carol's Interpretation***
TCP/IP	Default	None	None	Default	Default says r
TCP/IP	Default	Clyde,PW	None	Clyde	Default says Clyde says r
SSL	Tony	None	None	Tony	Tony says r
SSL	Tony	Clyde,PW	None	Clyde	Tony says Clyde says r
TCP/IP	Default	None	Alex	Default	Default says Alex says r
TCP/IP	Default	Clyde,PW	Alex	Clyde	Default says Clyde says Alex says r
SSL	Tony	None	Alex	Tony	Tony says Alex says r
SSL	Tony	Clyde,PW	Alex	Clyde	Tony says Clyde says Alex says r
TCP/IP	Default	None	Anon	Default	Default says Anon says r
TCP/IP	Default	Clyde,PW	Anon	Clyde	Default says Clyde says Anon says r
SSL	Tony	None	Anon	Tony	Tony says Anon says r
SSL	Tony	Clyde,PW	Anon	Clyde	Tony says Clyde says Anon says r

Table 1: Twelve Cases for Requests

The fourth case in the table and the associated statement Tony **says** Clyde **says** r can be used to describe a situation in which a bank customer (Clyde) connects to a web server in order to perform an online-banking operation. The web server passes on Clyde's userid, password, and request *r* to an online-banking server (Carol) over SSL. The web server uses SSL to authenticate as Tony. The CSIv2 protocol specifies the way in which the information is to be interpreted by the target principal. In addition, CSIv2 describes the trust relationships that are necessary for Carol to conclude that Clyde really is making the request *r* and that Tony is really acting on Clyde's behalf as his delegate.

Because delegated authority must be carefully controlled and limited to prevent fraud, reasoning about delegation must be done precisely, accurately, and consistently. To meet these needs we use a specialized calculus and modal logic for reasoning about brokered requests and delegations. In the next section, we give an overview of this logic; we then revisit the banking scenario in Section 4.

3 Calculus for Reasoning about Brokered Requests

A logic introduced by Abadi, et al [LABW92,ABLP93,WABL94] is the basis for reasoning about authentication and access control many protocols and systems, such as the TAOS operating system [WABL94]. The authors of the CSIv2 protocol also took this work into account during the formulation of CSIv2. The calculus of principals describes the ordering relationships amongst principals while a modal logic provides a mechanism by which one can reason about principals and the statements that are attributed to them. We use an algebra of binary relations that forms a multiplicative semi-lattice semi-group to model the calculus semantics.

3.1 Calculus of Principals

The calculus describes well formed formulas that make up structured principals. To begin, we assume a countable set Name that contains atomic principal names, ranged over by the meta-variable A. We define a set of principal expressions, **PrinExp**, ranged over by the variables, P and Q, The abstract syntax of **PrinExp** is defined by the following grammar:

$$P \coloneqq A \;\mid\; P \,|\, Q \;\mid\; P \wedge Q \;\mid\; P \;\mathbf{for}\; Q$$

The operators, $|$, $\wedge$, and **for**, denote an intuitive meaning for the simple constructions. The construction $P|Q$ denotes that P is attributing something to Q, otherwise read as "*P quoting Q*". The construction $P \wedge Q$ denotes that P and Q agree. P **for** Q denotes that P is somehow authorized to attribute something to Q, otherwise read as "*P on behalf of Q*". We like to think that $P \wedge Q$ is stronger than just P or Q alone due to consensus, and P **for** Q is stronger than $P|Q$ because of the authorization. To support this intuition, we use a multiplicative semi-lattice semi-group (MSS) for the semantics.

A multiplicative semi-lattice semi-group is a mathematical structure, $<S, \frown, ||>$, in which a set, S, is organized with the two binary operators $\frown$ and $||$. The $\frown$ operator must be idempotent, associative, and commutative. The $||$ operator must be associative and must distribute over the $\frown$ operator in both arguments. Specifically, for all elements x, y, z $\in$

S, the following equations hold:

$$
\begin{aligned}
x \triangle x &= x \\
x \triangle y &= y \triangle x \\
x \triangle (y \triangle z) &= (x \triangle y) \triangle z) \\
x \parallel (y \parallel z) &= (x \parallel y) \parallel z \\
x \parallel (y \triangle z) &= (x \parallel y) \triangle (x \parallel z) \\
(y \triangle z) \parallel x &= (y \parallel z) \triangle (z \parallel x)
\end{aligned}
$$

The $\triangle$ operator induces a partial ordering on the set. The $\triangle$ operator is conventionally called the *meet* or *greatest lower bound* of its arguments. The $\triangle$ operator induces an ordering, $\leq$, on S such that $x \leq y$ if and only if $x = x \triangle y$.

A common MSS is the set of binary relations, S_R, over the elements of a set, S, that is closed under union, $\cup$, and relational composition, $\circ$. It is straightforward to verify that the structure $\langle S_R , \cup , \circ \rangle$ satisfies the MSS axioms. In this case, the $\leq$ ordering is the superset relation, $\supseteq$. Given $r_1, r_2 \ni S_R$, $r_1 \leq r_2$ is true if and only if $r_1 = r_1 \cup r_2$, is true, which is precisely when $r_1 \supseteq r_2$. This MSS gives us nice properties for manipulating principals.

Specifically, assume we have a set W, and a subset of binary relations, W_R, i.e. a subset of $W \times W$, such that the structure $\langle W_R , \cup , \circ \rangle$ satisfies the MSS axioms. Also assume that we have an interpretation function J that maps elements of **Name** to specific elements in W. Then it is straightforward to extend the function J to a function $\bar{J}$ encompassing principal expressions over the operators $\wedge$ and $|$ as follows:

$$
\begin{aligned}
\bar{J}(A) &= J(A) \\
\bar{J}(A \wedge B) &= \bar{J}(A) \triangle \bar{J}(B) = \bar{J}(A) \cup \bar{J}(B) \\
\bar{J}(A \mid B) &= \bar{J}(A) \parallel \bar{J}(B) = \bar{J}(B) \circ \bar{J}(A) \\
&= \{ (w, w'') \mid \exists\, w'.(w, w') \in \bar{J}(B) \,\&\, (w', w'') \in \bar{J}(A) \}
\end{aligned}
$$

This formulation induces an ordering on the interpretations of principal expressions, such that $\bar{J}(A) \leq \bar{J}(B)$ if and only if $\bar{J}(A) \supseteq \bar{J}(B)$. Furthermore, we find that our intuitions about the strength of principals constructed with the $\wedge$ operator are correct.

We extend the notation of $\leq$ to principal expressions such that $A \leq B$ if and only if $\bar{J}(A) \leq \bar{J}(B)$ holds. We see that the ordering $(A \wedge B) \leq A$ is preserved, which is to say that $(A \wedge B)$ is *below* A in the MSS, and therefore $(A \wedge B)$ can be thought of as stronger than A itself.

What about the **for** operator? Abadi suggests that the expression A **for** B can be thought of as the equivalent of $(A|B) \wedge (D|B)$ where D is a (fictional) distinguished principal that validates *A*'s authorization to act on behalf of *B*. We can easily see that our intuition about the **for** operator is correct as well, i.e. $(A \textbf{ for } B) \leq (A|B)$ also holds.

Although, it is suggested that the principal *D* is a fictional entity, it is beneficial as a real entity. We will see how this third principal plays out in CSIv2 user name and password authentication as a real entity that verifies user names and password combinations.

3.2 Modal Logic

In this section, we introduce a modal logic to reason about principals and the way in which they act. We use a logic of *belief*. We reason about what a requesting principal believes his authorizations are such that he can access and perform actions on an object. The logic encompasses classical propositional logic prefixed with modal operator, which is **says.** A logical formula, *P* **says** φ, where φ is a formula in the logic or in propositional logic, and *P* is a principal, means that P *believes* φ and that P can *say* or does *say* φ. The alternative intuitive notion is that if *P* says φ, then *P* believes (or knows) that φ is true to the best of his knowledge. We use the terms *believes* and *says* interchangeably in our discussion as appropriate, but for simplicity we stick with **says** as the formal operator. The other operator we use is **speaksfor**. Intuitively, the logical formula, *P* **speaksfor** *Q,* means that anything that *P* believes is also believed by *Q*, and therefore whatever *P* says, *Q* says also.

We present an abstract syntax for creating formulas in this logic. We first assume a countable set of propositional variables, **PropVar**, ranged over the meta-variable *p*. The two operators **speaksfor** and **says** along with the logical operators from classical logic make up a set, **LogExp**, of formulas, ranged over by meta variable φ. Finally, along with the two meta-variables *P* and *Q*, which range over the set **PrinExp**. the abstract syntax for formulas in **LogExp** are as follows:

$$
\begin{aligned}
\varphi ::== \; & p \\
& |\; P \textbf{ speaksfor } Q \\
& |\; \sim\varphi \\
& |\; \varphi_1 \;\&\; \varphi_2 \\
& |\; \varphi_1 \text{ or } \varphi_2 \\
& |\; \varphi_1 \supset \varphi_2 \\
& |\; \varphi_1 \Leftrightarrow \varphi_2 \\
& |\; P \textbf{ says } \varphi
\end{aligned}
$$

Note that principal expressions may be included as part of a logical formula, but a principal expression by itself is not a formula. We will see that, semantically, the formula *P* **speaksfor** *Q* holds when $P \leq Q$ in the underlying calculus of principals. It follows that

speaksfor is monotonic with respect to both $\wedge$ and $|$ (quoting).

To complement the abstract syntax we introduce a collection of logical rules for manipulating logical expressions in this logic. We introduce a derivability predicate, $\vdash$, such that $\varphi_1, ..., \varphi_n \vdash \varphi$ is true if φ is derivable from $\varphi_1, ..., \varphi_n$ using the logical rules. These logical rules are as follows:

1. Any tautological statement, φ, in classical propositional logic is vacuously derivable.
   $\vdash \varphi$
2. Modus Ponens.
   If $\vdash \varphi_1$ and $\vdash \varphi_1 \supset \varphi_2$, then $\vdash \varphi_2$
3. Any principal believes anything that is derivable.
   If $\vdash \varphi$, then $\vdash P$ **says** φ, for all *P*.
4. If *P* believes an implication statement, $\varphi_1 \supset \varphi_2$, then if *P* does believe φ_1, then *P* will believe φ_2 as well.
   $\vdash (P \textbf{ says } (\varphi_1 \supset \varphi_2)) \supset ((P \textbf{ says } \varphi_1) \supset (P \textbf{ says } \varphi_2))$
5. A consensus of principals believes a statement if and only if each one of them believe the same statement.
   $\vdash ((P \wedge Q) \textbf{ says } \varphi) \Leftrightarrow ((P \textbf{ says } \varphi) \; \& \; (Q \textbf{ says } \varphi))$
6. If one principal *P* believes what another principal *Q* says, then *P* can quote *Q* on the same statement.
   $\vdash ((P | Q) \textbf{ says } \varphi) \Leftrightarrow (P \textbf{ says } (Q \textbf{ says } \varphi))$
7. If one principal, *P*, speaks for another principal, *Q*, then if *P* believes a statement, then *Q* believes (or says) the same statement.
   $\vdash (P \textbf{ speaksfor } Q) \supset ((P \textbf{ says } \varphi) \supset (Q \textbf{ says } \varphi))$, for all φ

Using the MSS of binary relations we introduced above for the semantics of principal expressions and standard modal logic interpretations for logical expressions we can reason about the consistency of our logic.

We use *Kripke structures* to provide interpretations for the logical expressions. A Kripke structure $M = \langle W, w_0, I, J \rangle$ is a structure that organizes possible worlds. In this structure, *W* is a set of worlds. A *world* is a collection of truths or false hoods of which a principal may have knowledge. Furthermore, there is a question of consistency between his knowledge and whether he believes he may reside in certain worlds. For example, if a principal Alice can see that it is dark outside, and only two worlds, night and day, exist, then she believes that the only possible world in which she may reside is night. In contrast, if Bob is blind, he believes he may reside in either night or day.

Returning to the Kripke structure, *I* is a function that maps a propositional variable to a subset of *W*, and *J* is a function that maps a principal name to a subset of $W \times W$.

Intuitively, *I(p)* is the set of all possible worlds in which *p* is said to be true. Note, that means that *p* is false in the other worlds. *J(P)* is a binary relation between the worlds that the principal *P* cannot determine are different based on his knowledge. This function determines the knowledge of *P* based on the propositions that he knows to be true or false. The element w_0 is a distinguished element of *W*, which is said to be the "starting" world. This element in the structure intuitively says to us, that if Bob is actually in the night world, from there, he may believe he is also in the day world, and visa versa. On the contrary, if Alice is actually in the night world, she cannot believe she is in the day world, by virtue of her sight, i.e. knowledge of the "is dark outside" proposition. Note, one must be careful with this intuition for an arbitrary principal. There is no guarantee that *J(P)* relation is symmetric, reflexive, or transitive, as there is no expectation that an arbitrary principal is rational.

To formalize the semantics, we define a meaning function $\varepsilon_M : LogExp \rightarrow 2^W$ that gives us a set of worlds in which a formula in our logic is true in a particular Kripke structure, *M*:

$$\begin{array}{rcl}
\varepsilon_M[\![p]\!] & = & I(p) \\
\varepsilon_M[\![\neg\varphi]\!] & = & W - \varepsilon_M[\![\varphi]\!] \\
\varepsilon_M[\![\varphi_1 \,\&\, \varphi_2]\!] & = & \varepsilon_M[\![\varphi_1]\!] \cap \varepsilon_M[\![\varphi_2]\!] \\
\varepsilon_M[\![\varphi_1 \text{ or } \varphi_2]\!] & = & \varepsilon_M[\![\varphi_1]\!] \cup \varepsilon_M[\![\varphi_2]\!] \\
\varepsilon_M[\![\varphi_1 \supset \varphi_2]\!] & = & \varepsilon_M[\![\neg\varphi_1]\!] \cup \varepsilon_M[\![\varphi_2]\!] \\
\varepsilon_M[\![\varphi_1 \Leftrightarrow \varphi_2]\!] & = & \varepsilon_M[\![\varphi_1 \,\&\, \varphi_2]\!] \cup \varepsilon_M[\![\neg\varphi_1 \,\&\, \neg\varphi_2]\!] \\
\varepsilon_M[\![P \text{ says } \varphi]\!] & = & \{w \mid \mathcal{J}(P)\, w \subseteq \varepsilon_M[\![\varphi]\!]\} \\
 & = & \{w \mid \{w' \mid (w, w') \in \mathcal{J}(P)\} \subseteq \varepsilon_M[\![\varphi]\!]\} \\
\varepsilon_M[\![P \text{ speaksfor } Q]\!] & = & \left\{\begin{array}{c} W, \textit{if } \mathcal{J}(P) \supseteq \mathcal{J}(Q) \\ \emptyset, \textit{otherwise} \end{array}\right\}
\end{array}$$

We write $M, w \models \varphi$ provided that $w \in \varepsilon_M[\![\varphi]\!]$. We write $M \models \varphi$ provided that $M, w_0 \models \varphi$; this relationship is pronounced "*M* satisfies φ". We say that φ is valid if φ is satisfied in all Kripke structures; in such cases, we write $\models \varphi$.

Our logical rules are *sound* with respect to the Kripke semantics: for every formula φ, $\vdash \varphi$ implies that $\models \varphi$. Therefore, every formula that is derived from the logical rules is satisfied in all Kripke structures. However, the logical rules are not *complete; th*ere are formulas that are valid, but are not yet derivable from the rules. We do not consider this situation a problem.

4 Formal Analysis of a CORBA CSIv2 Request

We return to the online-banking scenario, changing the names of the actors for expository purposes.

> A banking customer, Bob, connects to a banking web server (WB) to pay a bill. The web server requires Bob to supply his userid and password, ($\langle Bob, pwd\rangle$), which the web server passes along with Bob's bill payment request (*r*), to a banking server (S) via a CORBA CSIv2 request.

In terms of Table 1, WB is playing the part of Tony, because it authenticates to the banking server via an SSL public key certificate. Bob is cast as Clyde because Bob's userid and password are passed in the client authentication field of the CSIv2 request. The banking server is the target service provider, and thus serves as Carol. There is no identity token included in the request, so this scenario conforms to the fourth line in the Table 1.

The banking server enforces an access policy related to the particular requests. Although, access control policies can be of arbitrary granularity, for simplicity, let's assume that the request *r* is protected by an access policy of only allowing Bob to perform *r*. The banking server knows that Bob can never perform the request directly, and he must perform this operation through a proxy server, namely WB. However, the banking server wants more than WB merely stating that it is making the request on behalf of Bob. The banking server wants a policy that states that WB is authorized to make the request on Bob's behalf. Therefore, the banking server has an access-control policy that allows WB $\mathbf{for}_S$ Bob to perform *r*. In other words, *r* can be performed only if S determines that WB is working on Bob's behalf.

The banking server interprets the request *r* as two pieces of information:

$$WB \textbf{ says } Bob \textbf{ says } r \qquad (1)$$
$$WB \textbf{ says } \langle Bob, pwd\rangle \textbf{ speaksfor } Bob \qquad (2)$$

Thus, from the banking server's perspective, WB is making two claims: (1) that Bob wants to perform the action *r*, and (2) that the userid-password pair belongs to Bob.

Remember the access policy that *r* can be performed only if S determines that WB is working on Bob's behalf. Recall that, in the Abadi calculus, WB **for** Bob is syntactic sugar for a principal $\mathrm{WB}|\mathrm{Bob} \wedge \mathrm{D}|\mathrm{Bob}$, where D is a (fictional) delegation authority. In an open system, however, there may be several such (possibly nonfictional) authorities. Thus, we generalize this approach by subscripting the **for** operator with the name of a specific authority, in this case the banking server S itself, such that

$$\mathrm{WB}|\mathrm{Bob} \wedge \mathrm{S}|\mathrm{Bob} = \mathrm{WB}\ \mathbf{for}_{\mathrm{S}}\ \mathrm{Bob}. \tag{3}$$

The online-banking server is prepared to believe that WB is working on Bob's behalf, provided that its trusted password server (PS) verifies the pair $\langle Bob, pwd\rangle$. Thus, the banking server is governed by the following password verification rule:

$$((\mathrm{WB} \wedge \mathrm{PS})\ \mathbf{says}\ (\langle Bob, pwd\rangle\ \mathbf{speaksfor}\ \mathrm{Bob})) \supset (\ \mathrm{WB}|\mathrm{Bob}\ )\ \mathbf{speaksfor}\ (\ \mathrm{S}|\mathrm{Bob}\ ) \tag{4}$$

Notice that, implicitly, the banking server believes that the only way WB would have Bob's password is if Bob provided it to allow WB to make requests on Bob s behalf. The banking server passes the userid-password pair on to the password server, which confirms the validity of that pair. Thus, from the perspective of the online-banking server,

$$\mathrm{PS}\ \mathbf{says}\ (\langle Bob, pwd\rangle\ \mathbf{speaksfor}\ \mathrm{Bob}). \tag{5}$$

Combining the pieces of information (2) and (5), S determines that

$$(\mathrm{WB} \wedge \mathrm{PS})\ \mathbf{says}\ (\langle Bob, pwd\rangle\ \mathbf{speaksfor}\ \mathrm{Bob}).$$

Therefore, using the password rule (4), the banking server deduces that

$$\mathrm{WB}|\mathrm{Bob}\ \mathbf{speaksfor}\ \mathrm{S}|\mathrm{Bob}\ .$$

Because **speaksfor** is a monotonic operator, this fact yields

$$\mathrm{WB}|\mathrm{Bob}\ \mathbf{speaksfor}\ (\mathrm{WB}|\mathrm{Bob} \wedge \mathrm{S}|\mathrm{Bob}),$$

and along with (3) we surmise

$$(\ \mathrm{WB}|\mathrm{Bob}\ )\ \mathbf{speaksfor}\ (\mathrm{WB}\ \mathbf{for}_{\mathrm{S}}\ \mathrm{Bob}).$$

Combining this result with the original request (1), the banking server can deduce that

$$\mathrm{WB}\ \mathbf{for}_{\mathrm{S}}\ \mathrm{Bob}\ \mathbf{says}\ r.$$

In accordance with its access-control policy for request *r*, the online-banking server grants the request.

In contrast, let's assume that WB only includes Bob's userid in the CSIv2 Identity Token in the request *r*, which relates to line 7 of Table 1. The banking server interprets the request as:

$$\text{WB } \textbf{says} \text{ Bob } \textbf{says } r.$$

According to our logic, namely logical rule 6, we can only deduce the following:

$$\text{WB}|\text{Bob } \textbf{says } r.$$

Hence, the requesting principal, $\text{WB}|\text{Bob}$ is not quite strong enough to satisfy the access policy of "WB $\textbf{for}_S$ Bob" for *r*, and the banking server denies the request. Note, it turns out that it must not be the case that $\text{WB} \leq \text{S}$. Otherwise, WB $\textbf{for}_S$ B would have the equivalent interpretation as $\text{WB}|\text{Bob}$. This restriction makes sense as we do not want the proxy (web server) to be stronger than the banking server itself.

In general, if an access control system labels a potential request with an access principal of Q, and P is said to make the request, the access control system must be able to deduce that $P \leq Q$ holds within the logic in accordance with any auxiliary policy rules in order for the request to be granted.

Supplying a userid-password credential is not the only way to deduce WB $\textbf{for}_S$ Bob using CSIv2. Bob may supply WB with other credentials in the SAS layer, such as a signed delegation certificate [WALB94] that authorizes WB to make requests on Bob's behalf. Therefore, the access control policy of allowing WB $\textbf{for}_S$ Bob remains valid, but the credentials and rules to deduce that principal may vary. Kuo and Humenn have shown the way such credentials may be used in dynamically authorizing the use of roles [KH02], but they have not yet formally analyzed the approach.

One important idea we find here, is that the security related information provided in the CSIv2 request and the access control policy on the resource alone is not enough to reason about access control. We see that the password verification rule (4), which is not actually contained in the access policy on request *r* itself, is an important part of the access control policy for the banking server. It is also important to note that, the rule (4) was investigated and formulated by analysis of CSIv2 userid-password verification using our logic. The question we asked is: *How do we formalize and reason about userid-password verification?* In general, one must be diligent in formalizing the rules governing the access control policies to enable the system as well as to protect the resources.

5 Conclusions

Security protocols such as CSIv2 are inherently complicated. For example, the systems using CORBA and CSIv2 must address the following fundamental question: *How does a CORBA based service provider deal with the security of brokered requests?* Making precise interpretations about the requests and subsequently determining the correct access-control decision require a logical system that supports reasoning about the requests, the principals who make the requests, and the presence or absence of trust relationships among those principals. We have shown that the logic of Abadi and colleagues provides a good basis for this purpose and we have expanded on its use with CSIv2.

Along the way, we have introduced a rule that formalizes the notion of userid-password credentials verification, which is known in CSIv2 as GSSUP (General Security Services Userid Password), and we have shown the way in which the formal rule must come into play in determining an access decision.

Finally, we cannot stress enough the need for formal methods. The Abadi logic was influential in the design the CSIv2 protocol as the first author of this paper (who was also a coauthor of the CSIv2 standard) will attest. The identity-assertion layer of the Secure Attribute Service (SAS) exists as a direct result of using the calculus to describe and interpret requests. Likewise, the inclusion of the security-attribute layer in the SAS arose from an analysis using the logic: the analysis demonstrated that the SAS data structure must provide a way to transport additional evidence (in the form of credentials) that a broker Bob is working on behalf of a principal Alice, rather than merely quoting Alice. Finally, Table 24 of the CSIv2 standard [CSI01] describes how the SAS data structure should be interpreted: the trust rules that appear at the bottom of that table were born out of an analysis using the Abadi logic.

Now that we have a precise formalization for reasoning about access control using CSIv2 and userid-password verification, it allows us to mechanize the analysis of access control policy with formal method tools, such as logic engines, and theorem provers. This approach leads to better assurance of system security.

6 References

[ABLP93] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. ACM Transactions on Programming Languages and Systems, 15(4):706 734, September 1993.

[COR98] CORBAServices: Common Object Services. Technical Report formal/98-07-05, Object Management Group, July 1998. Available via http://www.omg.org/cgi-bin/doc?formal/98-07-05.

[CSI01] The common secure interoperability version 2. Technical Report ptc/01-06-17, Object Management Group, June 2001. Available via http://www.omg.org/cgi-bin/doc?ptc/01-06-17.

[KH02] Joncheng Kuo and Polar Humenn. Dynamically authorized role-based access control for secure distributed computation. In Proceedings of the ACM Workshop on XML Security, 2002.

[LABW92] Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wobber. Authentication in distributed systems: Theory and practice. ACM Transactions on Computer Systems, 10(4):265 310, November 1992.

[WABL94] Edward Wobber, Martin Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. ACM Transactions on Computer Systems, 12(1):3 32, February 1994.

Appendix: Symbols used in this paper.

Symbol	***Example***	***Pronunciation***	***Explanation***
Algebraic and Set Theory Symbols			
⌓	a ⌓ b	a *“meet” b*	The ⌓ operator represents the greatest lower bound of elements a and b.
$\Vert$	$a \Vert b$	a “double bar” b	The $\Vert$ operator merely represents an associative relation between the elements a and b that distributes over applications of the ⌓ operator.
$\leq$	$a \leq b$	a “is below” b	The element a is below the element b in the lattice, i.e. $a = a$ ⌓ b.
$\circ$	$R_1 \circ R_2$	R_1 “composed with” R_2	Represents the relation (a,z) such that there exists an element b such that $R_1(a,b)$ & $R_2(b,z)$ holds.
$\subseteq$	$S_1 \subseteq S_2$	S_1 “subset” S_2	Every element of the set S_1 is in the set S_2.
$\supseteq$	$S_1 \supseteq S_2$	S_1 “superset” S_2	Every element of the set S_2 is in the set S_1.
$\in$	$a \in S$	a “member of” S	The element a is a member of the set S.
$\cup$	$S_1 \cup S_2$	S_1 “union” S_2	Represents the set containing every element of the set S_1 and the set S_2 together.
$\cap$	$S_1 \cap S_2$	S_1 “intersection” S_2	Represents the set containing only those elements of the set S_1 that are also in the set S_2.
Logical Symbols			
&	φ_1 & φ_2	φ_1 “and” φ_2	Both φ_1 and φ_2 are true.
or	φ_1 or φ_2	φ_1 “or” φ_2	One, or both of φ_1 and φ_2 are true.
$\supset$	$\varphi_1 \supset \varphi_2$	φ_1 “implies” φ_2	If φ_1 is true, then φ_2 is true.
$\Leftrightarrow$	$\varphi_1 \Leftrightarrow \varphi_2$	φ_1 “if and only if” φ_2	If φ_1 is true, then φ_2 is true, and if φ_2 is true, then φ_1 is true.
$\neg$	$\neg\varphi$	“not” φ	The statement φ is false.
$\vdash$	$\varphi_1, ..., \varphi_n \vdash \varphi$	φ “is derivable from” $\varphi_1, ..., \varphi_n$	The statement φ is derivable from the statements $\varphi_1, ..., \varphi_n$.
Principal Calculus Symbols			
says	P **says** s	P “says” s	The principal P makes a statement s, which also means P believes the statement s is true.
$\mid$	$P \mid Q$ **says** s	P “quoting” Q says s	The principal P believes Q makes the statement s.
$\wedge$	$P \wedge Q$ **says** s	P “and” Q says s	The principal P and Q together make the statement s.
for	P **for** Q **says** s	P “on behalf of” Q says s	The principal P is authorized to speak in Q's behalf.
speaksfor	P **speaksfor** Q	P “speaks for” Q	Whatever P says, Q says also.