A Formal Semantics for the Logic of Authentication

Jim Waldo
Sun Microsystems Laboratories
1 Network Drive
Burlington, MA 01803
Jim.waldo@sun.com

Abstract

The article Authentication in Distributed Systems: Theory and Practice[1] has become a standard work in security, offering a formal theory that can be used to think about systems that claim to authenticate principals in a distributed system. However, the formal parts of the paper are presented entirely axiomatically. This presentation allows easy translation to proof systems that run on a computer, but lends itself less easily to comprehension by the human user of the system.

To aid in human comprehension, we have developed a model for the logic that can be seen as giving a formal semantics to the system. We begin by discussing some of the reasons for constructing such a model, and then work through both the basic authentication logic and each of the extensions that are added to give a full authentication logic. While the model for the basic logic is fairly straightforward, extending the model to cover the full logic proves both challenging and instructive, as it shows some of the implications of the extensions that are not so easily seen in the axiomatic approach.

Introduction

In the article Authentication in Distributed Systems: Theory and Practice Lampson et. al. provide a formal language for thinking about authentication. This language is used to evaluate the practical systems that are constructed to do authentication; arguments in the formal language are used to justify design decisions made in the practical system.

The logic that is presented is axiomatic in nature, that is, the language is defined by a syntax and a number of axioms that are presented allowing the syntactic transformation of one set of statements in the language to another set of statements. Such a system, which relies purely on syntactic manipulation, is very easy for a computer to understand and to build programs that a computer can use to construct proofs. As such, a syntactic approach is often favored when the result of the language is meant to be the base for some programs that will be constructing proofs in the language.

However, for humans such systems are often hard to understand unless the languages are given an interpretation, that is, unless the language is somehow tied to something that people find meaningful. Lampson does give an intuitive interpretation of the language in the paper, explaining in an informal fashion what the intended meaning are for the various syntactic statements. However, these explanations are explicitly made only to help the intuitions; there is nothing formal in the interpretation that lets anyone construct any arguments in the logic of authentication based on the interpretation.

It is also possible to give a formal model or semantics\(^1\) for the language. Such a semantics allows us to talk about the meaning of the statements in the language, and allows humans (who are much better at dealing with meaning than with syntax) to understand the language in which the meaning is expressed. Further, constructing such a semantics often allows clarification in the implications of the syntactic rules for the language, and can show subtle differences between what those rules actually state and the informal interpretations they were meant to capture.

---

\(^1\) We will make clear the distinction between a formal model and the semantics of a language below, but at this point we will follow common custom and use the two terms (nearly) interchangeably.
In what follows, I will present a formal semantics of the language for authentication presented by Lampson. We will start by reviewing what a formal semantics is, and why such constructs are useful. We will then present the syntax of the language; this will look familiar to anyone who has read the paper even though it will be presented in a slightly different order. We will then construct a formal semantics for the language in a series of stages. We will start with the core of the language, and then add on to the semantics the extensions to the core having to do with delegation, roles, and groups. Finally, we will discuss the addition of hierarchical names and certificate authorities. In the end, we will have a complete semantics for the complete logic.

We begin with a characterization of a formal semantics and a model. A formal semantics is a mapping from the syntactic elements of a language to the meanings of those elements. The usual strategy is to state what the basic entities which constitute meaning are, and then establish a mapping from all of the sentences of the language to those meanings. This is generally started by picking some mapping from the primitive syntactic elements of the language to some other set of primitives. Once that is done, a semantic rule is given for each syntactic rule of the language, showing how the meaning of the output of the syntactic rule is determined by the meanings of the elements that were combined by the syntactic rule. The goal is to show how some basic element of the language (for example, a sentence) obtains its meaning (for example, its truth value) by the way in which the component parts are put together and the meaning of those component parts.

A model is a semantics (the set of rules for building meanings out of other meanings) and a domain of discourse, that is, a set of entities to which we can map the basic entities of the language and which obey the combination rules of the semantics.

This is actually not as complex as it sounds; those of you who remember truth tables from logic may be surprised to know that those tables provide a kind of formal semantics for the propositional logic. That logic has as syntactic primitives variables that represent sentences and are interpreted as mapping to the values \{T,F\} (or true and false). The semantics of the more complex sentences are given by the truth tables; if you have a sentence of the form A \lor B then it will be T if A is T or B is T, and F if both are F. The truth table represents the evaluation function for the semantics. The model for such a system is the combination of the domain of discourse (the truth values T and F) and the truth-table rules.

For a simple language that only had a set of syntactic rules, this is all that would be required to establish both a semantics and a model. However, most languages of interest specify not only a set of syntactic formation rules, but also a set of axioms that allow transforming some set of sentences in the language to some other set. When giving a formal model or semantics for such languages, it is also necessary to show that all of the axioms of the syntactic system are true in the set of semantic interpretations that have been supplied. If that can be done, then the semantics is complete.

There are lots of reasons to give semantics in this way. For example, if you can show that a semantic model exists for a language, and that the semantic model is itself consistent, then you have shown that the language is consistent. More than that, a semantics can sometimes illustrate why the axioms of the language are true, by mapping the axioms to some domain in which our intuitions are easier to come by.

**The basic logic of authentication**

Lampson begins by building a language in which there are principals (intuitively identified as entities that make statements), statements, and the relation of *saying* (which holds between a principal and a statement) and *speaking for* (which holds between a principal and a principal). This part of the language, which we will call \(L_A\), is defined as:

**Syntax of \(L_A\):**

**Principals:**

The set of principals, \(P\), is defined as

1) an infinite set of simple principals \(A, B, C, A_1, A_2, \ldots \in P\);

2) If \(A, B \in P\), then \(A \land B \in P\), \(A \lor B \in P\);
The set of statements of $L_A$, $S$, are defined as

1) an infinite set of atomic statements $s, s_1, s_2, \ldots \in S$;
2) if $s_1, s_2 \in S$, then so are $s_1 \land s_2, s_2 \lor s_1, s_1 \equiv s_2$;
3) if $A \in P$ and $s_1 \in S$, then A SAYS $s_1 \in S$;
4) if $A, B \in P$ then $A \Rightarrow B \in S$, $A = B \in S$.

The set of principals is defined as simple principals, and those formed by conjoining two other principals with the “$\land$” operation and by those formed with the quoting operator “$|$”. We can form statements from the atomic statements by using the usual logical operations of conjunction (“$\land$”), implication (“$\Rightarrow$”), and the biconditional (“$\equiv$”). We can also make a statement out of joining a principal with a statement using the SAYS operation, and by joining two principals with the speaks-for (“$\Rightarrow$”) operator.

Along with the syntactic rules that form these statements, the logic defines a set of axioms. A meta-convention in the language is that we indicate axioms or statements that can be proved by the axioms by preceding the statement with the symbol “$|$”. For this fragment of the logic of authentication, the axioms concerning statements are

If $s$ is an instance of a theorem of propositional logic then $| \vdash s$. (S1)
If $\vdash s$ and $\vdash s\supset s'$ then $\vdash s'$. (S2)
$\vdash (A \text{ SAYS } s \land A \text{ SAYS } (s \supset s')) \supset A \text{ SAYS } s'$. (S3)
If $\vdash s$ then $\vdash A \text{ SAYS } s$ for every principal $A$. (S4)
$\vdash A \text{ SAYS } (s \land s') \equiv (A \text{ SAYS } s) \land (A \text{ SAYS } s')$. (S5)

Axiom S1 imports all of the theorems of propositional logic that can be expressed in this language into the language. Likewise, S2 simply imports the rule of modus ponens into the system. Axiom S3 extends modus ponens to statements that occur inside the SAYS operation; it requires that principals say all of the things that are entailed by anything that those principals say. The final axiom allows moving from any axiom or theorem to the claim that all principals say that axiom or theorem. S5 allows the distribution of the SAYS operation over the conjunction operation.

There is also a set of axioms concerning principals; these are

$\vdash (A \land B) \text{ SAYS } s \equiv (A \text{ SAYS } s) \land (B \text{ SAYS } s)$. (P1)
$\vdash (A \mid B) \text{ SAYS } s \equiv A \text{ SAYS } (B \text{ SAYS } s)$. (P2)
$\vdash \land$ is associative, commutative, and idempotent (P4)
$\vdash \mid$ is associative (P5)
$\vdash \mid$ distributes over $\land$ in both arguments (P6)
$\vdash (A \Rightarrow B) \equiv (A = A \land B)$. (P7)
$\vdash (A \Rightarrow B) \supset ((A \text{ SAYS } s) \supset (B \text{ SAYS } s))$. (P8)
$\vdash (A = B) \equiv ((A \Rightarrow B) \land (B \Rightarrow A))$. (P9)
$\vdash (A \text{ SAYS } (B \Rightarrow A)) \supset (B \Rightarrow A)$. (P10)
$\vdash ((A \Rightarrow A) \land A \text{ SAYS } (B \Rightarrow A)) \supset (B \Rightarrow A)$. (P11)
$\vdash ((A \Rightarrow B) \land (B \Rightarrow A)) \supset (B \Rightarrow A)$. (P12)

Most of the axioms that are offered with respect to principals have intuitive interpretations that are plausible. The first axiom, P1, can be interpreted as stating that two principals jointly saying some statement is logically equivalent to both of those principals individually saying that statement. In a similar way, P2 can be taken as a definition of our ordinary notion of “quoting” by formalizing the notion that a principal quoting another principal saying some statement is the same as the first principal saying that the second principal says the statement.
However, we must be careful in ascribing interpretations to the axioms of $L_A$. While such interpretations may help us to understand the intuitions behind the language, they are not part of the language yet. All we have are axiomatic rules, which in turn only allow us to transform one set of well-formed-formulas into another set. To really get an interpretation of the language (and to see if that interpretation corresponds to the intuitions concerning authentication that we are trying to capture in our formal system) requires that we specify the interpretation of the language. It is to this that we now turn.

A Model for $L_A$

A model can now be specified that will give a range of semantics for the syntax described in the previous section. The model only gives a range because it under-specifies the evaluation function. Rather than specifying what the evaluation function gives as a meaning for every possible sentence, it specifies what the requirements are for certain forms of the sentences. In particular, it does not tell us what the evaluation of particular atomic statements or simple principals are; this is a fact about the world that can only be discovered by inspection. But given some such evaluation function (and, more importantly, given a particular mapping of principals to entities in the world and statements to truth values in the world), we can say what the truth of other sentences are that are made up of such sentences and principals.

A model, $M$, of $L_A$ is an ordered triple $M = < T, \Sigma, V>$ where
- $T$, the range of interpretation of the statements of $L_A$, is the set $\{T, F\}$,
- $\Sigma$, the range of interpretation of the principals of $L_A$, is $\{x | x \in \& (S)\}$ and
  - if $\neg s$ then $s \in x$ and
  - if $s \in x$ and $s \supset s_2 \in x$ then $s_2 \in x$

Like the propositional calculus, we are taking the meaning of sentences to be the truth values $T$ (or true) and $F$ (or false). In general semantic works, this is taken to be the "grass is green" rule, that is, the sentence "grass is green" is true if and only if grass is green\(^2\). In our calculus, we care even less about the actual truth of a sentence; what we really care about is who said it.

Principals are somewhat more interesting. We take the interpretation of a principal to be a set of statements (that is, a principal maps to a member of the power set of the set of statements in the language). Think of this as saying that, for the purposes of the meanings we are talking about, the identity of a principal is tied to the set of statements he or she makes. Not all sets of statements are to be identified with principals, however. To be identified with a principal, the particular member of the power set of statements must contain all of the theorems in the language $L_A$ and must be closed under modus ponens.

To finish up our model, we need a valuation function; this is defined as:

\[
V \text{ is a valuation function with domain } S \cup P \text{ and range } T \cup \Sigma \text{ such that}
\]
- if $A$ is a simple principal, then $V(A) \in \Sigma$;
- if $A \in P$ and $A$ has the form $B \land C$, then $V(A) = V(B) \cap V(C)$;
- if $A \in P$ and $A$ has the form $B \mid C$, then $V(A) =$
  \[
  \{x | (\exists s)(s \in V(B) \text{ and } s \text{ has the form } "C \text{ says } x" \text{ for } x \in S \text{ and } \\
  \text{if } s \in V(A) \text{ and } s' \text{ is implied by } s, \text{ then } s' \in V(A)\};
  \]

We identify simple principals with the set of statements that they say. The principal formed by conjoining two principles then becomes the intersection of what they each (individually) say. This seems straightforward.

\(^2\) This principle was first enunciated by David Lewis\cite{Lewis} in an article that is often credited with starting the endeavor of formal semantics.
Less straightforward is the interpretation of the quoting relationship. The result of quoting, remember, must be itself a principal, which means that it must be a set of statements. But the idea behind the quoting relation is that it is equivalent to "B says C says s." So this has to be a subset of the set of things that B says, and must be of the right form. So we get the interpretation above. We will talk more about this later.

Continuing with the valuation function, $V$, for sentences:

- If $s$ is an atomic statement of $S$, then $V(s) \in \{T,F\}$;
- If $s$ is of the form $s_1 \land s_2$, then $V(s) = T$ if and only if $V(s_1) = T$ and $V(s_2) = T$,
  and $F$ otherwise;
- If $s$ is of the form $s_1 \lor s_2$, then $V(s) = F$ if and only if $V(s_1) = T$ and $V(s_2) = F$,
  and $T$ otherwise;
- If $s$ is of the form $s_1 \equiv s_2$, then $V(s) = (V(s_1) = V(s_2))$;
- If $s$ is of the form $A \text{ says } s$, then $V(s) = T$ if and only if $s \in V(A)$;
- If $A, B \in P$, and $s$ is of the form $A \Rightarrow B$, then $V(s) = T$ if and only if $V(B) \supset V(A)$
  (note: here we are using the $\supset$ symbol to mean "subset", not implication; this should be clear since $V(A)$ and $V(B)$ are both sets).

Atomic statements get mapped to one of the two truth values. The usual logical operators are defined as they always are. The only statement form that is somewhat tricky is the form "A says s", but since we have defined the principal A as a set of statements, then this becomes simple set membership; "A says s" is true just in case s is in the set of statements that are the interpretation of A. Similarly, the "\Rightarrow" relationship simply becomes a subset relationship; if "A \Rightarrow B" then everything that A says is also something that B says, so the set of things said by A is a subset of the set of things said by B.

To show that this is really a model for the logic $L_A$, we need to show that all of the axioms of $L_A$ are true on all models that correspond to the restrictions that we have laid out above. The axioms for sentences, (S1) - (S4), are pretty straightforward. (S1) says that all of the sentences in $L_A$ that are theorems of propositional logic are theorems in $L_A$; since our valuation function for $\land$, $\lor$, and $\equiv$ are the same as the standard valuation functions for models of propositional logic, this is easily proved. Likewise, showing that (S2), modus ponens, holds in our model is just a case of showing that we have the same model for the appropriate connectors as is found in propositional logic, for which (S2) holds.

(S3) is the first interesting theorem from the point of view of our model. (S3) is modus ponens for the says operation, that is, it is a theorem of the language that

$$(A \text{ says } s \land A \text{ says } (s \supset s_1)) \Rightarrow A \text{ says } s_1$$

We can see that this is true in all models by looking at the way we have constructed the range of values for principals. A principal is a set of statements, which we have specified is closed under implication. A says $s$ will be true just in case $s$ is in the set that defines $A$; similarly A says $s$ just in case $s \supset s_1$ is in the set that defines $A$. But if those statements are in the set that defines $A$ and that set is closed under implication, then $s_1$ must be in the set that defines $A$ as well, so we can conclude that $A$ says $s_1$.

The final axiom concerning sentences, (S4), states that all principals say everything that is an axiom or theorem in the language. We can see that this follows in the model because of the way in which we have defined the range of the valuation function for principals; it is members of the power set of statements of the language which are closed under implication and which include all of the axioms of the language. Since every principal maps to such a set, and A says $s$ is true if $s$ is in the set that defines $A$, then it will be true of all principals $A$ that they says all axioms and theorems of the language.

The axioms concerning principals are also reasonably straightforward, for the most part. The first axiom,
\[(A \land B) \text{ says } s \equiv (A \text{ says } s) \land (B \text{ says } s)\]

is easy; since \(A\) and \(B\) are sets and the principal \((A \land B)\) is the intersection of the sets, it is pretty easy to prove that this will always be true. Similarly, \((P2)\), which states that

\[(A|B) \text{ says } s \equiv A \text{ says } B \text{ says } s\]

follows from the valuation function of \((A|B)\); it is in fact the reason that the valuation function for such constructs seems somewhat odd. In the same way, \((P3)\) follows from the way we have defined the interpretation of principals; since they are the sets of the statements that they make, it is easy to see that on all models it will be true that

\[A = B \supset (A \text{ says } s \equiv B \text{ says } s)\]

Likewise, since \(\land\) is interpreted as set intersection, it follows that it is associative, commutative, and idempotent. The interpretation of principals as sets of statements and the evaluation of \(A \Rightarrow B\) as the set of statements made by \(A\) being a subset of the set of statements made by \(B\) also give the other axioms having to do with the \(\Rightarrow\) relation, namely

\[
(A \Rightarrow B) \equiv (A = A \land B) \\
(A \Rightarrow B) \supset ((A \text{ says } s) \supset (B \text{ says } s)) \\
(A = B) \equiv ((A \Rightarrow B) \land (B \Rightarrow A))
\]

The first of these follows from the interpretation of \(\Rightarrow\) as "is a subset of" and \(\land\) as set intersection, while the second follows directly from the interpretation of \(\Rightarrow\) as "is a subset of". The third follows from the fact that if two sets are subsets of each other, they are identical.

So far, so good. But now we run into some trouble. Now we need to deal with the axioms having to do with the "|" operator. The main problem we have with our interpretation of this operator is dealing with the claim that the operator is associative, that is, that \(A|(B|C) = (A|B)|C\). To see the problem with this, let's look first at the interpretation of the principal \(A \mid (B|C)\); according to our current set of rules for the model this will be defined as (eliding the clause that closes the set under implication):

\[
\{x \mid (\exists s)(s \in V(A) \text{ and } s \text{ has the form } "B|C \text{ says } x") \}
\]

While the principal \((A|B) \mid C\) will be interpreted as (with the same eliding)

\[
\{x \mid (\exists s)(s \in V(A|B) \text{ as } s \text{ has the form } "C \text{ says } x")\}
\]

which itself is identical to (expanding the \(V(A|B)\)) to

\[
\{x \mid (\exists s)(s \in \{ y \mid (\exists s')(s' \in V(A) \text{ and } s' \text{ has the form } "B \text{ says } y")\} \text{ and } s \text{ has the form } "C \text{ says } x")\}
\]

which simplifies to the set

\[
\{x \mid (\exists x)(s \in V(A) \text{ and } s \text{ has the form } "B \text{ says C says } x")\}
\]

This gets us transitivity only if we can be sure that the model will guarantee that if \(s\) has the form "\(B|C \text{ says } x\)" and \(s'\) has the form "\(B \text{ says } C \text{ says } x\)" then \(s \in V(A)\) if and only if \(s' \in V(A)\). We can insure this by augmenting our definition of the set \(S\), our range for the interpretation function applied to principals, to
$\Sigma$, the range of interpretation of the principals of $L$, is

\{
  x \mid x \in P(S) \text{ and }
  
  \text{if } s \text{ has the form } "B \text{ says } C \text{ says } q" \text{ and } s' \text{ has the form } "B\text{|}C \text{ says } q"
  
  \text{for any } B,C \in P \text{ and } q \in S, \text{ then } s \in x \text{ iff } s' \in x \text{ and }

  \text{if } \vdash s \text{ then } s \in x \text{ and }

  \text{if } s \in x \text{ and } s \supset s_2 \in x \text{ then } s_2 \in x
\}\}

An alternate, and far more natural, way of getting the property that the "|" operation is associative, would be to treat the quotation operation in the same way the New Yorker or the New York Times treats quoting; that is, if "A|B says s" is true then B says s. We could get this with the following valuation function (again, eliding the clauses that close the set under implication):

\[
\text{if } A \in P \text{ and } A \text{ has the form } B \mid C, \text{ then } V(A) = \\
\{s \mid s \in V(C) \text{ and } (\exists s')(s' \in V(B) \text{ and } s' \text{ has the form } "C \text{ says } s")\};
\]

in other words, that the set of sentences that form the interpretation of the principal "B|C" is the set of sentences that C actually says and that B says that C says. This interpretation is pretty clearly transitive, and also has a notion of quoting that corresponds to at least one ordinary use of the term.

Unfortunately, this is an interpretation that is explicitly rejected by the authors of the paper. On page 8, the paper state "A|B says something if A quotes B as saying it. This does not mean B actually said it: A could be mistaken or lying." Such could not be the case in the above interpretation, so this interpretation must be rejected. This leaves us with our first, less natural, interpretation.

The final axiom that we need to make true in all models is P10, also known as the handoff axiom, which is really the most important of the axioms for the actual authentication system. This axiom makes a principal authoritative with respect to another principal speaking for it, that is,

\[(A \text{ says } (B \Rightarrow A)) \supset (B \Rightarrow A)\]

On the model we have so far, this reduces to a requirement that on all models,

\[
\text{if } "B \Rightarrow A" \in V(A) \text{ then } V(A) \supset V(B)
\]

This is clearly not true on all of the models as they are specified so far. So far, we have left the interpretation of what a principal says pretty unconstrained. We have said that what they say has to be closed under implication, and we have some conditions that say that any principal saying things about other principals has consequences (so that we can get the quoting relationship right). But these conditions have been conditions on the sets that can be the interpretation of any principal. What we need for the handoff axiom to work is a condition that applies only to particular principals.

To do this, we need to introduce a condition on the evaluation function rather than a condition on the range of that function. The reason for this is that we can't specify the condition without reference to the particular evaluation function for a principal. It is fine, for example, for any principal other than A to make statements like "B \Rightarrow A" without V(A) \supset V(B). If principal C were to do this, it would be fine; it might not be true, but it would be fine (C can say lots of things that are wrong). It is only when A makes this statement, i.e., when "B \Rightarrow A" \in V(A), that we have put some requirement on the set that is V(A).

The requirement on the valuation function is just what one would expect, namely, if $V$ is a valuation function for the model then

\[
\text{for all } A, B \in P \text{ if } "B \Rightarrow A" \in V(A) \text{ then } V(A) \supset V(B)
\]
This insures that the handoff axiom is true in all models. It also insures that the other theorems having to do with handoff and credentials are true on all models. In particular, we can look at (P11):

\[(A' \Rightarrow A) \land A' \text{ says } (B \Rightarrow A) \supset (B \Rightarrow A)\]

This is interpreted as saying that if \(V(A) \supset V(A')\) and "B \Rightarrow A" \(\in V(A')\) then "B \Rightarrow A" \(\in V(A)\) and therefore we can conclude \(V(A) \supset V(B)\). Using similar reasoning, we can show that (P12),

\[(A' \land B \Rightarrow A) \land (B \Rightarrow A') \supset (B \Rightarrow A)\]

is true on all models.

The final model, then, augmented with the conditions that will make the axioms dealing with quoting and speaks-for true on all possible interpretations, looks like this:

A model, \(M\), of \(L_A\) is an ordered triple < \(T, \Sigma, V\) > where

\(T\), the range of interpretation of the statements of \(L_A\), is the set \(\{T, F\}\),
\(\Sigma\), the range of interpretation of the principals of \(L\), is
\[
\{x \mid x \in \mathcal{P}(S) \& \text{ if } s \text{ has the form "B says C says q" and } s' \text{ has the form "B}C \text{ says q" for any } B, C \in P \text{ and } q \in S, \text{ then } s \in x \text{ iff } s' \in x \& \text{ if } s \in x \text{ and } s \supset s_2 \in x \text{ then } s_2 \in x\}
\]
\(V\) is a valuation function with domain \(S \cup P\) and range \(T \cup \Sigma\) such that

- if \(A\) is a simple principal, then \(V(A) \in \Sigma\);
- for all \(A, B \in P\) if "B \Rightarrow A'" \(\in V(A)\) then \(V(A) \supset V(B)\);
- if \(A \in P\) and \(A\) has the form \(B \land C\), then \(V(A) = V(B) \cap V(C)\);
- if \(A \in P\) and \(A\) has the form \(B \mid C\), then \(V(A) = \{x \mid (\exists s)(s \in V(B) \text{ and } s \text{ has the form "C says x") or } (\exists s)(s \in V(A) \text{ and } s \text{ implies } x)\}\);
- if \(s\) is an atomic statement of \(S\), then \(V(s) = \{T, F\}\);
  - if \(s\) is of the form \(s_1 \land s_2\), then \(V(s) = T\) if and only if \(V(s_1) = T\) and \(V(s_2) = T\), and \(F\) otherwise;
  - if \(s\) is of the form \(s_1 \supset s_2\), then \(V(s) = F\) if and only if \(V(s_1) = T\) and \(V(s_2) = F\), and is \(T\) otherwise;
  - if \(s\) is of the form \(s_1 \equiv s_2\), then \(V(s) = (V(s_1) = V(s_2))\);
  - if \(s\) is of the form \(A \text{ says } s\), then \(V(s) = T\) iff \(s \in V(A)\);

Having this specification does not prove that there actually exists a model for the logic of authentication. All we have done is to say what such a model would be. To actually prove that such a model exists we would need to prove that there actually is something that has the characteristics specified above. The existence of the sets \(T\) and \(\Sigma\) are pretty easy to prove; what is hard is to show that there is a function with the characteristics of \(V\). It isn't impossible, but we haven't done it. However, it doesn't look like there are any real problems here, either. So we could say with a fair amount of confidence that the logic has a model, and therefore that the logic (or at least this subset of the logic) is consistent.

However, there are also some properties of the model that we should recognize. To begin with, the logic generates an infinite set of sentences (one of the characteristics of a recursive set of generation rules) and the interpretation starts with the range of the principals being the power set of that set. So we are talking big cardinalities here. The model may be useful for our thinking about the problem, but it isn't something that we are going to put into a computer (this is one of the reason for the popularity of axiomatic systems in computer
science). On the other hand, thinking of principals as the set of sentences that we can think of as being said by those principals allows us to tie our intuitions to a real formal system in which the ideas that we had to justify the axioms of the system can be clearly stated and tested.

**Extending the Model**

After establishing the basic model, the Lampson paper tries to extend the model to cover other aspects of authentication that, it is claimed, can be dealt with using minor extensions of this standard logic. The exposition on these extensions is somewhat difficult, in that the discussion often moves from extending the logic to the sorts of certificates that would be needed to support such an extension in the implementation of the logic, or from the kinds of deductions that can be made in the logic to the kinds of proofs that can be constructed in the implementation. The result is a more hurried and less careful discussion, which makes it more difficult to be sure of the interpretation. Never the less, we will try to extend our model in the same way, using the basic model with the extensions to deal not just with the authentication of basic principals but also to deal with the authentication of groups, use of roles, delegation of authentication, and cases in which hierarchical certificate authorities are used in the authentication.

**Groups**

The simplest of these extensions has to do with groups. A group, according to the paper, is a principal that has "no public key or other channel of its own." The paper then goes on to characterize a group as a set of credentials of the form "A ⇒ G" for all principals A that are members of the group G. If we were to try to capture this in our model, we would have a set of simple principals, G, G_1, G_2, ... that are the set of groups, and then extend the definition of our valuation function, V, so that

if G is a group, then \( V(G) \subseteq \{ x \mid x \text{ is a statement of the form } "A \Rightarrow G" \text{ for } A \in P, \) 
or \( x \text{ is a statement entailed by some other member of } V(G) \} \)

that is, the interpretation of a group is that a group is a subset of the set of all statements of the form "A speaks for G" for some principal A. Intuitively, the membership of the group is the set of principals that the group says speaks for the group. If we don't want to allow groups to be members of groups, then we can restrict the set of principals that are in the statements that are the interpretation of a group to those principals that are not groups. The more general formulation, which is what we have above, allows any principal, including a group, to be a member of a group.

The second clause of the evaluation statement for a group is somewhat more inclusive than it would appear at first glance. If "A ⇒ G" is part of the interpretation of the group G, then it follows that "G says A ⇒ G", since the interpretation of says is the set of sentences in the interpretation of a principal. But if that is true, then "A ⇒ G" is true. So from this it follows that everything in the interpretation of A will also be in the interpretation of the group G.

This interpretation does have the seemingly counter-intuitive result that the set of things that is said (i.e., allowed to be done) by any principal A is said by any group G in which A is a member. This follows from the fact that if A ⇒ G, then V(G) ⊇ V(A) (which is itself entailed by theorem (P8)). This has the odd effect of meaning that if some principal says s, then all of the groups of which that principal is a member also says s. While this may seem counter-intuitive, it is a result of the interpretation that allows any member of a group to have all of the access that is granted to the group; given a language that only has notions of saying something and speaking for another, this is the only way to express this relationship.

**Roles**

The second extension to the basic model is that of a role. The intuitive notion of a role is that it is a set of rules that a principal acting in the role will follow. The way this is introduced in the paper is to introduce a new relation, as, with the intuitive interpretation that "A as R" is the principal A acting in role R. This in turn is interpreted in terms of the says operation, so that "A as R says s" is taken to be equivalent to "A says R
says s." By our earlier results, we know that "A says R says s" is equivalent to "A|R says s", so "A as says R says s" can be reduced to "A|R says s."

This is merely a sketch of the notion of a role in the formal system, but we can deduce a number of the properties of a role from the sketch. Since "A|R" is well-formed (with A a principal and R a role) then roles must be something like a principal, since only principals can be quoted. However, roles are not simply principals; it does not appear that roles can say anything themselves (that is, it is not clear that "R says s" can be true outside the context of being quoted). However, it is possible for a non-role principal to quote a role saying something, so “R says s” must be syntactically well-formed. So roles seem to be their a special category of principal; adding roles to the logic can be done by adding to the syntax of $L_A$:

There is a set of roles $R$, $R_1$, $R_2$,... $\in$ Roles;
if $R \in$ Roles and $A \in P$, then "A as R" $\in$ P and "A|R" $\in$ P;
if $R \in$ Roles and $s \in S$, then “R says s” $\in$ S;

We need to add to our interpretation function a clause for interpreting basic roles, so that we can also interpret sentences of the form “R says s.” We want these sentences to always be false, since a role can’t say anything itself. To get this result, we add to our interpretation function

if $R \in$ Roles, then $V(R) = \{\emptyset\}$

No additional interpretation rule is needed for the principal "A|R"; this will just be the set of sentences, $s$, that are such that "R says s" is something that is in the interpretation of A. Since "A as R" is equal to "A|R", it will have the same interpretation. This also gives us the result that theorem (R1),

$$\vdash A as R = A | R$$

is true on all models.

Theorem R2 is somewhat more difficult. That theorem states

$$\vdash A \Rightarrow A as R \text{ for all } R \in \text{Roles}$$

The main problem with this theorem is that it is ambiguous. It could mean that, for each role in the set of Roles, A speaks for A-in-that-role. This would mean that the interpretation of any role that A adopts must be interpreted as a superset of the set of statements made by A. This seems an odd interpretation, since it would mean that any statement made by A (outside of any role) can also be made by A in any role. Given that the original idea behind roles was to allow A to limit the rights that it had by taking a role, this interpretation seems wrong. But there is also the remark in the paper that "A must be careful that what it says on its own is appropriate for any role it may adopt" (p. 27) that seems to support this interpretation. But on this interpretation the idea of a role is very different from the idea in our intuitions that this was meant to reflect.

A second interpretation of R2 corresponds much more closely to our intuitive notion of a role. This interpretation would understand R2 as saying that A speaks for the union of all of the things A could say in every role that A can have. On this interpretation, we would understand principals as always being in some role when they make a statement (thus getting the result that what the principal says is the union of what they says in each role they can adopt). This also fits the intuition that adopting a role can limit the rights of a principal. The role I take on when running an untrusted program can be quite limited while the role of superuser is unlimited; all that is required on this interpretation of R2 is that the set of statements that I says is part of the union of those two sets.

Which of the two interpretations of (R2) we use also determines the way in which we interpret roles in the formal semantics. If we use the interpretation on which a principal speaks for itself in any role, then we would extend our evaluation function to
If \( P \) is of the form "\( A \text{ as } R \)" for some \( R \in \text{Roles} \), then \( V(P) \subseteq \Sigma \) and \( V(P) \supseteq V(\Lambda) \)

If, on the other hand, we interpret (R2) in the second way, then we would extend the evaluation function in the model to

\[
\text{if } P \text{ is of the form } "A \text{ as } R" \text{ for some } R \in \text{Roles}, \text{ then } V(P) \subseteq \Sigma \text{ and } V(P) \supseteq \{V("A \text{ as } R" \text{ for all } R \in \text{Roles})\} \setminus V(A)
\]

In other words, the interpretation of "\( A \text{ as } R \)" is a subset of the set of statements that are part of the interpretation of \( A \), and the union of the interpretation of "\( A \text{ as } R \)" for all possible roles \( R \) is exactly the set of things that \( A \) says.

Unfortunately, in this more intuitive interpretation of the notion of a role we cannot get the truth of \( A \Rightarrow A \text{ as } R \) for any particular role \( R \). Indeed, what we get is the reverse of this, that for any particular role \( R \) it will always be true that \( A \text{ as } R \Rightarrow A \). This corresponds to the intuition we started with that \( A \) could limit its access by adopting a role, since the set of statements made in a role will always be subset of the set of statements made by the entity adopting the role. However, none of the discussions in the paper having to do with using roles requires as a premise \( A \Rightarrow A \text{ as } R \) for a particular role, so there seems to be nothing to prevent this interpretation. We will use this second interpretation in the rest of this paper.

**Delegation**

The mechanism used for delegation is less ambiguous than the mechanism used for roles. We begin by introducing a new operation on principals, the \( \text{for} \) operation. The intuitive idea is that a delegation from principal \( A \) to principal \( B \) will be represented as \( B \text{ for } A \), which is seen as principal \( B \) acting on behalf of principal \( A \). While never stated explicitly in the paper, the use of the \( \text{for} \) operation makes it clear that the result of operating on a pair of principals with that operation is itself a principal. So we would add to the syntax of the language \( L_\Lambda \) the rule

\[
\text{if } A, B \in P, \text{ then } "A \text{ for } B" \in P
\]

The two axioms that are given for the operation are

\[\vdash A \land B \mid A \Rightarrow B \text{ for } A\]

\[\vdash \text{for is monotonic and distributes over } \land\]

The idea behind the first of these axioms is that delegation should be thought of as both \( A \) says some statement and \( B \) quoting \( A \) says that statement; that is, both the principal doing the delegating and a quoting of that principal by the principal to whom the delegation was given need to be involved.

This can be captured fairly easily in the model. The notion is that the model will interpret the principal formed by "\( B \text{ for } A \)" as the intersection of the interpretation of \( A \) and the principal formed by "\( B \mid A \)". To do this we add the clause to our definition of the valuation function, \( V \),

\[
\text{if } C \in P \text{ is of the form } "B \text{ for } A" \text{, then } V(C) = V(\Lambda) \cap V(B \mid A)
\]

This gives us just what we want intuitively, namely, that the set of statements that are in the interpretation of the delegation from \( A \) to \( B \) are just those statements that \( A \) would make that are also statements that \( B \) quotes \( A \) as saying. Unfortunately, it is also a stronger interpretation than is justified by the axioms.

To see this, notice that the first of the axioms for \( \text{for} \) simply states that \( A \land A \mid B \Rightarrow A \text{ for } B \), which means that any statement made by \( A \land A \mid B \) is also made by \( A \text{ for } B \). But this doesn't entail that if a statement is made by \( A \text{ for } B \) that it is also made by \( A \); the \( \Rightarrow \) relation only goes one way. So from the axioms it would be fine for \( B \text{ for } A \)
to say anything it likes, even those things that are not something that A would say. This seems at best odd and at worst wrong; it certainly conflicts with our intuition about delegation.

Repairing this is, fortunately, pretty straightforward. What we really want is already in our interpretation, we just need to change the theorems of $L_A$ to reflect the stronger guarantees of the interpretation. This can be done by changing the first theorem for the as operation to

\[ \vdash A \land B | A = B \text{ for } A \]

The interpretation rule proposed above still works for this version of the theorem, and it also allows moving from statements of the form "$B \text{ for } A$ says $s$" to "$A$ says $s$", which we would seem to want.

**Hierarchical Names**

The final extension to $L_A$ that we need to incorporate into our model is that introduced to account for hierarchical names and multiple certificate authorities. This is treated rather briefly in the paper, and most of that treatment is in the form of an extended example that is based on the practical implementation of multiple certificate authorities. The additions to the formal logic are given quite briefly and in an incomplete fashion, so much of what follows is a reconstruction of what should have been said in the paper.

The intuition behind the section is that certificate authorities are often organized in a hierarchical fashion, and the chains of trust that are formed in such an environment should be as small as possible. The idea is that rather than using simple names, we can use hierarchical names to trace the chains of trust to the least tree spanning the principal making a request and the entity granting the request, and then form a proof that the authenticated principal is who he claims to be using certificates from the smallest possible set of certificate authorities. Further, we want to insure that the transfer of trust is always local; no one needs to certify the identity of anyone who is not closely related to him.

While all of these intuitions seem fine, there is a lot of mechanism that needs to be introduced to reflect the intuitions. We do not currently have the notion of a hierarchical name, nor do we have a notion of a hierarchical organization of any particular set of entities, nor do we have a notion of a certificate authority. Rather than introduce all of these extra entities, the paper simply adds an operation, `except`, which works on hierarchical names and allows navigation on the hierarchy of things that are named. We will try to be a bit more precise.

We will first specify the syntax for hierarchical names. We could do this in a way that does not distinguish between certificate authorities and other principals; after all, certificate authorities are just entities that make statements and so would seem to be a form of principal. However, certificate authorities only make statements of a particular kind (they are always statements of the form "$A \Rightarrow B"), and the hierarchy that we are going to build requires certificate authorities in certain parts of the hierarchy and other kinds of principals in other parts of the hierarchy. So it will be easier, in both the syntactic and semantic parts of the system, to distinguish certificate authorities from other kinds of principals, in much the same way that we distinguished roles and groups from other kinds of principals.

So we begin by defining the set, $CA$, of certificate authorities as follows:

there is a set of simple names, $c, c_1, c_2, \ldots \in CA$;
if $c, c_1 \in CA$ then "$c/c_1$" $\in CA$;

We then add to the overall syntax of $L_A$ the rules

if $A \in P$ and $c \in CA$, then $c/A \in P$
if $c \in CA$ and $A \in P$, then $c \Rightarrow A \in S$
if $c, c_1 \in CA$ then $c \Rightarrow c_1 \in S$
if $c_1, c_2 \in CA$ and $c$ is of the form "$c_1/c_2$", then "$c \text{ except } c_2" \in CA$;
if \(c_1 \in \text{CA}\) and \(A \in P\) and \(c\) is of the form "\(c_1/A\)"; then "\(c\ \text{except}\ A\)" \(\in P\);
if \(c_1 \in \text{CA}\) and \(x \in P \cup \text{CA}\), and \(c\) is of the form "\(c_1/x\)"; then "\(c\ \text{except}.." \(\in \text{CA}\)
if \(c_1 \in \text{CA}\) and \(s \in S\), then "\(c_1 \text{ says}\ s\)" \(\in S\)

These syntax rules are actually slightly more restrictive than required by the logic of authentication presented in the paper. In the paper, certificate authorities are treated simply as another kind of principal, and so any principal can occur in any position in a hierarchical name. However, all of the examples treat certificate authorities as the only entities that can occur as other than leaf nodes in the hierarchy, and making a distinction between the leaf nodes (which are principals) and the non-leaf nodes (which are certificate authorities) will simplify the overall interpretation without changing the use in practice of the logic. Also note that, since we are differentiating between certificate authorities and general principals, we need the rules that allow the \(\Rightarrow\) relationship to be defined between certificate authorities and principals and between certificate authorities and other certificate authorities. Similarly, we need to define the syntax for a certificate authority saying some statement. Since certificate authorities can say statements, they can also quote others in saying those statements. However, since quoting is not an important part of the role of certificate authorities, we will not introduce the quoting relationship in our syntax.

The syntax we have introduced is also far more restrictive in the use of the \(\text{except}\) operation. In the paper, "\(P\ \text{except}\ N\) is a principal that speaks for any path name that is an extension of \(P\) as long as the first name after \(P\) isn\'t \(N\), and for any prefix of \(P\) as long as \(N\) isn\'t '..". Our rules are far more restrictive; "\(c\ \text{except}\ c_2\)" is only well-formed when \(c\) has the form "\(c_1/c_2\)" and will be itself the name of a certificate authority, while "\(c\ \text{except}\ A\)" is only well formed if \(c\) has the form "\(c_1/A\)" and will be the name of a principal. Further, "\(c\ \text{except}.."\) is only well formed if \(c\) is itself a hierarchical name, and will be the name of a certificate authority. While these syntax rules are more restrictive than the ones informally described by Lampson, they cover all of the examples that he gives in the paper, and will make the effort of giving a model far simpler.

We now turn to our model for the enhanced logic. Unlike the previous extensions, where we only needed to extend our interpretation function \(V\), we need to add two new elements to the overall model. The first of these elements is a set of certificate authorities, which we will treat as a separate kind of entity. The second is a hierarchical structure on the set of principals and certificate authorities, and we can only interpret our syntactic elements with respect to this hierarchy. The way to introduce such a hierarchy is to add an ordering relation, >, over the union of the set of principals and certificate authorities, to our interpretation. This will be a partial order, in that there will be some elements that are unordered by the relation. We make the order induce a tree structure by requiring that the order be transitive, irreflexive, and asymmetric, that is, for any \(p, q, \) and \(r\) in the range of the partial order

\[
(\forall p) ((p > q \land q > r) \supset p > r)
\]
\[
(\forall p) \neg (p > p)
\]
\[
(\forall p)(\forall r)((p > r) \supset \neg (r > p))
\]

If we further specify that there is a single \(p\) such that

for all \(r \neq p\), \(p > r\)

we will have a single tree with root \(p\); without such a specification we could have a forest of disjoint trees. For ease of exposition, we also define two relationships, \(\text{parent}\) and \(\text{child}\), in the obvious ways:

\[
P \text{ parent } Q \equiv (P > Q \land \neg (\exists x)(P > x > Q))
\]
\[
P \text{ child } Q \equiv (Q > P \land \neg (\exists x)(Q > x > P))
\]

Clearly, a principal or certificate authority can have only a single parent, but could have many children (and, additionally, the set of children of a single parent are unordered by \(\rangle\)).
We now define a model of the logic, $L_A$, as an ordered five-tuple $< T, \Sigma, \Gamma, \mathcal{V}, \succ >$ exactly like before, with the addition that

$\Gamma$, the set of certificate authorities,
$\succ$, a partial order over $\Sigma \cup \Gamma$, the union of the set of certificate authorities and the set of principals, which is transitive, irreflexive, asymmetric, and such that if $x \in \Sigma$, then there is no $y$ such that $(x \succ y)$.

The interpretation function, $\mathcal{V}$, is extended as follows:

- if $c \in CA$, then $\mathcal{V}(c) \in \Gamma$;
- if $c$ is a complex name of the form "$c_1/c_2$", then
  - $\mathcal{V}(c) = \emptyset$ if not (($\mathcal{V}(c_1)$ parent $\mathcal{V}(c_2)$) or $\mathcal{V}(c) = \emptyset$);
  - $\mathcal{V}(c) = \emptyset$ otherwise;
- if $c$ is a complex name of the form "$c_1/A$" where $c_1\in CA$ and $A\in P$, then
  - $\mathcal{V}(c) = \emptyset$ if $\mathcal{V}(c_1) = \emptyset$ or if not ($\mathcal{V}(c_1)$ parent $\mathcal{V}(A)$); and
  - $\mathcal{V}(A)$ otherwise;
- if $c_1 \in CA$ and $A \in P$ and $s$ is of the form "$c_1 \Rightarrow A$", then $\mathcal{V}(s) = T$ if
  - $\mathcal{V}(c_1) \neq \emptyset$ and $\mathcal{V}(c_1)$ parent $\mathcal{V}(A)$;
  - $\mathcal{V}(A) = F$ otherwise;
- if $c_1, c_2 \in CA$ and $c$ has the form "$c_1 \Rightarrow c_2$", then $\mathcal{V}(c) = T$ if
  - $\mathcal{V}(c_1) \neq \emptyset$, $\mathcal{V}(c_2) \neq \emptyset$, and $\mathcal{V}(c_1)$ parent $\mathcal{V}(c_2)$;
  - $\mathcal{V}(c) = F$ otherwise;
- if $c_1, c_2 \in CA$ and $c$ has the form "$c_1$ except $c_2$", then $\mathcal{V}(c) = \emptyset$ if $\mathcal{V}(c_1) = \emptyset$ or $\neg (\mathcal{V}(c_1)$ parent $\mathcal{V}(c_2))$, otherwise
  - $\mathcal{V}(c) \in \mathcal{V}(c_1) \cup \{ x | (\mathcal{V}(c_1) \text{ parent } x) \land (x \neq \mathcal{V}(c_2))\}$;
- if $c_1 \in CA$ and $A \in P$ and $c$ has the form "$c_1 \text{ except } A$", then $\mathcal{V}(c) = \emptyset$ if $\mathcal{V}(c_1) = \emptyset$ or $\neg (\mathcal{V}(c_1)$ parent $\mathcal{V}(A))$, otherwise
  - $\mathcal{V}(c) \in \mathcal{V}(c_1) \cup \{ x | (\mathcal{V}(c_1) \text{ parent } x) \land x \neq \mathcal{V}(A)\}$;
- if $c_1 \in CA$ and $c$ is of the form "$c_1$ except .." then
  - $\mathcal{V}(c) = \emptyset$ if $\mathcal{V}(c_1) = \emptyset$; otherwise
  - $\mathcal{V}(c) \in \{ y | \mathcal{V}(c_1) \text{ parent } y\}$;
- if $c_1 \in CA$ and $s_1 \in S$ and $c$ has the form "$c_1$ says $s_1$", then $\mathcal{V}(c) = T$ iff
  - $\mathcal{V}(c_1) \neq \emptyset$ and ($\exists x)((\mathcal{V}(c_1) \Rightarrow x) = T) \land (\mathcal{V}(x \text{ says } s_1) = T)$,
  - $\mathcal{V}(c) = F$ otherwise.

A few notes about this interpretation of certificate authorities. We interpret certificate authorities by introducing a new set of entities, $G$. We don't say anything about this set other than to say that our interpretation function will map the simple names of certificate authorities to this set. We don't need to say anything more about the contents of the set because the meaning of certificate authorities is actually given by the partial ordering relationship that is defined over the union of the certificate authorities and the set of principals.

The partial ordering, $\succ$, gives us a tree structure made up of principals (which are required to be leaf nodes of the tree) and certificate authorities. We have not restricted the interpretation to have a single overall root; if we want such a root we can introduce it by simply specifying that there is some certificate authority that is $\succ$ all other certificate authorities and principals.

All of the clauses interpreting (possibly) compound names begin with a specification of when those names are interpreted as the null set. This is because our syntax allows us to build compound names that are not reflected in the hierarchy induced by the ordering relation. For example, if we have a segment of the tree in which the certificate authority named $C$ has as children certificate authorities named $A$ and $B$, there is nothing in our
syntax rules that preclude us forming the complex name A/B. Indeed, we wouldn't want such a rule, since there will be other models (with other mappings from names to certificate authorities and other partial orders) in which A is the parent of B. Whether or not a compound name actually picks something sensible out in the model depends on the interpretation function and the ordering relation. So we need to have some value that indicates that the name doesn't denote anything; in our model we make such compound names denote the empty set.

For those names that do reflect the structure of the tree, the thing named is simply whatever is the last element of the name. Which is what we would expect. This could be either a principal or a certificate authority, which is why there needs to be two different rules in both the syntax and the semantics for the system, as the types of the two expressions are different.

The interpretation that has been given for the "⇒" operation is that a certificate authority speaks for all of the entities for which it is the parent. We join this with our interpretation of the "says" operation, interpreting "says" for certificate authorities as the union of all of the statements that are said by any entity that the certificate authority speaks for; this gives the result that a certificate authority says all of the statements that the children of that certificate authority says. This gives the immediate result that if a certificate authority speaks for some other entity (either principal or certificate authority) then anything that is said by the entity spoken for is also said by the certificate authority. What does not follow is theorem (8) of the paper, which states

\[ \vdash (A \Rightarrow B) \supset ((A \text{ says } s) \supset (B \text{ says } s))\]

that is, that if A speaks for B, then everything A says B also says. For this to hold with certificate authorities and the hierarchy that has been constructed, a certificate authority could only say those things that were said by all of the entities for which it was the parent. This would not allow the certificate authority to issue statements that were informative to any of the entities for which it was a certificate authority, since anything said by the certificate authority would have to have been said by the principal the certificate authority dominates in the hierarchy.

In fact, this is why we can't interpret certificate authorities as simply a different kind of principal, as outlined in the paper. To do this would require that we not be able to use certificate authorities as they are used in the example in the paper. It seemed better to interpret these entities as something different than principals that would give the results needed by the paper than in a way that was consistent with the (few) things said about their place in the formal system.

This leaves only the three clauses for the except relation. The first of these interprets the except relationship between two certificate authorities; if c₁ and c₂ are interpreted as certificate authorities then "c₁ except c₂" is interpreted as either the certificate authority named by c₁ or one of the peers (in the hierarchy) of the certificate authority named by c₂. Similarly, if c₁ is the name of a certificate authority and A the name of a principal, "c₁ except A" is interpreted as either the certificate authority named by c₁ or one of the peers (in the hierarchy) of the principal named by A. The final clause will take a hierarchical name, c, and interpret "c except .." as anything at the level of the leaf of the name, but not higher. Interpreting the except operation this way allows moving in just the ways that the examples in the paper requires; from "a except b" (if b is not ").") you can move up or sideways in the hierarchy, and from "a except .." you can move down the hierarchy.

One unusual aspect of the interpretation of the except operation is that our interpretation function does not give a single value for the interpretation, but rather specifies a set in which the interpretation must be a member. This makes the interpretation of "c except d" ambiguous. This is somewhat at odds with the description in the paper, which takes "c except d" to be a single principal that speaks for a set of other principals. Our interpretation has "c except d" to be ambiguous, picking out any of the set of individuals (either certificate authorities or principals) that the unambiguous interpretation would speak for. It should be noted that all of the examples that are given in the paper can be accounted for on the ambiguous interpretation.
The interpretation that we have given to certificate authorities and compound names differs from what is said by Lampson in a number of ways. The most obvious is that we have introduced a new category of entity, the certificate authority, rather than just making certificate authorities another sort of principal. As noted before, we did this to avoid the absurdity of limiting what could be said by certificate authorities to the intersection of all of the entities for which they speak. In fact, Lampson is mute on the subject of certificate authorities speaking for any other principal; instead he gives rules concerning the "speaks for" relation only for the principals formed by using compound names and the `except` operation. In effect, the principal formed by the `except` operation speaks for all of the entities that are less trustworthy than the entity excluded by the `except` clause. For example, `c except A` speaks for all of the entities other than `A` that are peers or is a parent of `A`; this is just the set of entities that `A` needs to trust for hierarchical authentication. This is the intuition that the semantics has tried to capture.

In doing so, however, we have made true in all models a number of relations that Lampson has explicitly rejected. Since the parent of a node in the hierarchy speaks for all of its children, and since speaks for is transitive, higher levels of the hierarchy speak for all of the levels below them. In effect, if some entity `A` is in the hierarchy, `A ⇒ B` is true for all `B` such that `A > B`. This is explicitly rejected by Lampson, who states that we should not allow more distant certificate authorities to be able to authenticate more local authorities. However, this seems counter to our notions of the transfer of trust; if certificate authority `A` is our reason for trusting certificate authority `B`, then anything we trust because of `B` also seems to be trusted because of our trust in `A`.

As with our model for the basic logic, the model that we have given here is only a start if we were going to use the model to prove properties of the logic of authentication. We have not shown that there actually is a model that meets the requirements we have given, so we have not shown the consistency of the logic. At best we have shown that if the model we have specified exists, then the part of the logic of authentication that has been correctly captured by this model is consistent.

However, even this partial exercise is not without value. In attempting to build a model for the basic system, we have shown how to map the axiomatic system presented by Lampson into a set-theoretic framework. That set-theoretic framework may be more intuitive or easier to reason about than the axiomatic system, and given the mapping between the logic of authentication and the model we have provided we can be sure that the model will maintain any property of the logic.

We have also seen that the model can be used to interpret some, but not all, of the extensions of the logic of authentication given in the paper. One place where the interpretation seems useful is in the notion of a role, where attempts to extend the model showed an ambiguity in the notion of roles presented in the paper, and allowed us to think about the two possible interpretations that the ambiguity allows. Further, the attempt to extend the model to the area of hierarchical names showed how under-specified that section of the logic is, and the difficulties of supplying an interpretation for the formalized extension show areas where the logic needs further development.

**Appendix 1**

Syntax of $L_A$:
The set of principals, $P$, is defined as

1) an infinite set of simple principals $A, B, C, A_1, A_2, ... ∈ P$;
2) an infinite set of Roles, $R, R_1, R_2, ... ∈ Roles$
3) an infinite set of groups, $G, G_1, G_2, ... ∈ P$
4) If $A, B ∈ P$, then $A \wedge B ∈ P$, $A \vee B ∈ P$
5) If $A ∈ P$ and $R ∈ Roles$ then “$A | R$” $∈ P$
The set of certificate authorities, $CA$, is defined as

1) an infinite set of simple names, $c, c_1, c_2, ... ∈ CA$;
2) if c, c₁ ∈ CA then "c/c₁"∈ CA;
The set of statements of Lₐ, S, having to do with principals and statements, are defined as
1) an infinite set of atomic statements s, s₁, s₂, ... ∈ S;
2) if s₁, s₂ ∈ S, then so are s₁∧s₂, s₁ ⊃ s₂, and s₁≡ s₂
3) if A ∈ P and s₁ ∈ S, then A says s₁ ∈ S
4) if A, B ∈ P then A ⊸ B ∈ S, A = B ∈ S

The statements and expressions having to do with certificate authorities and compound names are defined as
1) if A ∈ P and c ∈ CA, then c/A ∈ P
2) if c ∈ CA and A ∈ P, then c ⊸ A ∈ S
3) if c, c₁; ∈ CA then c ⇒ c₁ ∈ S
4) if c₁, c₂; ∈ CA and c is of the form "c₁/c₂", then "c except c₂" ∈ CA;
5) if c₁ ∈ CA and A ∈ P and c is of the form "c₁/A", then "c except A" ∈ P;
6) if c₁ ∈ CA and x ∈ P ⊃ CA, and c is of the form "c₁/x" then "c except .." ∈ CA
7) if c₁ ∈ CA and s ∈ S, then "c₁ says s" ∈ S

Appendix 2

Semantics of Lₐ:
A model of the logic, Lₐ, as an ordered four-tuple < T, Σ, Γ ,V, > such that
T, the range of interpretation of the statements of Lₐ , is the set {T, F},
Σ , is the set of certificate authorities,
Γ , is a partial order over Σ∪Γ , the union of the set of certificate authorities and the set of principals,
such that if x ∈ Σ, then there is no y such that (x > y)
Σ , the range of interpretation of the principals of L , is
{x | x ∈ ϕ(S) and
if s has the form "B says C says q"
and s' has the form "B|C says q" for any B,C ∈ P and q ∈ S,
then s ∈ x iff s' ∈ x and
if ⊢ s then s ∈ x & if s ∈ x and s ⊃ s₂ ∈ x then s₂ ∈ x}
V is a valuation function with domain S ⊃ P and range T ⊃ Σ such that
if A is a simple principal, then V(A) ∈ Σ;
if G is a group, then V(G) ⊂ {x | x is a statement of the form "A ⊸ G" for A ∈ P,
or x is a statement entailed by some other member of V(G)};
if R is a Role, then V(R) = {Ø};
if c ∈ CA, then V(c) ∈ Γ;
for all A, B ∈ P if "B ⊸ A" ∈ V(A) then V(B) ⊃ V(A);
if A ∈ P and A has the form B ∧ C, then V(A) = V(B) ∩ V(C);
if A ∈ P and A has the form B | C, then V(A) =
{x | (∃ s)(s ∈ V(A) and s has the form "B|C says x") or
(∃ s)(s ∈ V(A) and s implies x)}
if s is an atomic statement of S, then V(s) ∈ {T,F};
if s is of the form s₁∧s₂, then V(s) = T if and only if V(s₁) = T and V(s₂) = T,
and F otherwise;
if s is of the form s₁ ⊃ s₂ , then V(s) = F if and only if V(s₁) = T and V(s₂) = F,
and is T otherwise;
if s if of the form s₁≡ s₂ , then V(s) = (V(s₁) = V(s₂));
if \( s \) is of the form \( A \textbf{ says } s \), then \( V(s) = \top \) iff \( s \in V(A) \);

if \( c \) is a complex name of the form "\( c_1/c_2 \)" , then
\[
V(c) = \emptyset \text{ if not } (V(c_1) \textbf{ parent } V(c_2)) \text{ or if } V(c_1) = \emptyset ;
\]
\[
V(c_2) \text{ otherwise};
\]

if \( c \) is a complex name of the form "\( c_1/A \)" where \( c_1 \in CA \text{ and } A \in P \), then
\[
V(c) = \emptyset \text{ if } V(c_1) = \emptyset \text{ or if not } (V(c_1) \textbf{ parent } V(A)); \text{ and}
\]
\[
V(A) \text{ otherwise};
\]

if \( c_1 \in CA \text{ and } A \in P \) and \( c \) is of the form "\( c_1 \Rightarrow A \)", then \( V(c) = \top \) if
\[
V(c_1) \neq \emptyset \text{ and } V(c_1) \textbf{ parent } V(A);
\]
\[
V(c) = \bot \text{ otherwise};
\]

if \( c_1, c_2 \in CA \) and \( c \) has the form "\( c_1 \Rightarrow c_2 \)" then \( V(c) = \top \) if
\[
V(c_1) \neq \emptyset, V(c_2) \neq \emptyset, \text{ and } V(c_1) \textbf{ parent } V(c_2);
\]
\[
V(c) = \bot \text{ otherwise};
\]

if \( c_1, c_2 \in CA \) and \( c \) has the form "\( c_1 \textbf{ except } c_2 \)" then \( V(c) = \)
\[
\emptyset \text{ if } V(c_1) = \emptyset \text{ or } V(c_2) = \emptyset \text{ or } \neg (V(c_1) \textbf{ parent } V(c_2)), \text{ otherwise}
\]
\[
V(c) \in V(c_1) \cup \{ x \mid (V(c_1) \textbf{ parent } x) \land (x \neq V(c_2)) \};
\]

if \( c_1 \in CA \) and \( A \in P \) and \( c \) has the form "\( c_1 \textbf{ except } A \)", then \( V(c) = \)
\[
\emptyset \text{ if } V(c_1) = \emptyset \text{ or } \neg (V(c_1) \textbf{ parent } V(A)), \text{ otherwise}
\]
\[
V(c) \in V(c_1) \cup \{ x \mid (V(c_1) \textbf{ parent } x) \land x \neq V(A) \};
\]

if \( c_1 \in CA \) and \( x \in P \cup CA \) and \( c \) is of the form "\( c_1/x \textbf{ except } .. \)" then \( V(c) = \)
\[
\emptyset \text{ if } V(c_1) = \emptyset \text{ or } V(x) = \emptyset \text{ or } \neg (V(c_1) \textbf{ parent } V(x)), \text{ otherwise}
\]
\[
V(c) \in \{ y \mid V(c_1) \textbf{ parent } y \};
\]

if \( c_1 \in CA \) and \( s_1 \in S \) and \( c \) has the form "\( c_1 \textbf{ says } s_1 \)" then \( V(c) = \top \) if
\[
V(c_1) \neq \emptyset \text{ and } (\exists x)(V(c_1 \Rightarrow x) = \top) \land (V(x \textbf{ says } s_1) = \top)).
\]

**Bibliography**
