

Compare Words
thesis-20000522.pdf

603 word(s) differ
353 word(s) added
502 word(s) deleted
Word(s) are rearranged

Compare Words
thesis-20000525.pdf



 for Andy



Abstract

I tackle the problem of naming and sharing resources across administrative boundaries. Conventional systems manifest the hierarchy of typical administrative structure in the structure of their own mechanism. While natural for communication that follows hierarchical patterns, such systems interfere with naming and sharing that cross administrative boundaries, and therefore cause headaches for both users and administrators. I propose to organize resource naming and security, not around administrative domains, but around the sharing patterns of users.

The dissertation is organized into four main parts. First, I discuss the challenges and tradeoffs involved in naming resources and consider a variety of existing approaches to naming.

Second, I consider the architectural requirements for user-centric sharing. I evaluate existing systems with respect to these requirements.

Third, to support the sharing architecture, I develop a formal logic of sharing that captures the notion of *restricted delegation*. Restricted delegation ensures that users can use the same mechanisms to share resources consistently, regardless of the origin of the resource, or with whom the user wishes to share the resource next. A formal semantics gives unambiguous meaning to the logic. I apply the formalism to the Simple Public Key Infrastructure and discuss how the formalism either supports or discourages potential extensions to such a system.

Finally, I use the formalism to drive a user-centric sharing implementation for distributed systems. I show how this implementation enables *end-to-end authorization*, a feature that makes heterogeneous distributed systems more secure and easier to audit. Conventionally, gateway services that bridge administrative domains, add abstraction, or translate protocols typically impede the flow of authorization information from client to server. In contrast, end-to-end authorization enables us to build gateway services that preserve authorization information, hence we reduce the size of the trusted computing base and enable more effective auditing. I demonstrate my implementation and show how it enables end-to-end authorization across various boundaries. I measure my implementation and argue that its performance tracks that of similar authorization mechanisms without end-to-end structure.

I conclude that my user-centric philosophy of naming and sharing benefits both users and administrators.



Contents

Abstract	v
I Introduction	1
1 Introduction	3
1.1 Motivation	3
1.2 Contributions	4
1.3 Overview	4
II Naming	7
2 Naming	9
2.1 Qualities of naming systems	9
2.1.1 User cost of establishing a name binding	9
2.1.2 Mnemonic value of a name binding	10
2.1.3 Semantic value of a name binding	10
2.1.4 Performance of name resolution	11
2.1.5 User cost of sharing a name binding	11
2.1.6 User cost of managing storage	11
2.2 Design and qualities of existing systems	11
2.2.1 Global namespaces	12
2.2.2 Plan 9	12
2.2.3 Sollins' negotiated names	13
2.2.4 Query-based naming	13
2.3 Design of Snowflake naming	14
2.4 Implementation of Snowflake naming	14
2.4.1 Infrastructure	14
2.4.2 Applications	17
2.4.3 System structure	17
2.5 Qualities of Snowflake naming	18
2.5.1 Cost of establishing a name binding	18
2.5.2 Mnemonic value of name bindings	19
2.5.3 Semantic value of name bindings	20

D	Review: The Simple Public Key Infrastructure	159
D.1	Certificate types	159
D.2	The SPKI 5-tuple	160
D.3	The SPKI 4-tuple	161
D.4	Tuple reduction	161

D	Review: The Simple Public Key Infrastructure	159
D.1	Certificate types	159
D.2	The SPKI 5-tuple	160
D.3	The SPKI 4-tuple	161
D.4	Tuple reduction	161

Volume Two

E	Snowflake software documentation	6
F	Experimental data	235

List of Figures

2.1	<i>The layers of software that constitute the Snowflake naming implementation.</i>	15
2.2	<i>Factoring naming services into a separate layer of system architecture . . .</i>	18
2.3	<i>The same namespace can capture names at multiple levels of abstraction. . .</i>	19
2.4	<i>An order-of-magnitude comparison showing the overhead associated with using RMI as a distribution substrate.</i>	22
5.1	<i>An example that shows when inherited names can become idempotent. . . .</i>	49
8.1	<i>A resource server contains an ACL that refers directly to Alice, a user in the same administrative domain.</i>	76
8.2	<i>With restricted delegation, Alice can introduce the remote principal Bob, and describe what authority he has over her resources.</i>	76
8.3	<i>When Alice connects to a resource server from the same machine, the server may trust the kernel to correctly identify her</i>	77
8.4	<i>Alice's request for a high-level resource involves not only her authority over the final low-level resource, but some interaction with the service providing a level of abstraction.</i>	78
8.5	<i>Gateways are necessary to translate between protocols, but they frequently impede the flow of authorization information.</i>	79
9.1	<i>A hypothetical SPKI sequence.</i>	83
9.2	<i>A structured proof that shows that a name defined by a client ($K_C \cdot N$) is bound to a particular document (H_D).</i>	84
9.3	<i>A look inside Alice's Prover.</i>	85
10.1	<i>Treating a channel as a principal</i>	88
10.2	<i>How my ssh RMI channel is integrated with Snowflake's authentication service.</i>	88
10.3	<i>The HTTP authorization protocol.</i>	92
10.4	<i>An HTTP authorization challenge message from a Snowflake server.</i>	93
10.5	<i>A response message from a Snowflake proxy.</i>	94
10.6	<i>The browser's interface to the Snowflake HTTP user agent</i>	95
11.1	<i>A transaction involving a quoting gateway.</i>	98
12.1	<i>The cost of introducing Snowflake authorization to RMI.</i>	102
12.2	<i>The cost of introducing Snowflake authorization to HTTP.</i>	105

Part I

Introduction

Chapter 4

The logic and semantics of restricted delegation

In this chapter, I introduce the formal logic and semantics of restricted delegation. I assume that the reader is familiar with the basic operation of modal logic, the Calculus for Access Control, and the semantics that support it. I provide a brief introduction to modal logic in Appendix B, and an overview of the Calculus for Access Control in Appendix C.

Formulas labeled with a P, S, L, or A prefix are from the Calculus for Access Control, and are defined in Appendix C. I label formulas developed in this chapter with an E prefix to indicate that they are part of my extended logic. I label formulas developed in Chapter 6 with a T prefix to indicate that they are part of the separate tag semantics.

The semantics given in [ABLP93] uses symbols that are conventional in the modal logic community but may surprise those from other backgrounds. The reader should be aware that I use the symbol $\supset$ for logical implication, and the symbol $\Rightarrow$ for the “speaks-for” operator on principals. Table 4.1 summarizes the notation I use for sets in the following sections.

4.1 Definition of restricted delegation

Lampson et al. mention in passing the idea of a restricted speaks-for operator [LABW92, p. 272]. In this section, I introduce my *speaks-for-regarding* operator, which formalizes the notion of the restricted speaks-for operator. It is written $B \stackrel{T}{\Rightarrow} A$, and read “ B speaks for A regarding the set of statements in T .” T is any subset of Σ^* . The desired meaning is that when $\sigma \in T$,

$$B \stackrel{T}{\Rightarrow} A \Leftrightarrow ((B \text{ says } \sigma) \Rightarrow (A \text{ says } \sigma))$$

The power of the speaks-for-regarding operator $\stackrel{T}{\Rightarrow}$ is that A can delegate a subset of its authority *without modifying any ACLs*. Contrast the situation with the use of roles in the Calculus for Access Control (see Appendix C.5), where to delegate authority over a restricted subset of her resources, a user had to define a role and install that role in the ACLs of each resource to be shared.

Chapter 4

The logic and semantics of restricted delegation

In this chapter, I introduce the formal logic and semantics of restricted delegation. I assume that the reader is familiar with the basic operation of modal logic, the Calculus for Access Control, and the semantics that support it. I provide a brief introduction to modal logic in Appendix B, and an overview of the Calculus for Access Control in Appendix C.

Formulas labeled with a P, S, L, or A prefix are from the Calculus for Access Control, and are defined in Appendix C. I label formulas developed in this chapter with an E prefix to indicate that they are part of my extended logic. I label formulas developed in Chapter 6 with a T prefix to indicate that they are part of the separate tag semantics.

The semantics given in [ABLP93] uses symbols that are conventional in the modal logic community but may surprise those from other backgrounds. The reader should be aware that I use the symbol $\supset$ for logical implication, and the symbol $\Rightarrow$ for the “speaks-for” operator on principals. Table 4.1 summarizes the notation I use for sets in the following sections.

4.1 Definition of restricted delegation

Lampson et al. mention in passing the idea of a restricted speaks-for operator [LABW92, p. 272]. In this section, I introduce my *speaks-for-regarding* operator, which formalizes the notion of the restricted speaks-for operator. It is written $B \stackrel{T}{\Rightarrow} A$, and read “ B speaks for A regarding the set of statements in T .” T is any subset of Σ^* . The desired meaning is that when $\sigma \in T$,

$$B \stackrel{T}{\Rightarrow} A \Rightarrow ((B \text{ says } \sigma) \supset (A \text{ says } \sigma))$$

The power of the speaks-for-regarding operator $\stackrel{T}{\Rightarrow}$ is that A can delegate a subset of its authority *without modifying any ACLs*. Contrast the situation with the use of roles in the Calculus for Access Control (see Appendix C.5), where to delegate authority over a restricted subset of her resources, a user had to define a role and install that role in the ACLs of each resource to be shared.

<i>Set</i>	<i>Example members</i>	<i>Description</i>
Σ	s, t	The set of primitive propositions. They represent resources.
Σ^*	σ, τ $s \wedge t$	The set of well-formed formulas (statements) constructed from Σ , $\wedge$, $\neg$, $\mathcal{A}$ says, and $\mathcal{B} \Rightarrow \mathcal{A}$
2^{Σ^*}	S, T, V	The set of sets of statements
P	A, B	The set of primitive principals. They represent agents, including people, machines, programs, and communications channels.
P^*	$\mathcal{A}, \mathcal{B}$ $A \wedge B$	The set of compound principals constructed from P , $\wedge$, $ $, and $\cdot N$
$\mathcal{N}$	N	The set of local names

Table 4.1: *The symbols used to represent sets in this dissertation.*

Restricted speaks-for is transitive:

$$\vdash (\mathcal{C} \xRightarrow{T} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{T} \mathcal{A}) \quad (\text{Axiom E1})$$

We expect the $\wedge$ operation on principals to be monotonic over $\xRightarrow{T}$:

$$\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \wedge \mathcal{C}) \xRightarrow{T} (\mathcal{A} \wedge \mathcal{C}) \quad (\text{Axiom E2})$$

Restricted control over two principals is the same as restricted control over their conjunct:

$$\vdash (\mathcal{C} \xRightarrow{T} \mathcal{A}) \wedge (\mathcal{C} \xRightarrow{T} \mathcal{B}) \equiv \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B}) \quad (\text{Axiom E3})$$

Let $\mathcal{U}$ be the universe of all well-formed formulas; that is, those formulas over which a model $\mathcal{M}$ defines $\mathcal{E}$. Restricted speaks-for degenerates to the original speaks-for when the restriction set is the set of all statements:

$$\vdash (\mathcal{B} \xRightarrow{\mathcal{U}} \mathcal{A}) \equiv (\mathcal{B} \Rightarrow \mathcal{A}) \quad (\text{Axiom E4})$$

If Bob speaks for Alice regarding a set of statements T , he surely speaks for her regarding a subset $T' \subseteq T$:

$$\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{T'} \mathcal{A}) \quad (\text{Axiom E5})$$

Using Axiom E5, a chain of delegations can be collapsed to a single delegation, connecting the head principal in the chain to the tail, whose restriction set is the intersection of the restriction sets of each of the original delegations.

$$\vdash (\mathcal{C} \xRightarrow{S} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{S \cap T} \mathcal{A}) \quad (\text{Theorem E6})$$

Set	Example members	Description
Σ	s, t	The set of primitive propositions. They represent resources.
Σ^*	σ, τ $s \wedge t$	The set of well-formed formulas (statements) constructed from Σ , $\wedge$, $\neg$, $\mathcal{A}$ says, and $\mathcal{B} \Rightarrow \mathcal{A}$
2^{Σ^*}	S, T, V	The set of sets of statements
P	A, B	The set of primitive principals. They represent agents, including people, machines, programs, and communications channels.
P^*	$\mathcal{A}, \mathcal{B}$ $A \wedge B$	The set of compound principals constructed from P , $\wedge$, $ $, and $\cdot N$
$\mathcal{N}$	N	The set of local names

Table 4.1: The symbols used to represent sets in this dissertation.

Restricted speaks-for is transitive:

$$\vdash (\mathcal{C} \xRightarrow{T} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{T} \mathcal{A}) \quad (\text{Axiom E1})$$

We expect the $\wedge$ operation on principals to be monotonic over $\xRightarrow{T}$:

$$\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \wedge \mathcal{C}) \xRightarrow{T} (\mathcal{A} \wedge \mathcal{C}) \quad (\text{Axiom E2})$$

Restricted control over two principals is the same as restricted control over their conjunct:

$$\vdash (\mathcal{C} \xRightarrow{T} \mathcal{A}) \wedge (\mathcal{C} \xRightarrow{T} \mathcal{B}) \equiv \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B}) \quad (\text{Axiom E3})$$

Let $\mathcal{U}$ be the universe of all well-formed formulas; that is, those formulas over which a model $\mathcal{M}$ defines $\mathcal{E}$. Restricted speaks-for degenerates to the original speaks-for when the restriction set is the set of all statements:

$$\vdash (\mathcal{B} \xRightarrow{\mathcal{U}} \mathcal{A}) \equiv (\mathcal{B} \Rightarrow \mathcal{A}) \quad (\text{Axiom E4})$$

If Bob speaks for Alice regarding a set of statements T , he surely speaks for her regarding a subset $T' \subseteq T$:

$$\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{T'} \mathcal{A}) \quad (\text{Axiom E5})$$

Using Axiom E5, a chain of delegations can be collapsed to a single delegation, connecting the head principal in the chain to the tail, whose restriction set is the intersection of the restriction sets of each of the original delegations.

$$\vdash (\mathcal{C} \xRightarrow{S} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{S \cap T} \mathcal{A}) \quad (\text{Theorem E6})$$

This is not to say that $\mathcal{C}$ may not speak for $\mathcal{A}$ regarding more statements than those in the intersection; I address this topic further in Section 7.9.

If we have two restricted delegations from Alice to Bob, we might expect Alice to speak for Bob with respect to the union of the restriction sets. Because of the semantics I choose for $\overset{T}{\Rightarrow}$, however, this intuition does not hold.

$$(B \overset{S}{\Rightarrow} A) \wedge (B \overset{T}{\Rightarrow} A) \not\Rightarrow B \overset{S \cup T}{\Rightarrow} A \quad (\text{Result E7})$$

In Section 4.2.1, I describe a relation weaker than $\overset{T}{\Rightarrow}$ for which the intuitive statement holds.

The quoting operator on principals ($|$) is monotonic in both arguments over $\Rightarrow$. Quoting is still monotonic over $\overset{T}{\Rightarrow}$ in its left argument:

$$\vdash (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset \mathcal{C}|\mathcal{B} \overset{T}{\Rightarrow} \mathcal{C}|\mathcal{A} \quad (\text{Axiom E8})$$

My semantics does not justify monotonicity in the right argument, however:

$$(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \not\Rightarrow \mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C} \quad (\text{Result E9})$$

Hence, when quoting others, principals cannot automatically invoke the same delegated authority they have when speaking directly. The same counterexample that shows Result E9 shows the same property for the weak speaks-for-regarding relation defined in Section 4.2.1, so it seems that the notion of quoting simply does not mix easily with restricted delegation. This result appears to limit the usefulness of quoting, because principals cannot employ quoting with the same ease as in the Calculus for Access Control.

We can salvage some of the convenience of quoting, however, by propagating the quoted principal through the restriction set. Let T^* be the closure of T with respect to the propositional operators $\neg$ and $\wedge$: $T \subseteq T^*$, and if $\sigma, \tau \in T^*$, then $\neg\sigma \in T^*$ and $\sigma \wedge \tau \in T^*$. Furthermore let TC be the closure of T with respect to the modal operator $\mathcal{C}$ says: $T \subseteq TC$, and if $\sigma \in TC$, then $(\mathcal{C} \text{ says } \sigma) \in TC$. Now $(T^*)C$ is the modal closure applied to the propositional closure of some original set T . With these definitions, we can justify this axiom:

$$\vdash \left(\mathcal{B} \overset{(T^*)C}{\Rightarrow} \mathcal{A} \right) \supset \left(\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C} \right) \quad (\text{Axiom E10})$$

When $T = \mathcal{U}$, this axiom reduces to showing right-monotonicity for the original speaks-for relation. This axiom means that $\mathcal{A}$'s restricted delegation to $\mathcal{B}$ must explicitly include any “quotes” of $\mathcal{C}$ about which it is willing to believe $\mathcal{B}$. It seems awkward, but it is a useful result. Why? Because in any possible-worlds semantics wherein $(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C})$ for *all* principals $\mathcal{C}$, the relation representing $\mathcal{A}$ depends on every other principal relation. The introduction of malicious principals with cleverly-chosen relations into such a system can effectively expand T until $T = \mathcal{U}$.

4.2 Semantics of $\overset{T}{\Rightarrow}$

I use a semantics based on possible worlds, modeling a system with a *model* $\mathcal{M} = \langle W, w_0, I, J \rangle$ whose components are defined as in [ABLP93]. The semantic definition of

This is not to say that $\mathcal{C}$ may not speak for $\mathcal{A}$ regarding more statements than those in the intersection; I address this topic further in Section 7.9.

If we have two restricted delegations from Alice to Bob, we might expect Alice to speak for Bob with respect to the union of the restriction sets. Because of the semantics I choose for $\overset{T}{\Rightarrow}$, however, this intuition does not hold.

$$(B \overset{S}{\Rightarrow} A) \wedge (B \overset{T}{\Rightarrow} A) \not\Rightarrow B \overset{S \cup T}{\Rightarrow} A \quad (\text{Result E7})$$

In Section 4.2.1, I describe a relation weaker than $\overset{T}{\Rightarrow}$ for which the intuitive statement holds.

The quoting operator on principals ($|$) is monotonic in both arguments over $\Rightarrow$. Quoting is still monotonic over $\overset{T}{\Rightarrow}$ in its left argument:

$$\vdash (\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \Rightarrow \mathcal{C}|\mathcal{B} \overset{T}{\Rightarrow} \mathcal{C}|\mathcal{A} \quad (\text{Axiom E8})$$

My semantics does not justify monotonicity in the right argument, however:

$$(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \not\Rightarrow \mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C} \quad (\text{Result E9})$$

Hence, when quoting others, principals cannot automatically invoke the same delegated authority they have when speaking directly. The same counterexample that shows Result E9 shows the same property for the weak speaks-for-regarding relation defined in Section 4.2.1, so it seems that the notion of quoting simply does not mix easily with restricted delegation. This result appears to limit the usefulness of quoting, because principals cannot employ quoting with the same ease as in the Calculus for Access Control.

We can salvage some of the convenience of quoting, however, by propagating the quoted principal through the restriction set. Let T^* be the closure of T with respect to the propositional operators $\neg$ and $\wedge$: $T \subseteq T^*$, and if $\sigma, \tau \in T^*$, then $\neg\sigma \in T^*$ and $\sigma \wedge \tau \in T^*$. Furthermore let TC be the closure of T with respect to the modal operator $\mathcal{C}$ says: $T \subseteq TC$, and if $\sigma \in TC$, then $(\mathcal{C} \text{ says } \sigma) \in TC$. Now $(T^*)C$ is the modal closure applied to the propositional closure of some original set T . With these definitions, we can justify this axiom:

$$\vdash \left(\mathcal{B} \overset{(T^*)C}{\Rightarrow} \mathcal{A} \right) \Rightarrow \left(\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C} \right) \quad (\text{Axiom E10})$$

When $T = \mathcal{U}$, this axiom reduces to showing right-monotonicity for the original speaks-for relation. This axiom means that $\mathcal{A}$'s restricted delegation to $\mathcal{B}$ must explicitly include any “quotes” of $\mathcal{C}$ about which it is willing to believe $\mathcal{B}$. It seems awkward, but it is a useful result. Why? Because in any possible-worlds semantics wherein $(\mathcal{B} \overset{T}{\Rightarrow} \mathcal{A}) \Rightarrow (\mathcal{B}|\mathcal{C} \overset{T}{\Rightarrow} \mathcal{A}|\mathcal{C})$ for *all* principals $\mathcal{C}$, the relation representing $\mathcal{A}$ depends on every other principal relation. The introduction of malicious principals with cleverly-chosen relations into such a system can effectively expand T until $T = \mathcal{U}$.

4.2 Semantics of $\overset{T}{\Rightarrow}$

I use a semantics based on possible worlds, modeling a system with a *model* $\mathcal{M} = \langle W, w_0, I, J \rangle$ whose components are defined as in [ABLP93]. The semantic definition of

$\xRightarrow{T}$ is based on the notion of *projecting* a model into a space where only the statements in set T are relevant. The idea behind this definition is that if one were to take the “quotient” of a model M with respect to the dual of T , the resulting model $\overline{M}$ would be concerned only with statements in T . $B \Rightarrow A$ in $\overline{M}$ should be equivalent to $B \xRightarrow{T} A$ in the original model. The model $\overline{M}$ is a projection of M that only preserves information about statements in T .

We begin the construction by defining an equivalence relation $\cong_T: W \times W$ that relates two worlds whenever they agree on all statements in T :

$$w \cong_T w' \text{ iff } (\forall \sigma \in T, w \in \mathcal{E}(\sigma)) \text{ iff } w' \in \mathcal{E}(\sigma) \quad (\text{Definition E11})$$

Then we define the mapping $\phi_T: W \rightarrow \overline{W}$ that takes worlds from the original model to equivalence classes under $\cong_T$:

$$\phi_T(w) = \phi_T(w') \text{ iff } w \cong_T w' \quad (\text{Definition E12})$$

The equivalence classes belong to a set $\overline{W} = 2^T$; notice that worlds (equivalence class representatives) in $\overline{M}$ cannot be confused with those in M . I give a construction of $\phi_T(w)$ in Appendix Section A.1.

Next we extend ϕ_T to the function $\phi_T^w: 2^W \rightarrow 2^{\overline{W}}$ that maps a set of worlds $S_w \subseteq W$ to a set of equivalence class representatives in the projected model:

$$\phi_T^w(S_w) = \{\overline{w} \mid \exists w \in S_w, \overline{w} = \phi_T(w)\} \quad (\text{Definition E13})$$

I use bar notation ($\overline{w}$) to indicate an equivalence class representative (member of a world of a projected model) as opposed to a member of W in the original model.

We can now give a semantic definition of restricted delegation:

$$\begin{aligned} \mathcal{E}(\mathcal{B} \xRightarrow{T} A) \\ = \begin{cases} W & \text{if } \forall w_0 \left(\begin{array}{l} \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \\ \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \end{array} \right) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned} \quad (\text{Definition E14})$$

For the justifications of several of the axioms it is more convenient to shift the projection (ϕ) operation to one side of the subset relation. To do so, we define

$$\phi_T^+(R) = \{\langle w_0, w'_1 \rangle \mid \exists w_1 \cong_T w'_1, \langle w_0, w_1 \rangle \in R\} \quad (\text{Definition E15})$$

Think of ϕ_T^+ as a function that introduces as many edges as it can to a relation without disturbing its projection under T .

We can use ϕ_T^+ to give an equivalent definition of $\xRightarrow{T}$:

$$\mathcal{E}(\mathcal{B} \xRightarrow{T} A) = \begin{cases} W & \text{if } \mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B})) \\ \emptyset & \text{otherwise} \end{cases} \quad (\text{Definition E16})$$

The symbolic gymnastics of moving the projection to the right side of the $\subseteq$ relation is equivalent to the definition in terms of ϕ_T^w , but it makes some of the proofs more concise. The equivalence is shown in Appendix Section A.2.

$\xRightarrow{T}$ is based on the notion of *projecting* a model into a space where only the statements in set T are relevant. The idea behind this definition is that if one were to take the “quotient” of a model M with respect to the dual of T , the resulting model $\overline{M}$ would be concerned only with statements in T . $B \Rightarrow A$ in $\overline{M}$ should be equivalent to $B \xRightarrow{T} A$ in the original model. The model $\overline{M}$ is a projection of M that only preserves information about statements in T .

We begin the construction by defining an equivalence relation $\cong_T: W \times W$ that relates two worlds whenever they agree on all statements in T :

$$w \cong_T w' \text{ iff } (\forall \sigma \in T, w \in \mathcal{E}(\sigma) \text{ iff } w' \in \mathcal{E}(\sigma)) \quad (\text{Definition E11})$$

Then we define the mapping $\phi_T: W \rightarrow \overline{W}$ that takes worlds from the original model to equivalence classes under $\cong_T$:

$$\phi_T(w) = \phi_T(w') \text{ iff } w \cong_T w' \quad (\text{Definition E12})$$

The equivalence classes belong to a set $\overline{W} = 2^T$; notice that worlds (equivalence class representatives) in $\overline{M}$ cannot be confused with those in M . Section A.1 gives a construction of $\phi_T(w)$.

Next we extend ϕ_T to the function $\phi_T^w: 2^W \rightarrow 2^{\overline{W}}$ that maps a set of worlds $S_w \subseteq W$ to a set of equivalence class representatives in the projected model:

$$\phi_T^w(S_w) = \{\overline{w} \mid \exists w \in S_w, \overline{w} = \phi_T(w)\} \quad (\text{Definition E13})$$

I use bar notation ($\overline{w}$) to indicate an equivalence class representative (member of a world of a projected model) as opposed to a member of W in the original model.

We can now give a semantic definition of restricted delegation:

$$\begin{aligned} \mathcal{E}(\mathcal{B} \xRightarrow{T} A) \\ = \begin{cases} W & \text{if } \forall w_0 \left(\begin{array}{l} \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \\ \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \end{array} \right) \\ \emptyset & \text{otherwise} \end{cases} \end{aligned} \quad (\text{Definition E14})$$

For the justifications of several of the axioms it is more convenient to shift the projection (ϕ) operation to one side of the subset relation. To do so, we define

$$\phi_T^+(R) = \{\langle w_0, w_1 \rangle \mid \exists w_1 \cong_T w'_1, \langle w_0, w'_1 \rangle \in R\} \quad (\text{Definition E15})$$

Think of ϕ_T^+ as a function that introduces as many edges as it can to a relation without disturbing its projection under T .

We can use ϕ_T^+ to give an equivalent definition of $\xRightarrow{T}$:

$$\mathcal{E}(\mathcal{B} \xRightarrow{T} A) = \begin{cases} W & \text{if } \mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B})) \\ \emptyset & \text{otherwise} \end{cases} \quad (\text{Definition E16})$$

The symbolic gymnastics of moving the projection to the right side of the $\subseteq$ relation is equivalent to the definition in terms of ϕ_T^w , but it makes some of the proofs more concise. Section A.2 shows the equivalence.

A casual intuition for this definition is that ϕ_T projects from the full model M down to a model in which worlds are only distinguished if they differ with regard to the truth of statements in T . If we collapse away the accessibility arrows that do not say anything about what is happening in T , and A 's relation is a subset of B 's relation in the projection, then A believes everything B believes about statements in T . This intuition is exactly what we want for restricted delegation.

What happens if we take an alternative semantic definition for restricted delegation? I explore one seemingly-natural but undesirable alternative in Appendix A.3. Presently, in Sections 4.2.1 and 4.2.2, I explore two intriguing possibilities.

4.2.1 The $\xrightarrow{T}$ relation

Abadi mentions a weaker version of the speaks-for operator, $B \rightarrow A$,¹ that is true exactly when $B \text{ says } \sigma \supseteq A \text{ says } \sigma$. Note however that A may have some different reason than B to say σ . Semantically, A may consider a totally different set of worlds possible; it just happens that σ is still true in those worlds. For that reason, $B \Rightarrow A$ is a stronger relation than $B \rightarrow A$. The former requires a special (subset) relationship to appear in the model.

I define Abadi's $\rightarrow$ operator as:

$$\mathcal{E}(B \rightarrow A) = \begin{cases} W & \text{if } \forall w \in W, \sigma \in \Sigma^*, \\ & \mathcal{R}(B)(w) \subseteq \mathcal{E}(\sigma) \supseteq \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \\ \emptyset & \text{otherwise} \end{cases}$$

and extend the definition to allow restriction:

$$\mathcal{E}(B \xrightarrow{T} A) = \begin{cases} W & \text{if } \forall w \in W, \sigma \in T, \\ & \mathcal{R}(B)(w) \subseteq \mathcal{E}(\sigma) \supseteq \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \\ \emptyset & \text{otherwise} \end{cases}$$

These are not particularly exciting definitions; they simply state just what the axiom says:

$$(B \rightarrow A) = (B \text{ says } \sigma \supseteq A \text{ says } \sigma)$$

It is obvious that weak speaks-for-regarding degenerates into Abadi's unrestricted $\rightarrow$ operator, since $\mathcal{U} = \Sigma^*$:

$$B \xrightarrow{\mathcal{U}} A \equiv B \rightarrow A$$

4.2.2 The $\xRightarrow{T}$ relation

Having witnessed a weaker relation $\rightarrow$, one may wonder why Abadi et al. preferred a definition of speaks-for ($\Rightarrow$) that was stronger than it needed to be. The intuition seems

¹Read " B weakly speaks for A ."

A casual intuition for this definition is that ϕ_T projects from the full model M down to a model in which worlds are only distinguished if they differ with regard to the truth of statements in T . If we collapse away the accessibility arrows that do not say anything about what is happening in T , and A 's relation is a subset of B 's relation in the projection, then A believes everything B believes about statements in T . This intuition is exactly what we want for restricted delegation.

What happens if we take an alternative semantic definition for restricted delegation? I explore one seemingly-natural but undesirable alternative in Appendix A.3. Presently, in Sections 4.2.1 and 4.2.2, I explore two intriguing possibilities.

4.2.1 The $\xrightarrow{T}$ relation

Abadi mentions a weaker version of the speaks-for operator, $B \rightarrow A$,¹ that is true exactly when B **says** $\sigma \supseteq A$ **says** σ . Note however that A may have some different reason than B to say σ . Semantically, A may consider a totally different set of worlds possible; it just happens that σ is still true in those worlds. For that reason, $B \Rightarrow A$ is a stronger relation than $B \rightarrow A$. The former requires a special (subset) relationship to appear in the model.

I define Abadi's $\rightarrow$ operator as:

$$\mathcal{E}(B \rightarrow A) = \begin{cases} W & \text{if } \forall w \in W, \sigma \in \Sigma^*, \\ & \mathcal{R}(B)(w) \subseteq \mathcal{E}(\sigma) \supseteq \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \\ \emptyset & \text{otherwise} \end{cases}$$

and extend the definition to allow restriction:

$$\mathcal{E}(B \xrightarrow{T} A) = \begin{cases} W & \text{if } \forall w \in W, \sigma \in T, \\ & \mathcal{R}(B)(w) \subseteq \mathcal{E}(\sigma) \supseteq \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \\ \emptyset & \text{otherwise} \end{cases}$$

These are not particularly exciting definitions; they simply state just what the axiom says:

$$(B \rightarrow A) = (B \text{ says } \sigma \supseteq A \text{ says } \sigma)$$

It is obvious that weak speaks-for-regarding degenerates into Abadi's unrestricted $\rightarrow$ operator, since $\mathcal{U} = \Sigma^*$:

$$B \xrightarrow{\mathcal{U}} A \equiv B \rightarrow A$$

4.2.2 The $\xRightarrow{T}$ relation

Having witnessed a weaker relation $\rightarrow$, one may wonder why Abadi et al. preferred a definition of speaks-for ($\Rightarrow$) that was stronger than it needed to be. The intuition seems

¹Read " B weakly speaks for A ."

to be that the stronger semantics captures the fact that A understands B 's *reasons* for believing various statements. My $\xRightarrow{T}$ is “strong” in the same sense, and it degenerates to $\Rightarrow$ when $T = \mathcal{U}$.

My first attempt to build a strong speaks-for-regarding relation, however, ended up too strong. I call the definition below the “*mighty*” speaks-for-regarding:

$$\mathcal{E}(B \xRightarrow{T} A) = \begin{cases} W & \text{if } \forall w \in W, \\ & \mathcal{R}(A)(w) - \bigcap_{\sigma \in T} \mathcal{E}(\sigma) \subseteq \mathcal{R}(B)(w) \\ \emptyset & \text{otherwise} \end{cases}$$

The idea here is that if a subset relationship restricts A to say everything B says, then permitting A 's relation to grow a little permits A to not say some of the statements B says (those not in T). The definition above preserves the desired relationship: if B **says** σ , every world B can see has σ true; because A can only see those worlds and other worlds where σ is true, A **says** σ . So A has no “reasons” (edges to possible worlds) to not say σ that B does not also have.

The $\xRightarrow{T}$ relation seemed promising because it degenerates into $\Rightarrow$ when $T = \mathcal{U}$. It appears too strong, however. For example, if $T = \{s, \neg s\}$, then $\bigcap_T \mathcal{E}(s) = \emptyset$, so any $B \xRightarrow{T} A$ would imply $B \Rightarrow A$. The semantic requirements are so strong that many interesting choices of T are not meaningful. In fact, I present $\xRightarrow{T}$ here precisely because it emphasizes the importance of having a satisfying semantics to lend intuitive meaning to the logic. I explore this issue further in Section 7.9.

4.2.3 Relationships among the relations

Both $\xRightarrow{T}$ and $\Rightarrow$ are strictly stronger than $\xrightarrow{T}$:

$$B \xRightarrow{T} A \supset B \xrightarrow{T} A$$

$$B \xrightarrow{T} A \not\supset B \xRightarrow{T} A$$

$$B \Rightarrow A \supset B \xrightarrow{T} A$$

$$B \xrightarrow{T} A \not\supset B \Rightarrow A$$

The $\xRightarrow{T}$ relation is certainly not stronger than $\Rightarrow$; it seems that it should be strictly weaker, but a corner case prevents it from being so:

$$B \xRightarrow{T} A \not\supset B \Rightarrow A$$

$$B \Rightarrow A \not\supset B \xRightarrow{T} A$$

I establish each of these relationships in **Appendix** Section A.5.

to be that the stronger semantics captures the fact that A understands B 's *reasons* for believing various statements. My $\xRightarrow{T}$ is “strong” in the same sense, and it degenerates to $\Rightarrow$ when $T = \mathcal{U}$.

My first attempt to build a strong speaks-for-regarding relation, however, ended up too strong. I call the definition below the “*mighty*” speaks-for-regarding:

$$\mathcal{E}(B \xRightarrow{T} A) = \begin{cases} W & \text{if } \forall w \in W, \\ & \mathcal{R}(A)(w) - \bigcap_{\sigma \in T} \mathcal{E}(\sigma) \subseteq \mathcal{R}(B)(w) \\ \emptyset & \text{otherwise} \end{cases}$$

The idea here is that if a subset relationship restricts A to say everything B says, then permitting A 's relation to grow a little permits A to not say some of the statements B says (those not in T). The definition above preserves the desired relationship: if B **says** σ , every world B can see has σ true; because A can only see those worlds and other worlds where σ is true, A **says** σ . So A has no “reasons” (edges to possible worlds) to not say σ that B does not also have.

The $\xRightarrow{T}$ relation seemed promising because it degenerates into $\Rightarrow$ when $T = \mathcal{U}$. It appears too strong, however. For example, if $T = \{s, \neg s\}$, then $\bigcap_T \mathcal{E}(s) = \emptyset$, so any $B \xRightarrow{T} A$ would imply $B \Rightarrow A$. The semantic requirements are so strong that many interesting choices of T are not meaningful. In fact, I present $\xRightarrow{T}$ here precisely because it emphasizes the importance of having a satisfying semantics to lend intuitive meaning to the logic. I explore this issue further in Section 7.9.

4.2.3 Relationships among the relations

Both $\xRightarrow{T}$ and $\Rightarrow$ are strictly stronger than $\xrightarrow{T}$:

$$B \xRightarrow{T} A \supset B \xrightarrow{T} A$$

$$B \xrightarrow{T} A \not\supset B \xRightarrow{T} A$$

$$B \Rightarrow A \supset B \xrightarrow{T} A$$

$$B \xrightarrow{T} A \not\supset B \Rightarrow A$$

The $\xRightarrow{T}$ relation is certainly not stronger than $\Rightarrow$; it seems that it should be strictly weaker, but a corner case prevents it from being so:

$$B \xRightarrow{T} A \not\supset B \Rightarrow A$$

$$B \Rightarrow A \not\supset B \xRightarrow{T} A$$

I establish each of these relationships in Section A.5.

5.1.1 Local namespaces.

First, a principal should control the meaning of any names defined relative to itself:

$$\begin{aligned} &\forall \text{ principals } \mathcal{A}, \text{ names } N : \\ &(\mathcal{A} \text{ says } (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N)) \supset (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N) \end{aligned}$$

I do not take this statement as an axiom for the same reason that Abadi, Lampson et al. do not accept the handoff axiom [LABW92, p. 715], [ABLP93, p. 273]. In particular, my semantics does not support it. Instead, as with the handoff axiom, I allow the implementation to assume appropriate instances of it.

5.1.2 Left-monotonicity.

Second, name application should be monotonic over speaks-for. If Alice binds her name “barber” to Bob, and Bob binds his name “butcher” to Charlie, then we want “Alice’s barber’s butcher” to be bound to Charlie.

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset (\mathcal{B} \cdot N \Rightarrow \mathcal{A} \cdot N) \quad (\text{Axiom E17})$$

Using this rule, we can write the following to capture the desired intuition:

$$\begin{aligned} &(\mathcal{B} \Rightarrow \mathcal{A} \cdot N_{\text{barber}}) \supset \\ &\mathcal{B} \cdot N_{\text{butcher}} \Rightarrow \mathcal{A} \cdot N_{\text{barber}} \cdot N_{\text{butcher}} \end{aligned}$$

5.1.3 Distributivity.

We combine the following pair of results

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad (\text{Theorem E18})$$

$$\vdash (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \Rightarrow (\mathcal{A} \wedge \mathcal{B}) \cdot N \quad (\text{Axiom E19})$$

to show that names distribute over principal conjunction:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N = (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad (\text{Theorem E20})$$

Here is a motivating example: If Alice has two doctors Ed and Fred, and Bob visits doctors Fred and George, then who is “(Alice and Bob)’s doctor?”

$$E \Rightarrow A \cdot N_{\text{doctor}}$$

$$F \Rightarrow A \cdot N_{\text{doctor}}$$

$$F \Rightarrow B \cdot N_{\text{doctor}}$$

$$G \Rightarrow B \cdot N_{\text{doctor}}$$

Applying Theorem E20, we conclude:

$$F \Rightarrow (A \wedge B) \cdot N_{\text{doctor}}$$

That is, Fred is the only person who serves as both people’s doctor.

5.1.1 Local namespaces.

First, a principal should control the meaning of any names defined relative to itself:

$$\begin{aligned} &\forall \text{ principals } \mathcal{A} \text{ names } N : \\ &(\mathcal{A} \text{ says } (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N)) \supset (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A} \cdot N) \end{aligned}$$

I do not take this statement as an axiom for the same reason that Abadi, Lampson et al. do not accept the handoff axiom [LABW92, p. 715], [ABLP93, p. 273]. In particular, my semantics does not support it. Instead, as with the handoff axiom, I allow the implementation to assume appropriate instances of it.

5.1.2 Left-monotonicity.

Second, name application should be monotonic over speaks-for. If Alice binds her name “barber” to Bob, and Bob binds his name “butcher” to Charlie, then we want “Alice’s barber’s butcher” to be bound to Charlie.

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset (\mathcal{B} \cdot N \Rightarrow \mathcal{A} \cdot N) \quad (\text{Axiom E17})$$

Using this rule, we can write the following to capture the desired intuition:

$$\begin{aligned} &(\mathcal{B} \Rightarrow \mathcal{A} \cdot N_{\text{barber}}) \supset \\ &\mathcal{B} \cdot N_{\text{butcher}} \Rightarrow \mathcal{A} \cdot N_{\text{barber}} \cdot N_{\text{butcher}} \end{aligned}$$

5.1.3 Distributivity.

We combine the following pair of results

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad (\text{Theorem E18})$$

$$\vdash (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \Rightarrow (\mathcal{A} \wedge \mathcal{B}) \cdot N \quad (\text{Axiom E19})$$

to show that names distribute over principal conjunction:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N = (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad (\text{Theorem E20})$$

Here is a motivating example: If Alice has two doctors Emily and Fred, and Bob visits doctors Fred and George, then who is “(Alice and Bob)’s doctor?”

$$E \Rightarrow A \cdot N_{\text{doctor}}$$

$$F \Rightarrow A \cdot N_{\text{doctor}}$$

$$F \Rightarrow B \cdot N_{\text{doctor}}$$

$$G \Rightarrow B \cdot N_{\text{doctor}}$$

Applying Theorem E20, we conclude:

$$F \Rightarrow (A \wedge B) \cdot N_{\text{doctor}}$$

That is, Fred is the only person who serves as both people’s doctor.

<i>Abadi's notation</i>	<i>My notation</i>
S	Σ
$\mu : S \times \mathcal{W} \rightarrow \{\text{true}, \text{false}\}$	$I : \Sigma \rightarrow 2^{\mathcal{W}}$
$\rho : N \times \mathcal{W} \rightarrow 2^{\mathcal{W}}$	$K : P \times \mathcal{N} \rightarrow 2^{\mathcal{W} \times \mathcal{W}}$
$a \in \mathcal{W}$	$w \in \mathcal{W}$
principals p, q	$\mathcal{A}, \mathcal{B} \in P^*$
$n \in N$	$N \in \mathcal{N}$
$\llbracket n \rrbracket_a = \rho(n, a)$	$\mathcal{R}(\mathcal{A} \cdot N)(w) = K(\mathcal{A}, N)(w)$
$\llbracket p's\ n \rrbracket_a$	$\mathcal{R}(\mathcal{A} \cdot N)(w)$

Table 5.1: *A guide to translating between Abadi's notation and mine*

I initially thought it arbitrary that SPKI allowed restricted authorizations but only unrestricted name bindings. In the light of my current semantics, however, I cannot justify extending name bindings to allow restriction.

5.4 Abadi's semantics for linked local namespaces

Abadi gives an alternate logic and semantics for SPKI-style linked local namespaces [Aba98]. (He refers to SDSI, from which SPKI 2.0 derives.) Abadi's notation diverges from that used in the Calculus for Access Control [ABLP93], but the semantics are the same. Table 5.1 helps translate the notation. My semantics differs in three interesting ways.

First, SPKI has special global names, so that if N_G is a global name, $\mathcal{A} \cdot N_G = N_G$. The result is that the same syntactic construct can be used to bind a local name to another local name or to a globally-specified name. All names in linking statements are implicitly prefixed by the name of the speaking principal; but if the explicitly mentioned name is global, the prefix has no consequence. I consider this syntactic sugar, and leave it to an implementation to determine from explicit cues (such as a key specification or a SDSI global name with the special !! suffix) whether a mentioned principal should be interpreted as local to the speaker.

Second, Abadi's logic adopts the handoff rule for names, which he calls the “Linking” axiom. Here it is, translated to my terminology:

$$\mathcal{A} \text{ says } (\mathcal{B} \Rightarrow (\mathcal{A} \cdot N)) \supset (\mathcal{B} \Rightarrow (\mathcal{A} \cdot N))$$

He validates the axiom by the use of composition to model name application, with which I disagree.

The third and most important way my semantics differs from Abadi's is that Abadi's semantics models name application as quoting (composition). Each unqualified (local) name is mapped to a single relation. This property can introduce crosstalk between otherwise unconnected principals; recall the example from Section 4.3.2. Even when a name relation is not constrained to be a role, the same problem arises. For example, let N represent the name “doctor.” Imagine that Bob assigns Charlie to be his doctor: $C \Rightarrow B|N$. This is fine; Charlie should be able to do some things on Bob's behalf, but not everything: If $B|N \xRightarrow{T} B$, then Charlie can do the things in T .

<i>Abadi's notation</i>	<i>My notation</i>
S	Σ
$\mu : S \times \mathcal{W} \rightarrow \{\text{true}, \text{false}\}$	$I : \Sigma \rightarrow 2^{\mathcal{W}}$
$\rho : N \times \mathcal{W} \rightarrow 2^{\mathcal{W}}$	$K : P \times \mathcal{N} \rightarrow 2^{\mathcal{W} \times \mathcal{W}}$
$a \in \mathcal{W}$	$w \in \mathcal{W}$
principals p, q	$\mathcal{A}, \mathcal{B} \in P^*$
$n \in N$	$N \in \mathcal{N}$
$\llbracket n \rrbracket_a = \rho(n, a)$	$\mathcal{R}(\mathcal{A} \cdot N)(w) = K(\mathcal{A}, N)(w)$
$\llbracket p's\ n \rrbracket_a$	$\mathcal{R}(\mathcal{A} \cdot N)(w)$

Table 5.1: *A guide to translating between Abadi's notation and mine*

I initially thought it arbitrary that SPKI allowed restricted authorizations but only unrestricted name bindings. In the light of my current semantics, however, I cannot justify extending name bindings to allow restriction.

5.4 Abadi's semantics for linked local namespaces

Abadi gives an alternate logic and semantics for SPKI-style linked local namespaces [Aba98]. (He refers to SDSI, from which SPKI 2.0 derives.) Abadi's notation diverges from that used in the Calculus for Access Control [ABLP93], but the semantics are the same. Table 5.1 helps translate the notation. My semantics differs in three interesting ways.

First, SPKI has special global names, so that if N_G is a global name, $\mathcal{A} \cdot N_G = N_G$. The result is that the same syntactic construct can be used to bind a local name to another local name or to a globally-specified name. All names in linking statements are implicitly prefixed by the name of the speaking principal; but if the explicitly mentioned name is global, the prefix has no consequence. I consider this syntactic sugar, and leave it to an implementation to determine from explicit cues (such as a key specification or a SDSI global name with the special !! suffix) whether a mentioned principal should be interpreted as local to the speaker.

Second, Abadi's logic adopts the handoff rule for names, which he calls the “Linking” axiom. Here it is, translated to my terminology:

$$\mathcal{A} \text{ says } (\mathcal{B} \Rightarrow (\mathcal{A} \cdot N)) \supset (\mathcal{B} \Rightarrow (\mathcal{A} \cdot N))$$

He validates the axiom by the use of composition to model name application, with which I disagree.

The third and most important way my semantics differs from Abadi's is that Abadi's semantics models name application as quoting (composition). Each unqualified (local) name is mapped to a single relation. This property can introduce crosstalk between otherwise unconnected principals; recall the example from Section 4.3.2. Even when a name relation is not constrained to be a role, the same problem arises. For example, let N represent the name “doctor.” Imagine that Bob assigns Charlie to be his doctor: $C \Rightarrow B|N$. This is fine; Charlie should be able to do some things on Bob's behalf, but not everything: If $B|N \xRightarrow{T} B$, then Charlie can do the things in T .

Chapter 6

The semantics of authorization tag notation

My logic requires a notation for propositions and sets of propositions. An important part of SPKI are user-defined *tags* that can represent either. SPKI tags are attractive because, when used as sets, users may extend them to arbitrarily fine granularity, ensuring that users can always refine delegations they wish to share. SPKI provides a prose description of how tags are to be intersected, which gives the user a rough idea about the meaning of tags.

In this chapter, I derive a formal language for tags and show that tags are (almost) closed under intersection. Besides clarifying the SPKI notion of tag, the semantics serves two important purposes: First, the proofs of closure imply an unambiguous implementation of tag intersection. Second, I show that tags have a desirable property that I depend upon for my formalization of SPKI's properties in Chapter 7.9.

6.1 Overview

One confusing aspect of tags is that they serve to represent both single requests and delegation restrictions (sets of requests to be permitted). How do they do this? The short answer is containment. Most tags represent an infinite set of finite strings. Let us call those strings *atoms*, the indivisible unit of privilege. A request to be verified is a set of atoms; the intuition is that executing the request requires that the requester have at least a certain set of privileges. A permission describes another set of atoms. If the permission's set contains the request's set, then the permission grants at least each of the atoms required for the request.

Why use infinite sets of atoms? For extensibility. In general, a permission is represented by a tag, an infinite set of atoms. Therefore, it can always be further subdivided into more-specific permissions, each still represented by an infinite subset of the original permission's atoms.

Given this motivation for the structure of tags, I construct tags from the ground up using grammars. I begin by defining bytestrings and the atomic finite strings I have called atoms.

Chapter 6

The semantics of authorization tag notation

My logic requires a notation for propositions and sets of propositions. An important part of SPKI are user-defined *tags* that can represent either. SPKI tags are attractive because, when used as sets, users may extend them to arbitrarily fine granularity, ensuring that users can always refine delegations they wish to share. SPKI provides a prose description of how tags are to be intersected, which gives the user a rough idea about the meaning of tags.

In this chapter, I derive a formal language for tags and show that tags are (almost) closed under intersection. In addition to clarifying the SPKI notion of tag, the semantics serves two important purposes: First, the proofs of closure imply an unambiguous implementation of tag intersection. Second, the semantics show that tags have a desirable property I depend upon for my formalization of SPKI's properties in Section 7.9.

6.1 Overview

One confusing aspect of tags is that they serve to represent both single requests and delegation restrictions (sets of requests to be permitted). How do they do this? The short answer is containment. Most tags represent an infinite set of finite strings. Let us call those strings *atoms*, the indivisible unit of privilege. A request to be verified is a set of atoms; the intuition is that executing the request requires that the requester have at least a certain set of privileges. A permission describes another set of atoms. If the permission's set contains the request's set, then the permission grants at least each of the atoms required for the request.

Why use infinite sets of atoms? For extensibility. In general, a permission is represented by a tag, an infinite set of atoms. Therefore, it can always be further subdivided into more-specific permissions, each still represented by an infinite subset of the original permission's atoms.

Given this motivation for the structure of tags, I construct tags from the ground up using grammars. I begin by defining bytestrings and the atomic finite strings I have called atoms.

6.2 Bytestrings and atoms

Let Σ be the natural alphabet of our system; in our case, let it be the set of 256 octets. Let $\mathbf{B}$ be the set of all finite-length strings of octets, Σ^* :

$$\mathbf{B} := \sigma \mathbf{B} \quad (\forall \sigma \in \Sigma)$$

ε

This is the set SPKI calls “bytestrings.” Next we extend Σ with three metasymbols to unambiguously demarcate the list structure of an atom: $\Sigma' = \Sigma \cup \{\langle, _, \rangle\}$. Now define the mutually-recursive sets of expressions ($\mathbf{E}$), lists of expressions ($\mathbf{L}$), and non-empty lists of expressions ($\mathbf{N}$):

$$\begin{aligned} \mathbf{E} &:= \mathbf{B} \\ &\quad | \langle \mathbf{L} \rangle \\ \mathbf{L} &:= \mathbf{N} \\ &\quad | \varepsilon \\ \mathbf{N} &:= \mathbf{E} _ \mathbf{N} \\ &\quad | \mathbf{E} \end{aligned}$$

The gymnastics with the non-empty lists serve to prevent lists from ending with a $_$ just before the $\rangle$; this feels right but it is not important for the development of the semantics.

We define the set of all atoms to be exactly the set of expressions derived by the grammar for $\mathbf{E}$. Notice that every atom is a finite object with an unambiguous tree structure defined by the special delimiters $\{\langle, _, \rangle\}$. Every internal node is a list, and every leaf node is a bytestring.

6.3 Auths

Recall that a tag, which specifies either a request or a set of delegated permissions, represents a (usually infinite) set of atoms. A tag is the finite notation for a request or a restriction set; I call the (usually infinite) set that a given tag represents an *auth*. Our next task is to define the set of *auths*, each a subset of $\mathbf{E}$. In the next section, I show that the set of SPKI tags is, with a few exceptions, isomorphic to the set of auths I define here.

6.2 Bytestrings and atoms

Let Σ be the natural alphabet of our system; in our case, let it be the set of 256 octets. Let $\mathbf{B}$ be the set of all finite-length strings of octets, Σ^* :

$$\mathbf{B} := \epsilon \mid \sigma \mathbf{B} \quad (\forall \sigma \in \Sigma)$$

This is the set SPKI calls “bytestrings.” Next we extend Σ with three metasymbols to unambiguously demarcate the list structure of an atom: $\Sigma' = \Sigma \cup \{ \langle \cdot \rangle, \langle \cdot \rangle, \langle \cdot \rangle \}$. Now define the mutually-recursive sets of expressions ($\mathbf{E}$), lists of expressions ($\mathbf{L}$), and non-empty lists of expressions ($\mathbf{N}$):

$$\begin{aligned} \mathbf{E} &:= \mathbf{B} \\ &\quad \mid \langle \mathbf{L} \rangle \\ \mathbf{L} &:= \mathbf{N} \\ &\quad \mid \epsilon \\ \mathbf{N} &:= \mathbf{E} _ \mathbf{N} \\ &\quad \mid \mathbf{E} \end{aligned}$$

The gymnastics with the non-empty lists serve to prevent lists from ending with a $_$ just before the $\rangle$; this feels right but it is not important for the development of the semantics.

We define the set of all atoms to be exactly the set of expressions derived by the grammar for $\mathbf{E}$. Notice that every atom is a finite object with an unambiguous tree structure defined by the special delimiters $\{ \langle \cdot \rangle, \langle \cdot \rangle, \langle \cdot \rangle \}$. Every internal node is a list, and every leaf node is a bytestring.

6.3 Auths

Recall that a tag, which specifies either a request or a set of delegated permissions, represents a (usually infinite) set of atoms. A tag is the finite notation for a request or a restriction set; I call the (usually infinite) set that a given tag represents an *auth*. Our next task is to define the set of *auths*, each a subset of $\mathbf{E}$. In the next section, I show that the set of SPKI tags is, with a few exceptions, isomorphic to the set of auths I define here.

How can we escape the dilemma? We can omit the **range** special form, but that would not provide a satisfying model of SPKI. We could introduce an intersection operator analogous to the union operator A_{set} , but that would be a hack. Since A is otherwise closed under intersection, an intersection operator should never be used except when intersecting these curious bytestring expressions, for it would only lead to needlessly larger representations for auths. The final option, that I adopt, is to accept the incompleteness of tags. Assume $t_3 = t_1 \cap t_2$, that is, the tag-intersection procedure run on tags t_1 and t_2 produces tag t_3 . Let $A(t)$ be a function mapping a tag to the auth it represents, a typically-infinite subset of **E**. If tags are complete, then $A(t_3) = A(t_1) \cap A(t_2)$. If we must sacrifice the completeness of tags, we still know that $A(t_3) \subseteq A(t_1) \cap A(t_2)$. This provides assurance that the authorization procedure is at least still sound: we will not conclude a chained delegation confers authority (atoms) that is not conferred by both members of the chain.

Treating the intersection of a **range** with a **range** or **prefix** as null should not be terribly limiting in practice. When either form is used, it is specifying a value for some field with a particular interpretation; it is likely that in a real system any given field would only have one meaningful mode of comparison.

6.5.4 Semantics of special tags

SPKI contains several special forms for tags: **(*)** represents the auth A_* . **(* set ...)** represents the auth $A_{set}(\dots)$. **(* prefix ...)** and **(* range ...)** represent (possibly infinite) sets of bytestrings. To model these tags, we need to extend the base definition of A :

$$A_{prefix}(p) = \{b \mid b = ps, s \in \Sigma\} \quad (\text{Definition T8})$$

$$A_{range}(f) = \{b \mid f(b) = true\} \quad (\text{Definition T9})$$

$$A_0 = \begin{array}{l} A_{null} \\ \cup \\ \bigcup_{b \in \mathbf{B}} A_* \\ \bigcup_{p \in \mathbf{B}} A_{prefix}(p) \\ \bigcup_{b \in f} A_{range}(f) \end{array} \quad (\text{Definition T10})$$

The function $f : \Sigma^* \rightarrow \{true, false\}$ selects a range of bytestrings, and is used here as an abbreviation to hide that complexity. The function depends on the specified ordering (alpha, numeric, time, binary, or date), the (optional) low and high bounds, and bits specifying whether each bound is exclusive or inclusive.

The matrix of intersection cases must now be extended to support the new possibilities (see Table 6.2). This extension of course will no longer show the completeness of A under intersection (unless A_{range} is removed). But it is still useful as a thorough guide to intersecting tags.

Case VII. In this case, assume $a_x = A_{bs}(b) = \{b\}$ and $a_y = A_{prefix}(p)$ or $a_y = A_{range}(f)$. This case is similar to case III: if $b \in a_y$, then the intersection is $a_z = A_{bs}(b) = a_x$; otherwise it is $\emptyset = A_{null}$.

Case VIII. We have $a_x = A_{prefix}(p_x)$ and $a_y = A_{prefix}(p_y)$. Assume WOLOG $|p_x| > |p_y|$ (p_x is a longer string). If p_y is a prefix of p_x (that is, $p_x = p_y s$), then $a_x \cap a_y = a_x$: if $b \in a_x$,

How can we escape the dilemma? We can omit the **range** special form, but that would not provide a satisfying model of SPKI. We could introduce an intersection operator analogous to the union operator A_{set} , but that would be a hack. Since A is otherwise closed under intersection, an intersection operator should never be used except when intersecting these curious bytestring expressions, for it would only lead to needlessly larger representations for auths. The final option, that I adopt, is to accept the incompleteness of tags. Assume $t_3 = t_1 \cap t_2$, that is, the tag-intersection procedure run on tags t_1 and t_2 produces tag t_3 . Let $A(t)$ be a function mapping a tag to the auth it represents, a typically-infinite subset of **E**. If tags are complete, then $A(t_3) = A(t_1) \cap A(t_2)$. If we must sacrifice the completeness of tags, we still know that $A(t_3) \subseteq A(t_1) \cap A(t_2)$. This provides assurance that the authorization procedure is at least still sound: we will not conclude a chained delegation confers authority (atoms) that is not conferred by both members of the chain.

Treating the intersection of a **range** with a **range** or **prefix** as null should not be terribly limiting in practice. When either form is used, it is specifying a value for some field with a particular interpretation; it is likely that in a real system any given field would only have one meaningful mode of comparison.

6.5.4 Semantics of special tags

SPKI contains several special forms for tags: **(*)** represents the auth A_* . **(* set ...)** represents the auth $A_{set}(\dots)$. **(* prefix ...)** and **(* range ...)** represent (possibly infinite) sets of bytestrings. To model these tags, we need to extend the base definition of A :

$$A_{prefix}(p) = \{b \mid b = ps, s \in \Sigma\} \quad (\text{Definition T8})$$

$$A_{range}(f) = \{b \mid f(b) = true\} \quad (\text{Definition T9})$$

$$A_0 = \begin{array}{l} A_{null} \\ \cup \\ \bigcup_{b \in \mathbf{B}} A_* \\ \bigcup_{p \in \mathbf{B}} A_{prefix}(p) \\ \bigcup_{b \in f} A_{range}(f) \end{array} \quad (\text{Definition T10})$$

The function $f : \Sigma^* \rightarrow \{true, false\}$ selects a range of bytestrings, and is used here as an abbreviation to hide that complexity. The function depends on the specified ordering (alpha, numeric, time, binary, or date), the (optional) low and high bounds, and bits specifying whether each bound is exclusive or inclusive.

The matrix of intersection cases must now be extended to support the new possibilities (see Table 6.2). This extension of course will no longer show the completeness of A under intersection (unless A_{range} is removed). But it is still useful as a thorough guide to intersecting tags.

Case VII. In this case, assume $a_x = A_{bs}(b) = \{b\}$ and $a_y = A_{prefix}(p)$ or $a_y = A_{range}(f)$. This case is similar to case III: if $b \in a_y$, then the intersection is $a_z = A_{bs}(b) = a_x$; otherwise it is $\emptyset = A_{null}$.

Case VIII. We have $a_x = A_{prefix}(p_x)$ and $a_y = A_{prefix}(p_y)$. Assume WOLOG $|p_x| > |p_y|$ (p_x is a longer string). If p_y is a prefix of p_x (that is, $p_x = p_y s$), then $a_x \cap a_y = a_x$: if $b \in a_x$,

intersection of the atoms). If the tags have a null intersection, we treat that object just like any other; however, because it is an empty set of atoms, it is worthless in that no request will be authorized by it.

SPKI's `AIntersect` procedure for tag intersection suggests that an delegation tag includes a permission if the intersection of the two is non-null. My tag semantics tells us that to ensure that such intersection is meaningful, one must ensure that all permissions are finer-grained than the sets described by any delegation. Such restriction hampers the elegant extensibility of tags. In contrast, when authorizing requests, we intersect the delegation tag with the request tag, *and test whether the result equals the request tag*. If so, we conclude that all of the (typically infinite) set of atoms demanded by the request are granted by the delegation. If not, we conclude that some smaller (possibly empty) set of atoms were granted; in any case, they are not sufficient to justify granting the request.

Set containment provides a concrete, mathematically sound interpretation for specifying authorizations in an infinitely-extensible fashion.

6.7 Order dependence

The semantics of SPKI tags specifically depend on the order of elements in a list; intersection of two lists involves pairwise intersection of each list's elements. Because lists are implicitly followed by an arbitrarily-long supply of A_* s, lists are extensible in that a new property can be defined for the list and assigned to the next unused position in the list.

For example, imagine that one is defining a tag format for delegating access to a database of employee records. A first-cut tag format might look like:

```
(employee
  (id (*))
  (salary (*))
)
```

Such a definition gives users of the system the ability to delegate to others rights such as the right to inspect only employee records with a specific ID number (`employee (id 01247) (*)`), or those of employees earning more than \$50,000 (`employee (*) (salary (range gt 50000))`). Because tags are extensible, one may later decide that the ability to select employees based on anniversary year is useful, so the definition is extended to:

```
(employee
  (id (*))
  (salary (*))
  (anniversary (*))
)
```

Because list tags are followed by implicit $(*)$ members, all existing delegation tags continue to be meaningful even when the new format is deployed.

What happens, however, if two independent organizations want to extend the format independently? In our example, perhaps one department of the corporation wishes to add the anniversary extension (and does so for their internal applications), and another department

intersection of the atoms). If the tags have a null intersection, we treat that object just like any other; however, because it is an empty set of atoms, it is worthless in that no request will be authorized by it.

SPKI's `AIntersect` procedure for tag intersection suggests that a delegation tag includes a permission if the intersection of the two is non-null. My tag semantics tells us that to ensure that such intersection is meaningful, one must ensure that all permissions are finer-grained than the sets described by any delegation. Such restriction hampers the elegant extensibility of tags. In contrast, when authorizing requests, we intersect the delegation tag with the request tag, *and test whether the result equals the request tag*. If so, we conclude that all of the (typically infinite) set of atoms demanded by the request are granted by the delegation. If not, we conclude that some smaller (possibly empty) set of atoms were granted; in any case, they are not sufficient to justify granting the request.

Set containment provides a concrete, mathematically sound interpretation for specifying authorizations in an infinitely-extensible fashion.

6.7 Order dependence

The semantics of SPKI tags specifically depend on the order of elements in a list; intersection of two lists involves pairwise intersection of each list's elements. Because lists are implicitly followed by an arbitrarily-long supply of A_* s, lists are extensible in that a new property can be defined for the list and assigned to the next unused position in the list.

For example, imagine that one is defining a tag format for delegating access to a database of employee records. A first-cut tag format might look like:

```
(employee
  (id (*))
  (salary (*))
)
```

Such a definition gives users of the system the ability to delegate to others rights such as the right to inspect only employee records with a specific ID number (`employee (id 01247) (*)`), or those of employees earning more than \$50,000 (`employee (*) (salary (range gt 50000))`). Because tags are extensible, one may later decide that the ability to select employees based on anniversary year is useful, so the definition is extended to:

```
(employee
  (id (*))
  (salary (*))
  (anniversary (*))
)
```

Because list tags are followed by implicit $(*)$ members, all existing delegation tags continue to be meaningful even when the new format is deployed.

What happens, however, if two independent organizations want to extend the format independently? In our example, perhaps one department of the corporation wishes to add the anniversary extension (and does so for their internal applications), and another department

to mean that the revalidation principal R must verify that this certificate (with hash H_1) is valid. Principal R signs a revalidation instrument I with a short validity interval T_V

$$R \text{ says } I \xRightarrow{T_V} R$$

and a given revalidation instrument would agree with all valid outstanding certificates:

$$\begin{aligned} I \text{ says } \mathbf{0} &\Rightarrow I|H_1 \\ I \text{ says } \mathbf{0} &\Rightarrow I|H_2 \\ &\vdots \end{aligned}$$

The principal $\mathbf{0}$ has relation $\mathcal{R}(\mathbf{0}) = \emptyset$, so that every principal speaks for $\mathbf{0}$. Using the logic, we can reason that

$$\mathbf{0} \Rightarrow I|H_1 \xRightarrow{T_V} R|H_1$$

and since $B = B \wedge \mathbf{0}$, $B \xRightarrow{T_V} A$. Notice the treatment of a certificate's hash as a principal. In the logic, principals are general entities and can be used to represent many objects and actors.

Negative certificate revocation lists can be handled similarly; an implementation examining a revocation list would conclude $I \text{ says } \mathbf{0} \Rightarrow I|H_1$ for any H_1 *not* present in the list.

One-time revalidations are meant to be interpreted as having a zero validity interval. A system verifying a request s creates a nonce E , understanding $E \text{ says } s$, and sends it to the revalidator R . R replies with a statement meant to be interpreted

$$R \text{ says } E \xRightarrow{\{s\}} R|H_1$$

Now both B and $E \xRightarrow{\{s\}} R|H_1$ say s , so $A \text{ says } s$. Any future request of the same sort will require another revalidation, for its s will have a different effective time.

7.8 Safe extensions

My semantics suggests that SPKI may be safely extended to support a variety of principals other than public keys. Channels protected by secret keys or a trusted computing base, for example, are easily modeled as principals in the logic. In the examples in this dissertation, I represent principals with symbolic names. Real principals, however, are represented by some mechanism that can verify that a given request comes from a particular principal. Examples of mechanisms for authenticating users include the UID mechanism in Unix, the Kerberos authentication server, and public key cryptography. Lampson et al. show how many common system components can be modeled as principals [LABW92].

Compound principals let us represent useful trust relationships other than delegation. A conjunct principal $(A \wedge B)$, for example, represents a principal that only believes σ when both A and B believe σ . Hence, a delegation to a conjunct principal is analogous to a check

to mean that the revalidation principal R must verify that this certificate (with hash H_1) is valid. Principal R signs a revalidation instrument I with a short validity interval T_V

$$R \text{ says } I \xRightarrow{T_V} R$$

and a given revalidation instrument would agree with all valid outstanding certificates:

$$I \text{ says } \mathbf{0} \Rightarrow I|H_1$$

$$I \text{ says } \mathbf{0} \Rightarrow I|H_2$$

$$\vdots$$

The principal $\mathbf{0}$ has relation $\mathcal{R}(\mathbf{0}) = \emptyset$, so that every principal speaks for $\mathbf{0}$. Using the logic, we can reason that

$$\mathbf{0} \Rightarrow I|H_1 \xRightarrow{T_V} R|H_1$$

and since $B = B \wedge \mathbf{0}$, $B \xRightarrow{T_V} A$. Notice the treatment of a certificate's hash as a principal. In the logic, principals are general entities and can be used to represent many objects and actors.

Negative certificate revocation lists can be handled similarly; an implementation examining a revocation list would conclude $I \text{ says } \mathbf{0} \Rightarrow I|H_1$ for any H_1 *not* present in the list.

One-time revalidations are meant to be interpreted as having a zero validity interval. A system verifying a request s creates a nonce E , understanding $E \text{ says } s$, and sends it to the revalidator R . R replies with a statement meant to be interpreted

$$R \text{ says } E \xRightarrow{\{s\}} R|H_1$$

Now both B and $E \xRightarrow{\{s\}} R|H_1$ say s , so $A \text{ says } s$. Any future request of the same sort will require another revalidation, for its s will have a different effective time.

7.8 Safe extensions

My semantics suggests that SPKI may be safely extended to support a variety of principals other than public keys. Channels protected by secret keys or a trusted computing base, for example, are easily modeled as principals in the logic. In the examples in this dissertation, I represent principals with symbolic names. Real principals, however, are represented by some mechanism that can verify that a given request comes from a particular principal. Examples of mechanisms for authenticating users include the UID mechanism in Unix, the Kerberos authentication server, and public key cryptography. Lampson et al. show that many common system components can be modeled as principals [LABW92].

Compound principals let us represent useful trust relationships other than delegation. A conjunct principal $(A \wedge B)$, for example, represents a principal that only believes σ when both A and B believe σ . Hence a delegation to a conjunct principal is analogous to a check

With these semantics, one might fear that no restriction is actually meaningful. How might we escape it? One option is to abandon the **K** axiom ($\mathcal{A} \text{ believes } s \wedge \mathcal{A} \text{ believes } (s \supset t) \supset \mathcal{A} \text{ believes } t$), so that principals no longer believe every consequence of their beliefs. This option is undesirable because it cripples the logic to only operate outside the scope of belief operators.

A second option is to both disallow negative statements in restriction sets and to use the weaker $\mathcal{B} \xrightarrow{T} \mathcal{A}$ relation instead of $\mathcal{B} \xRightarrow{T} \mathcal{A}$ to model delegation.

A third option is to prevent principals from making contradictory statements. This is difficult in general in a distributed system. One approach is to prevent principals from making negative statements at all. SPKI takes this approach. Its tags, which represent both restriction sets and individual statements, cannot represent both a statement and its logical negation. I provide a formal treatment of tags in Chapter 6.

Another extension might be to allow SPKI name bindings (4-tuples) to include authorization restrictions. As mentioned in Section 5.3, my semantics suggests that this seemingly-natural extension has undesirable consequences.

I conclude that in certain dimensions, SPKI is as strong as it can be. Changing SPKI by allowing principals to make negative statements or by allowing negative statements in restriction sets would push SPKI “over the edge,” making its restrictions meaningless. Those proposing to augment SPKI or other systems based on a logic of restricted delegation such as that of Chapter 4 must be wary of this hazard.

7.10 Related work

Abadi provides a semantics for SPKI names in [Aba98], but its definition shares a flaw with that used for roles in [ABLP93]. I discuss Abadi’s name semantics in Section 5.4.

Halpern and van der Meyden supply an alternate semantics for SPKI names [HvdM99], but it only encompasses the containment relation among names, and does not treat names as principals. As a result, it cannot relate names to compound principals nor relate names to other principals that are only connected by a restricted delegation.

Aura supplies a semantics for SPKI restricted delegation in [Aur98], but it is unsatisfying in that it essentially says what the reduction procedure says: a delegation is in place if there is a chain of delegation certificates and principals. It does not lend intuition about what the delegations mean. In contrast, my semantics connects restricted delegation to the logic of belief, a formal model that describes what a principal means when it delegates authority.

With these semantics, one might fear that no restriction is actually meaningful. How might we escape it? One option is to abandon the **K** axiom ($\mathcal{A} \text{ believes } s \wedge \mathcal{A} \text{ believes } (s \supset t) \supset \mathcal{A} \text{ believes } t$), so that principals no longer believe every consequence of their beliefs. This option is undesirable because it cripples the logic to only operate outside the scope of belief operators.

A second option is to both disallow negative statements in restriction sets and to use the weaker $\mathcal{B} \xrightarrow{T} \mathcal{A}$ relation instead of $\mathcal{B} \xRightarrow{T} \mathcal{A}$ to model delegation.

A third option is to prevent principals from making contradictory statements. This is difficult in general in a distributed system. One approach is to prevent principals from making negative statements at all. SPKI takes this approach. Its tags, which represent both restriction sets and individual statements, cannot represent both a statement and its logical negation. I provide a formal treatment of tags in Chapter 6.

Another extension might be to allow SPKI name bindings (4-tuples) to include authorization restrictions. As mentioned in Section 5.3, my semantics suggests that this seemingly-natural extension has undesirable consequences.

I conclude that in certain dimensions, SPKI is as strong as it can be. Changing SPKI by allowing principals to make negative statements or by allowing negative statements in restriction sets would push SPKI “over the edge,” making its restrictions meaningless. Those proposing to augment SPKI, or other systems based on a logic of restricted delegation such as that of Chapter 4, must be wary of this hazard.

7.10 Related work

Abadi provides a semantics for SPKI names in [Aba98], but its definition shares a flaw with that used for roles in [ABLP93]. I discuss Abadi’s name semantics in Section 5.4.

Halpern and van der Meyden supply an alternate semantics for SPKI names [HvdM99], but it only encompasses the containment relation among names, and does not treat names as principals. As a result, it cannot relate names to compound principals nor relate names to other principals that are only connected by a restricted delegation.

Aura supplies a semantics for SPKI restricted delegation [Aur98], but it is unsatisfying in that it essentially says what the reduction procedure says: a delegation is in place if there is a chain of delegation certificates and principals. It does not lend intuition about what the delegations mean. In contrast, my semantics connects restricted delegation to the logic of belief, a formal model that describes what a principal means when it delegates authority.

authority? My logic gives system administrators or other interested parties two tools to monitor downstream sharing. First, conjunction principals let the system administrator (or his agent) remain involved in any transactions Alice performs using the resources she receives from the administrator. Second, since access control decisions based on my logic involve a proof of authority, the administrator has the ability to comprehensively audit how the resources under his care have been delegated.

8.2 Spanning network scales

A second boundary that interferes with end-to-end authorization is network scale. Network scale affects an application's choice of hop-by-hop authorization protocol. For example, a strong encryption protocol is appropriate when crossing a wide-area network. Inside a firewall where routers are locally administered, some installations may base authority decisions on IP source addresses. On a local machine, we can often trust the OS kernel to correctly identify the participants in an interprocess communication.

If a service exploits my unified authorization model, the server uses a single API to reason about authority. One can plug in protocols (hop-by-hop authorization mechanisms), and the server can reason about the authorizations supported by each protocol. If the server does not support a given mechanism natively, it can still reason about authorization information from a protocol-translating gateway that passes through end-to-end authorization information using my system.

Figure 8.3 illustrates how we reason about authorization at different network scales. When Alice is at the same machine as the resource server (a), the server trusts the kernel to reliably report her identity, from which it derives her authority. Objects in the logic encode the statements “the kernel says this interprocess communication (IPC) channel speaks for Alice” and “the server says the kernel speaks the truth.” The server uses the authorization library to conclude that requests arriving on the IPC channel represent Alice's wishes.

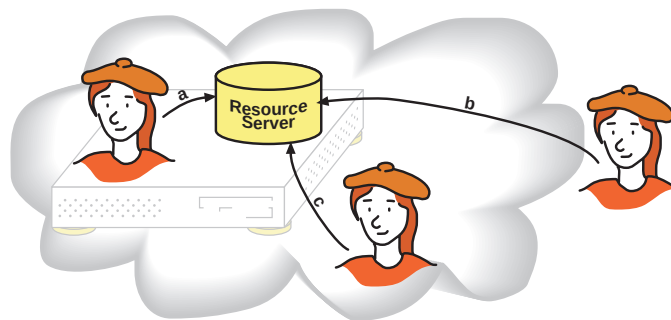


Figure 8.3: When Alice connects to a resource server from the same machine, the server may trust the kernel to correctly identify her (a). When Alice connects from elsewhere on the Internet, the server may require strong encryption to verify her authority (b). When Alice connects from inside a department firewall, the server may accept IP source addresses as sufficient proof of her authority (c).

In the wide-area network case (b), a standard encryption-based authentication protocol

authority? My logic gives system administrators or other interested parties two tools to monitor downstream sharing. First, conjunction principals let the system administrator (or his agent) remain involved in any transactions Alice performs using the resources she receives from the administrator. Second, since access-control decisions based on my logic involve a proof of authority, the administrator has the ability to comprehensively audit how the resources under his care have been delegated.

8.2 Spanning network scales

A second boundary that interferes with end-to-end authorization is network scale. Network scale affects an application's choice of hop-by-hop authorization protocol. For example, a strong encryption protocol is appropriate when crossing a wide-area network. Inside a firewall where routers are locally administered, some installations may base authority decisions on IP source addresses. On a local machine, we can often trust the OS kernel to correctly identify the participants in an interprocess communication.

If a service exploits my unified authorization model, the server uses a single API to reason about authority. One can plug in protocols (hop-by-hop authorization mechanisms), and the server can reason about the authorizations supported by each protocol. If the server does not support a given mechanism natively, it can still reason about authorization information from a protocol-translating gateway that passes through end-to-end authorization information using my system.

Figure 8.3 illustrates how we reason about authorization at different network scales. When Alice is at the same machine as the resource server (a), the server trusts the kernel to reliably report her identity, from which it derives her authority. Objects in the logic encode the statements “the kernel says this interprocess communication (IPC) channel speaks for Alice” and “the server says the kernel speaks the truth.” The server uses the authorization library to conclude that requests arriving on the IPC channel represent Alice's wishes.

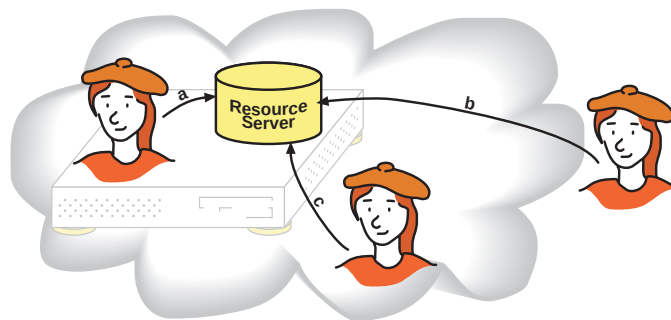


Figure 8.3: When Alice connects to a resource server from the same machine, the server may trust the kernel to correctly identify her (a). When Alice connects from elsewhere on the Internet, the server may require strong encryption to verify her authority (b). When Alice connects from inside a department firewall, the server may accept IP source addresses as sufficient proof of her authority (c).

In the wide-area network case (b), a standard encryption-based authentication protocol

opcodes	operand stack
(sequence	
(public-key K_C)	K_C
(do hash sha1)	$K_C \Rightarrow H_{K_C}, H_{K_C} \Rightarrow K_C$
(do pop)	$H_{K_C} \Rightarrow K_C$
(do name-monotonicity N)	$H_{K_C} \cdot N \Rightarrow K_C \cdot N$
(cert	} $K_S \Rightarrow H_{K_C} \cdot N, H_{K_C} \cdot N \Rightarrow K_C \cdot N$
(issuer $H_{K_C} \cdot N$)	
(subject K_S))	
(do transitive 0 1)	$K_S \Rightarrow K_C \cdot N$
(cert	} $H_D \Rightarrow K_S, K_S \Rightarrow K_C \cdot N$
(issuer K_S)	
(subject H_D))	
(do transitive 0 1)	$H_D \Rightarrow K_C \cdot N$
)	

Figure 9.1: A hypothetical SPKI sequence. A linear sequence of instructions (left) advances a stack machine (right) to arrive at an intended conclusion ($H_D \Rightarrow K_C \cdot N$). Since SPKI does not define the sequence operations, I have created imaginary stack-manipulation operations that introduce statements and apply theorems of the logic.

9.4 The prover

A **Prover** object helps Snowflake applications collect and create proofs. It has three tasks: it collects delegations, caches proofs, and constructs new delegations.

A user's application collects delegations from other users. Gateways collect delegations directly from client applications. Both sorts of applications use a **Prover** to maintain their collected delegations in a graph where nodes represent principals and edges represent a proof of authority  one principal to the next (see Figure 9.3). The **Prover** traverses the graph breadth first to find proofs of delegation  the application requires. For example, if the **Prover** must prove that a channel K_{CH} speaks for a server S , it works backwards from the node S to find the proof that $A \xRightarrow{V \cap X} S$. A is *final*, meaning that the **Prover** can make statements as A ; therefore, **Prover** simply issues a delegation $K_{CH} \Rightarrow A$ to complete the proof.

When the **Prover** receives a delegation that is actually a proof involving several steps, the **Prover** “digests” the proof into its component parts for storage in the graph. Whenever it receives or computes a derived proof composed of smaller components, the **Prover**  stores a shortcut edge  in the graph  to hold the proof. These shortcuts form a cache that eliminates most deep traversals of the graph.

When an application controls one or more principals (e.g., by holding the corresponding private key or capability), its **Prover** can store a closure in its graph for the controlled principal. When desired, the **Prover** can not only find existing proofs, but complete new proofs by finding an existing chain of delegations from the controlled principal to the re-

opcodes	operand stack
(sequence	
(public-key K_C)	K_C
(do hash sha1)	$K_C \Rightarrow H_{K_C}, H_{K_C} \Rightarrow K_C$
(do pop)	$H_{K_C} \Rightarrow K_C$
(do name-monotonicity N)	$H_{K_C} \cdot N \Rightarrow K_C \cdot N$
(cert	} $K_S \Rightarrow H_{K_C} \cdot N, H_{K_C} \cdot N \Rightarrow K_C \cdot N$
(issuer $H_{K_C} \cdot N$)	
(subject K_S))	
(do transitive 0 1)	$K_S \Rightarrow K_C \cdot N$
(cert	} $H_D \Rightarrow K_S, K_S \Rightarrow K_C \cdot N$
(issuer K_S)	
(subject H_D))	
(do transitive 0 1)	$H_D \Rightarrow K_C \cdot N$
)	

Figure 9.1: A hypothetical SPKI sequence. A linear sequence of instructions (left) advances a stack machine (right) to arrive at an intended conclusion ($H_D \Rightarrow K_C \cdot N$). Since SPKI does not define the sequence operations, I have created imaginary stack-manipulation operations that introduce statements and apply theorems of the logic.

9.4 The prover

A **Prover** object helps Snowflake applications collect and create proofs. It has three tasks: it collects delegations, caches proofs, and constructs new delegations.

A user’s application collects delegations from other users. Gateways collect delegations directly from client applications. Both sorts of applications use a **Prover** to maintain their collected delegations in a graph where nodes represent principals and edges represent a proof of authority from one principal to the next (see Figure 9.3). The **Prover** traverses the graph breadth first to find proofs of delegation required by the application. For example, if the **Prover** must prove that a channel K_{CH} speaks for a server S , it works backwards from the node S to find the proof that $A \stackrel{V \cap X}{\Rightarrow} S$. A is *final*, meaning that the **Prover** can make statements as A ; therefore, **Prover** simply issues a delegation $K_{CH} \Rightarrow A$ to complete the proof.

When the **Prover** receives a delegation that is actually a proof involving several steps, the **Prover** “digests” the proof into its component parts for storage in the graph. Whenever it receives or computes a derived proof composed of smaller components, the **Prover** adds a shortcut edge to the graph to represent the proof. These shortcuts form a cache that eliminates most deep traversals of the graph.

When an application controls one or more principals (e.g., by holding the corresponding private key or capability), its **Prover** can store a closure in its graph for the controlled principal. When desired, the **Prover** can not only find existing proofs, but complete new proofs by finding an existing chain of delegations from the controlled principal to the re-

my Java environment, I treat the JVM and a few system classes as the trusted host, and bypass encryption when connecting to a server in the same JVM.

In the local case, the `ssh` channel is replaced with a Java “IPC” pipe implemented without any operating system IPC services, and the public keys corresponding to the channel endpoints (K_1 and K_2) are swapped directly. Because it was involved in constructing the key pairs and the keys are stored in immutable objects, the trusted system class knows whether a client holds the private key corresponding to a given public key. Hence when a client is colocated in the same JVM with the server, there is no encryption or system-call overhead associated with the channel, only RMI serialization costs.

10.3 Signed requests

Not all applications can assume that my `ssh`-enhanced version of RMI is available as an RPC mechanism. Indeed, the most visible RPC mechanism on the Internet is HTTP. To facilitate applications that use HTTP, I created a Snowflake version of the HTTP authorization protocol.

HTTP defines a simple, extensible challenge-response authorization mechanism [FHBH⁺99] diagrammed in Figure 10.3.² The client sends an HTTP request to the server. The server replies with a “401 Unauthorized” response, including a `WWW-Authenticate` header describing the method and other parameters of the required authorization. The client resends its request, this time including an `Authorization` header. If the `Authorization` satisfies the server’s challenge, the server honors the request and replies with the return value of the operation. Otherwise, the server returns a “403 Forbidden” response to indicate the authorization failure.

HTTP defines two standard authorization methods. In Basic Authentication, the client’s `Authorization` header includes a password in cleartext. In Digest Authentication, the server’s `WWW-Authenticate` challenge includes a nonce, and the client’s `Authorization` header consists of a secure hash of the nonce and the user’s password. Both methods *authenticate* the client as the holder of a secret password, and leave *authorization* to an ACL at the server.

In my new method, called Snowflake Authorization, the parameters embedded in the server’s `WWW-Authenticate` challenge are the issuer that the client needs to speak for and the minimum restriction set that the delegation must allow. The `Authorization` header in the client’s second request simply includes a Snowflake proof that the request speaks for the required issuer regarding the specified restriction set. The subject of the proof is a hash of the request, less the `Authorization` header. Figures 10.4 and 10.5 show examples of these messages.

10.3.1 Signed request optimization

As described, the signed request protocol requires an expensive public-key signature on every request. Since it is common that a client and server interact in more than one

²The protocol specification itself uses the terms “authentication” and “authorization” interchangeably; when not quoting the specification, I use the term authorization since it best describes my goal. ~~We~~ reduce the authentication problem to authorization in Section 10.3.2.

my Java environment, I treat the JVM and a few system classes as the trusted host, and bypass encryption when connecting to a server in the same JVM.

In the local case, the `ssh` channel is replaced with a Java “IPC” pipe implemented without any operating system IPC services, and the public keys corresponding to the channel endpoints (K_1 and K_2) are swapped directly. Because it was involved in constructing the key pairs and the keys are stored in immutable objects, the trusted system class knows whether a client holds the private key corresponding to a given public key. Hence when a client is colocated in the same JVM with the server, there is no encryption or system-call overhead associated with the channel, only RMI serialization costs.

10.3 Signed requests

Not all applications can assume that my `ssh`-enhanced version of RMI is available as an RPC mechanism. Indeed, the most visible RPC mechanism on the Internet is HTTP. To facilitate applications that use HTTP, I created a Snowflake version of the HTTP authorization protocol.

HTTP defines a simple, extensible challenge-response authorization mechanism [FHBH⁺99] diagrammed in Figure 10.3.² The client sends an HTTP request to the server. The server replies with a “401 Unauthorized” response, including a `WWW-Authenticate` header describing the method and other parameters of the required authorization. The client resends its request, this time including an `Authorization` header. If the `Authorization` satisfies the server’s challenge, the server honors the request and replies with the return value of the operation. Otherwise, the server returns a “403 Forbidden” response to indicate the authorization failure.

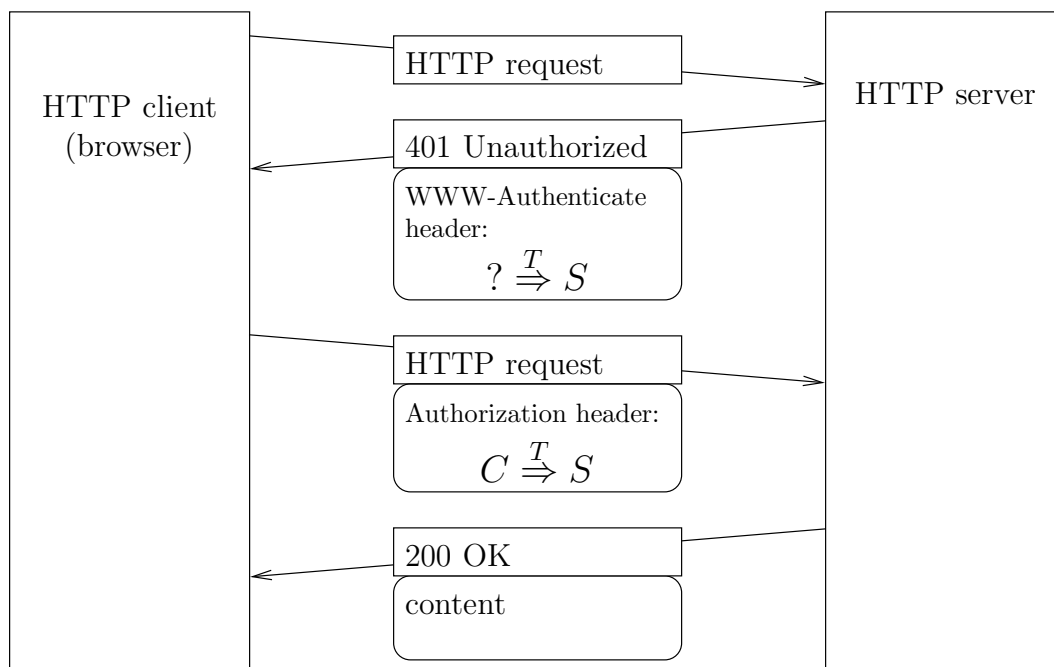
HTTP defines two standard authorization methods. In Basic Authentication, the client’s `Authorization` header includes a password in cleartext. In Digest Authentication, the server’s `WWW-Authenticate` challenge includes a nonce, and the client’s `Authorization` header consists of a secure hash of the nonce and the user’s password. Both methods *authenticate* the client as the holder of a secret password, and leave *authorization* to an ACL at the server.

In my new method, called Snowflake Authorization, the parameters embedded in the server’s `WWW-Authenticate` challenge are the issuer that the client needs to speak for and the minimum restriction set that the delegation must allow. The `Authorization` header in the client’s second request simply includes a Snowflake proof that the request speaks for the required issuer regarding the specified restriction set. The subject of the proof is a hash of the request, less the `Authorization` header. Figures 10.4 and 10.5 show examples of these messages.

10.3.1 Signed request optimization

As described, the signed request protocol requires an expensive public-key signature on every request. Since it is common that a client and server interact in more than one

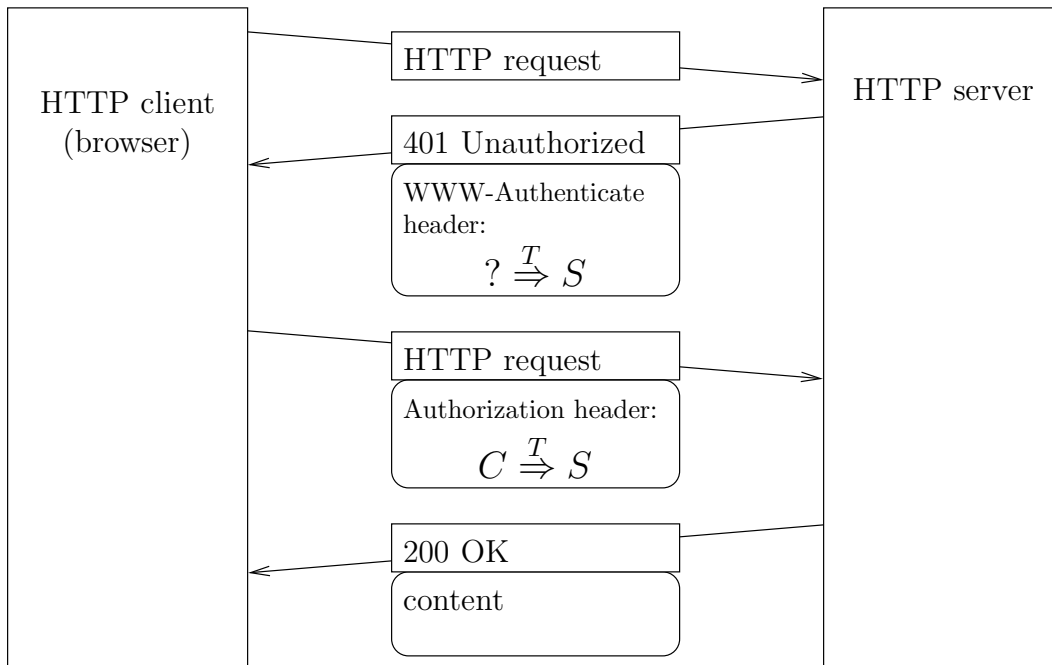
²The protocol specification itself uses the terms “authentication” and “authorization” interchangeably; when not quoting the specification, I use the term authorization since it best describes my goal. We reduce the authentication problem to authorization in Section 10.3.2.

Figure 10.3: *The HTTP authorization protocol.*

request, I wish to amortize the signature operation over many requests. One approach is to send multiple requests over a stream that guarantees integrity; among other things, this is a property SSL can provide.

I chose to implement an alternative protocol with a different set of performance and security characteristics. When sending in a signed request, a client may specify a public key under which it wishes to receive an encrypted secret message authentication code (MAC). If the server agrees, it generates a random authenticator, encrypts it with the specified public key, and returns it to the client. Then the client signs a certificate $H_{MAC} \Rightarrow K_C$ delegating its authority to the secret MAC, named by its hash so that the certificate need not remain secret. For future requests, instead of signing the specific request, the client need only hash the request together with the MAC. This hash shows that the text of the request speaks for the MAC: $H_{request} \Rightarrow H_{MAC}$.

The protocol is fast because secure hashing has replaced expensive public-key signatures. It requires that the client authenticate the server, or else a man-in-the-middle can supply the client with a bogus MAC. The client will also likely want to make the statement $H_{MAC} \xRightarrow{V} K_C$ with a short validity interval V to limit the trust extended to the MAC. This protocol highlights one of the attractive properties of my unified authorization protocol: it can make any authorization trust relationship explicit, auditable, and visible end-to-end. The final server receiving a request can see that my home-brew MAC protocol was involved in the transaction.

Figure 10.3: *The HTTP authorization protocol.*

request, I wish to amortize the signature operation over many requests. One approach is to send multiple requests over a stream that guarantees integrity; among other things, this is a property SSL can provide.

I chose to implement an alternative protocol with a different set of performance and security characteristics. When sending in a signed request, a client may specify a public key under which it wishes to receive an encrypted secret message authentication code (MAC). If the server agrees, it generates a random authenticator, encrypts it with the specified public key, and returns it to the client. Then the client signs a certificate $H_{MAC} \Rightarrow K_C$ delegating its authority to the secret MAC, named by its hash so that the certificate need not remain secret. For future requests, instead of signing the specific request, the client need only hash the request together with the MAC. This hash shows that the text of the request speaks for the MAC: $H_{request} \Rightarrow H_{MAC}$.

The **Snowflake HTTP authorization** protocol is fast because secure hashing has replaced expensive public-key signatures. It requires that the client authenticate the server, or else a man-in-the-middle can supply the client with a bogus MAC. The client will also likely want to make the statement $H_{MAC} \xRightarrow{V} K_C$ with a short validity interval V to limit the trust extended to the MAC. This protocol highlights one of the attractive properties of my unified authorization protocol: it can make any authorization trust relationship explicit, auditable, and **visible** end-to-end. The final server receiving a request can see that my home-brew MAC protocol was involved in the transaction.

```

HTTP/1.0 401 UNAUTHORIZED
Content-Type: text/html
MIME-Version: 1.0
Server: MortBay-Jetty-2.3.3
Date: Sat, 08 Apr 2000 15:18:47 GMT
WWW-Authenticate: SnowflakeProof Authorize-Client
Sf-ServiceIssuer: (hash md5 |ehtQYd4EpQX0a/ON6Smesg==|)
Sf-MinimumTag: (tag
  (web (method GET)
    (service |Sm9uJ3MgUHHvdGVjdGVkIFNlcnZpY2U=|)
    (resourcePath "")))
Connection: close

```

Figure 10.4: An HTTP authorization challenge message from a Snowflake server. It indicates the method, the required resource issuer, and the minimum restriction of a delegation that must be proven.

10.3.2 Server authorization

Often a client also wants to verify that it is communicating with the “right” server. The notion of “right” can be as simple as the server speaking for the client’s idea of a well-known name like `www.dartmouth.edu`, but in general the real question is still one of authorization: Does this server have the right to claim authority about Dartmouth’s course list? Does that server have authority to receive my e-mail?

I defined a simple protocol by which an HTTP server can communicate its authority to serve a document to a client. The server P_S simply constructs a proof that the document speaks for the server’s name for the document $P_S \cdot N_{document}$. The server returns the proof in the `Sf-DocSpeaksForServerName` header that precedes the document. The client queries its `Prover` to see who the server speaks for. Perhaps the document was returned as a result of a user clicking a link with embedded authorization information, in which case the client expects to find a proof that the server speaks with that authority; or perhaps the client just wishes to find any locally-trusted name for the document to report to the user³. Notice that although the client and server are in opposite roles in the trust relationship, the burden of proof is still at the client: the protocol design assumes that clients have more resources, and that the client’s history of requests may provide hints for efficient proof.

My server authorization protocol leaves a lot to be desired; for example, because it does not attest to the authority of the complete transaction including headers, it cannot help with the man-in-the-middle attack described in Section 10.3.1. An alternate tactic would have the server deliver the document over a stream with integrity (one of the features of SSL), and delegate its authority to the stream. End-to-end authorization still occurs: the client arrives at a proof showing the participation of either the document hash (in my protocol) or the stream.

³My current implementation only supports the latter naming operation.


```

HTTP/1.0 401 UNAUTHORIZED
Content-Type: text/html
MIME-Version: 1.0
Server: MortBay-Jetty-2.3.3
Date: Sat, 08 Apr 2000 15:18:47 GMT
WWW-Authenticate: SnowflakeProof Authorize-Client
Sf-ServiceIssuer: (hash md5 |ehtQYd4EpQX0a/ON6Smesg==|)
Sf-MinimumTag: (tag
  (web (method GET)
    (service |Sm9uJ3MgUHVJvdGVjdGVkIFNlcnZpY2U=|)
    (resourcePath "")))
Connection: close

```

Figure 10.4: An HTTP authorization challenge message from a Snowflake server. It indicates the method, the required resource issuer, and the minimum restriction of a delegation that must be proven.

10.3.2 Server authorization

Often a client also wants to verify that it is communicating with the “right” server. The notion of “right” can be as simple as the server speaking for the client’s idea of a well-known name like `www.dartmouth.edu`, but in general the real question is still one of authorization: Does this server have the right to claim authority about Dartmouth’s course list? Does that server have authority to receive my e-mail?

I defined a simple protocol by which an HTTP server can communicate its authority to serve a document to a client. The server P_S simply constructs a proof that the document speaks for the server’s name for the document $P_S \cdot N_{document}$. The server returns the proof in the `Sf-DocSpeaksForServerName` header that precedes the document. The client queries its `Prover` to see who the server speaks for. Perhaps the document was returned as a result of a user clicking a link with embedded authorization information, in which case the client expects to find a proof that the server speaks with that authority; or perhaps the client just wishes to find any locally-trusted name for the document to report to the user³. Notice that although the client and server are in opposite roles in the trust relationship, the burden of proof is still at the client: the protocol design assumes that clients have more resources, and that the client’s history of requests may provide hints for efficient proof.

The Snowflake HTTP server authorization protocol leaves much to be desired; for example, because it does not attest to the authority of the complete transaction including headers, it cannot help with the man-in-the-middle attack described in Section 10.3.1. An alternate tactic would have the server deliver the document over a stream with integrity (one of the features of SSL), and delegate its authority to the stream. End-to-end authorization still occurs: the client arrives at a proof showing the participation of either the document hash (in my protocol) or the stream.

³My current implementation only supports the latter naming operation.

Chapter 11

Applications

I built three applications to demonstrate the Snowflake architecture for sharing.

11.1 Protected web server

The first application is simply a protected web file server that uses Snowflake's sharing architecture. One user initially controls the file server (by specifying the hash of his public key when starting up the server); he may delegate permission to read subtrees or individual files from the server to others using the mechanisms described in Section 10.3.

The server implementation extends the `ProtectedServlet` class of Section 10.3.3. This `FileServlet` class overrides the `getResourceTag()` method of its superclass to map URLs into Snowflake tags that describe the request. The superclass ensures that the client is authorized to make the request, then the `FileServlet`'s `servePage()` method maps the request URL to a filesystem path, reads the file, and returns its contents.

11.2 Protected database

The second application attaches Snowflake security to a relational email database. The original database server accepts insert, update, and select requests as RMI invocations on a `Remote Database` object, and returns the results of the query as serialized objects from the database. Adapting the application to Snowflake required only minimal changes. I modified the database instance constructor to use a `SshSocketFactory` so that all connections to the object use my `ssh` secure channels. Then, I prepended each implementation of a method in the remote interface with a call to the `checkAuth()` method. The database clients required only a modification to their initialization code to install an `SSHContext` and a `Prover`.

In my case, the database contains only mail owned by an individual. Perhaps, however, a site-wide email database might be controlled by a database administrator. That administrator delegates to each user restricted authority to manipulate his part of the database. The resource each user sees, his slice of the database records, is a first-class resource that he can treat as though it was a complete database.

Chapter 11

Applications

I built three applications to demonstrate the Snowflake architecture for sharing.

11.1 Protected web server

The first application is simply a protected web file server that uses Snowflake's sharing architecture. One user establishes control over the file server by specifying the hash of his public key when starting up the server; he may delegate to others permission to read subtrees or individual files from the server using the mechanisms described in Section 10.3.

The server implementation extends the `ProtectedServlet` class of Section 10.3.3. This `FileServlet` class overrides the `getResourceTag()` method of its superclass to map URLs into Snowflake tags that describe the request. The superclass ensures that the client is authorized to make the request, then the `FileServlet`'s `servePage()` method maps the request URL to a filesystem path, reads the file, and returns its contents.

11.2 Protected database

The second application attaches Snowflake security to a relational email database. The original database server accepts insert, update, and select requests as RMI invocations on a `Remote Database` object, and returns the results of the query as serialized objects from the database. Adapting the application to Snowflake required only minimal changes. I modified the database instance constructor to use a `SshSocketFactory` so that all connections to the object use my `ssh` secure channels. Then, I prepended each implementation of a method in the remote interface with a call to the `checkAuth()` method. The database clients required only a modification to their initialization code to install an `SSHContext` and a `Prover`.

In my case, the database contains only mail owned by an individual. Perhaps, however, a site-wide email database might be controlled by a database administrator. That administrator delegates to each user restricted authority to manipulate his part of the database. The resource each user sees, his slice of the database records, is a first-class resource that he can treat as though it was a complete database.

to discount startup costs such as loading class files. Where they are within three orders of magnitude of the y -intercept quantity, I report the standard deviation of errors (σ), from which one can infer the amount of noise over the linear process. When the quality of fit $R^2 < 0.99$, I report it as well; experiments with small R^2 have shallow slopes, and hence the linear model does not contribute much information beyond the constant line at the mean of the y observations. I computed 95% confidence intervals on the linear regression parameters and found them always to be at least three orders of magnitude smaller than the quantity they bound. I round all values to two significant figures. I visually inspected each scatter plot to ensure that the linear models are reasonable; the plots appear in Appendix F.

Sometimes the ten trials of an experiment with a single collection of values exhibited a significant coefficient of variation (C.V.). I checked the C.V.s for each individual experiment, and re-ran those with C.V.s greater than 0.1. My understanding of the system and my inspection of the data give me confidence that by doing so, I have eliminated noise due to asynchronous events such as `cron` jobs or external network traffic, rather than rubbing out observations of rare events in my experimental system.

12.2 RMI authorization with Snowflake

In this section, I quantify my implementation of Snowflake authorization over Java remote method invocation as described in Section 10.1. Figure 12.1 summarizes the overhead my prototype adds to RMI.

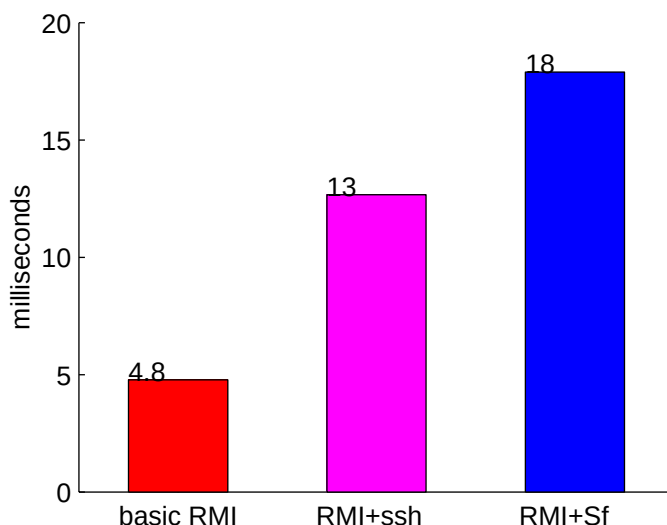


Figure 12.1: *The cost of introducing Snowflake authorization to RMI. This graph summarizes Table 12.2. A basic RMI call costs 4.8 ms. Securing the channel with `ssh` introduces significant overhead. Mapping the request into Snowflake and verifying the client’s authority adds another 5 ms.*

The test operation is a Remote object that returns the contents of a file. According to Table 12.2, the setup time for this operation in basic RMI is 4.8 ms, and the compute-

to discount startup costs such as loading class files. Where they are within three orders of magnitude of the y -intercept quantity, I report the standard deviation of errors (σ), from which one can infer the amount of noise over the linear process. When the quality of fit $R^2 < 0.99$, I report it as well; experiments with small R^2 have shallow slopes, and hence the linear model does not contribute much information beyond the constant line at the mean of the y observations. I computed 95% confidence intervals on the linear regression parameters and found them always to be at least three orders of magnitude smaller than the quantity they bound. I round all values to two significant figures. I visually inspected each scatter plot to ensure that the linear models are reasonable; the plots appear in Appendix F.

Sometimes the ten trials of an experiment with a single collection of values exhibited a significant coefficient of variation (C.V.). I checked the C.V.s for each individual experiment, and re-ran those with C.V.s greater than 0.1. My understanding of the system and my inspection of the data give me confidence that by doing so, I have eliminated noise due to asynchronous events such as `cron` jobs or external network traffic, rather than rubbing out observations of rare events in my experimental system.

12.2 RMI authorization with Snowflake

In this section, I quantify my implementation of Snowflake authorization over Java remote method invocation as described in Section 10.1. Figure 12.1 summarizes the overhead my prototype adds to RMI.

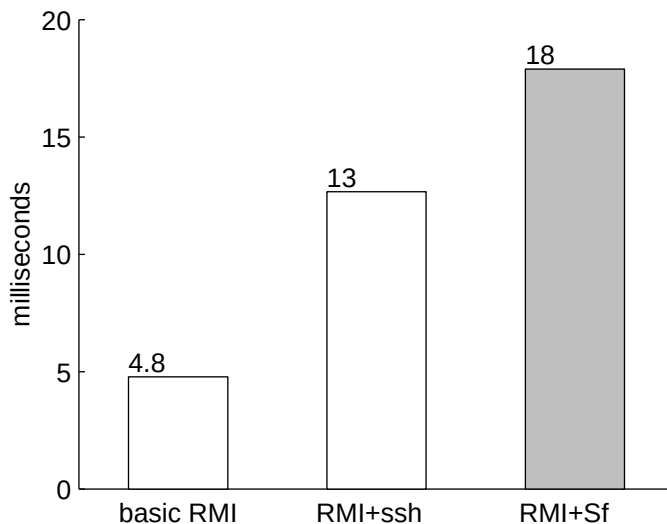


Figure 12.1: *The cost of introducing Snowflake authorization to RMI. This graph summarizes Table 12.2. A basic RMI call costs 4.8 ms. Securing the channel with `ssh` introduces significant overhead. Mapping the request into Snowflake and verifying the client’s authority adds another 5 ms.*

The test operation is a Remote object that returns the contents of a file. According to Table 12.2, the setup time for this operation in basic RMI is 4.8 ms, and the compute-

bound data-copy overhead is 990 ms/MB. Performing the same operation over an open `ssh` channel increases the fixed cost to 13 ms, and the data copy cost to 6100 ms/MB. Adding the Snowflake protocol brings the total setup time to 18 ms: the extra work is the server’s `checkAuth()` call, which retrieves the caller’s public key, finds a cached proof for that subject, and sees that the proof has already been verified.

protocol	request ms	copy ms/MB	σ ms	R^2
RMI	4.8	990	5.7	0.96
remote	6.9	1100	1.7	
ssh alone	13	6100	4.4	
remote	13	3900	6.6	
Snowflake	18	6100	6.7	
remote	16	3900	5.3	

Table 12.2: A setup and bandwidth *experiment* for RMI. Results from experiments where client and server are on separate hosts appear on lines marked *remote*.

Table 12.2 shows results from a warm connection. The gap between 4.8 ms and 13 ms reflects the extra overhead of `ssh` stream processing on the few packets that carry an individual request. The next jump up to 18 ms includes the costs of looking up the client’s identity, mapping the request into a SPKI tag, and verifying that a cached proof shows the client’s authority for the request. The bold values measure a connection between two processes on the same host, and the values labeled “remote” measure a connection between two processes on hosts separated by a shared 10 Mbps Ethernet segment. For protocols that do not encrypt the byte stream, the network bandwidth dominates the copy cost. For protocols that do encrypt, CPU time dominates the copy cost; hence copy costs are lower in the remote case than the local, since two CPUs are available.

Table 12.3 presents results from cold connection caches. It costs 470 ms to establish a new connection, reflecting the public-key operation the client performs to delegate its authority to the channel. When the client caches the delegation but the server forgets its copy after each use, the server spends about 190 ms parsing and verifying the proof from the client. If the server caches the proof, the connection achieves approximately the 20 ms per request that appears in Table 12.2 (σ accounts for the variation from 18 ms). The Snowflake overhead is higher than we might expect of an optimized implementation; I discuss reasons for this overhead in Section 12.5.

12.3 HTTP authorization with Snowflake

In this section, I quantify my implementation of Snowflake authorization over the HTTP protocol as described in Section 10.3. Figure 12.2 visually summarizes the overhead my prototype adds to HTTP, and Figure 12.1 relates the performance of the prototype to SSL.

bound data-copy overhead is 990 ms/MB. Performing the same operation over an open `ssh` channel increases the fixed cost to 13 ms, and the data copy cost to 6100 ms/MB. Adding the Snowflake protocol brings the total setup time to 18 ms: the extra work is the server’s `checkAuth()` call, which retrieves the caller’s public key, finds a cached proof for that subject, and sees that the proof has already been verified.

protocol	request ms	copy ms/MB	σ ms	R^2
RMI	4.8	990	5.7	0.96
remote	6.9	1100	1.7	
ssh alone	13	6100	4.4	
remote	13	3900	6.6	
Snowflake	18	6100	6.7	
remote	16	3900	5.3	

Table 12.2: A setup and bandwidth *experiment* for RMI. Results from experiments where client and server are on separate hosts appear on lines marked *remote*.

Table 12.2 shows results from a warm connection. The gap between 4.8 ms and 13 ms reflects the extra overhead of `ssh` stream processing on the few packets that carry an individual request. The next jump up to 18 ms includes the costs of looking up the client’s identity, mapping the request into a SPKI tag, and verifying that a cached proof shows the client’s authority for the request. The bold values measure a connection between two processes on the same host, and the values labeled “remote” measure a connection between two processes on hosts separated by a shared 10 Mbps Ethernet segment. For protocols that do not encrypt the byte stream, the network bandwidth dominates the copy cost. For protocols that do encrypt, CPU time dominates the copy cost; hence copy costs are lower in the remote case than the local, since two CPUs are available.

Table 12.3 presents results from cold connection caches. It costs 470 ms to establish a new connection, reflecting the public-key operation the client performs to delegate its authority to the channel. When the client caches the delegation but I make the server forget its copy after each use, we learn that the server spends 190 ms parsing and verifying the proof from the client. If the server caches the proof, the connection achieves approximately the 20 ms per request that appears in Table 12.2 (σ accounts for the variation from 18 ms). The Snowflake overhead is higher than we might expect of an optimized implementation; I discuss reasons for this overhead in Section 12.5.

12.3 HTTP authorization with Snowflake

In this section, I quantify my implementation of Snowflake authorization over the HTTP protocol as described in Section 10.3. Figure 12.2 visually summarizes the overhead my prototype adds to HTTP, and Figure 12.3 relates the performance of the prototype to SSL.

proofs cached	request ms	copy ms/MB	σ ms
none	470	6200	14
remote	410	3800	5.1
client	190	6400	19
remote	140	3800	5.5
server	20	6100	7.1
remote	17	3800	4.4

Table 12.3: *Disabling caches reveals the cost of proof generation, transmission and verification.*

client	server	request ms	copy ms/MB	σ ms	R^2
C	Apache	4.6	61	0.27	0.98
	remote	4.8	960	0.56	
Java	Apache	11	93	0.92	0.89
	remote	10	950	0.51	
	Java	17	180	1.0	0.96
	remote	17	980	0.70	
	Jetty	25	240	2.3	0.90
	remote	24	1000	6.1	0.96

Table 12.4: *Performance baselines for HTTP 1.0.*

12.3.1 Client baseline

First, I measured the time it takes to access an Apache web server with a C client and with my Java client using an HTTP stream-processing package (part of the Jetty web server). The results are presented in Table 12.4. With the C client, a connection costs about 4.6 ms, each megabyte of data transferred locally costs about 61 ms/MB more, and as one might expect, each megabyte transferred over a shared 10 Mbps Ethernet segment costs about 960 ms/MB. From my Java client, the connection costs 11 ms plus 93 ms/MB locally and still around 950 ms/MB remote. This experiment establishes the baseline cost of my Java timing client that I use in all of the further tests.

12.3.2 Server baseline

Next, as shown in the second half of Table 12.4, I compared the Apache server with both a trivial server written in Java and the Jetty web server, a full-fledged Java web server with rich support for the features of HTTP streams. Taking the jump to Java adds about 6 ms to the cost of a connection, and doubles the per-MB cost. These costs likely reflect the fact that while Apache uses C system call interfaces directly, Java’s I/O interfaces are overhead-incurring wrappers linked to the same C glue routines.

proofs cached	request ms	copy ms/MB	σ ms
none	470	6200	14
remote	410	3800	5.1
client	190	6400	19
remote	140	3800	5.5
server	20	6100	7.1
remote	17	3800	4.4

Table 12.3: *Disabling caches reveals the cost of proof generation, transmission and verification.*

client	server	request ms	copy ms/MB	σ ms	R^2
C	Apache	4.6	61	0.27	0.98
	remote	4.8	960	0.56	
Java	Apache	11	93	0.92	0.89
	remote	10	950	0.51	
	Java	17	180	1.0	0.96
	remote	17	980	0.70	
	Jetty	25	240	2.3	0.90
	remote	24	1000	6.1	0.96

Table 12.4: *Performance baselines for HTTP 1.0.*

12.3.1 Client baseline

First, I measured the time it takes to access an Apache web server with a C client and with my Java client using an HTTP stream-processing package (part of the Jetty web server). Table 12.4 presents the results. With the C client, a connection costs about 4.6 ms, each megabyte of data transferred locally costs about 61 ms/MB more, and as one might expect, each megabyte transferred over a shared 10 Mbps Ethernet segment costs about 960 ms/MB. From my Java client, the connection costs 11 ms plus 93 ms/MB locally and still around 950 ms/MB remote. This experiment establishes the baseline cost of my Java timing client that I use in all of the further tests.

12.3.2 Server baseline

Next, as shown in the second half of Table 12.4, I compare the Apache server with both a trivial server written in Java and the Jetty web server, a full-fledged Java web server with rich support for the features of HTTP streams. Taking the jump to Java adds about 6 ms to the cost of a connection, and doubles the per-MB cost. These costs likely reflect the fact that while Apache uses C system call interfaces directly, Java's I/O interfaces are overhead-incurring wrappers linked to the same C glue routines.

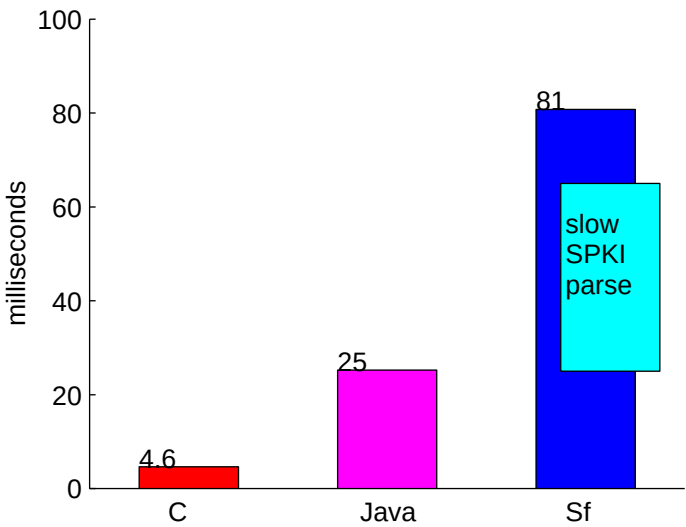


Figure 12.2: The cost of introducing Snowflake authorization to HTTP. This graph summarizes Tables 12.4 and 12.6. A trivial C client accessing an Apache server takes 4.6 ms. Replacing the client and server with convenient but inefficient Java packages brings the baseline for HTTP to 25 ms. Most of Snowflake’s overhead reflects the use of inefficient SPKI libraries, shown graphically as an inset box; for details, see Section 12.5.3.

My implementation is based on the convenient services and clean factorization provided by the Jetty web server. Using Jetty adds another 8 ms, and increases the data transfer cost another 33%. These costs reflect Jetty’s fairly complete request parser and Jetty’s stream filters needed to support the various HTTP models for content length and chunking. The Jetty implementation could certainly stand some optimization, but I present the Jetty costs as a baseline since my implementation is based on Jetty.

Table 12.5 shows the results of an experiment that factors the cost of an HTTP request from the cost of establishing the underlying TCP connection. With HTTP/1.1, I issue multiple requests per connection, and infer the setup and per-request cost of an HTTP connection. The costs do not correspond exactly with those shown in Table 12.4, since

server	connect ms	request ms/req	 ms
Apache	4.3	4.6	1.6
remote	3.6	4.5	0.80
Jetty	10	22	2.0
remote	16	28	9.9

Table 12.5: A setup and per-request experiment that reveals the baseline cost of a single HTTP/1.1 request. By varying the number of requests per connection, a linear regression provides the setup cost of a connection and the cost of each request thereafter.

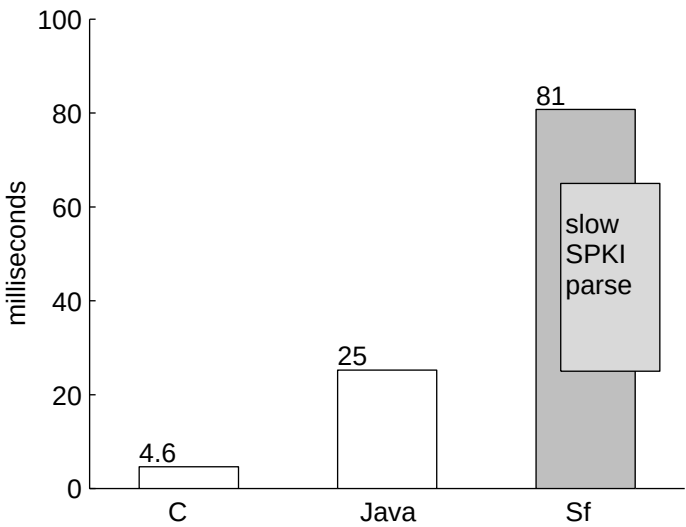


Figure 12.2: *The cost of introducing Snowflake authorization to HTTP. This graph summarizes Tables 12.4 and 12.6. A trivial C client accessing an Apache server takes 4.6 ms. Replacing the client and server with convenient but inefficient Java packages brings the baseline for HTTP to 25 ms. Most of Snowflake’s overhead reflects the use of inefficient SPKI libraries, shown graphically as an inset box; for details, see Section 12.5.3.*

My implementation is based on the convenient services and clean factorization provided by the Jetty web server. Using Jetty adds another 8 ms, and increases the data transfer cost another 33%. These costs reflect Jetty’s fairly complete request parser and Jetty’s stream filters needed to support the various HTTP models for content length and chunking. The Jetty implementation could certainly stand some optimization, but I present the Jetty costs as a baseline since my implementation is based on Jetty.

Table 12.5 shows the results of an experiment that factors the cost of an HTTP request from the cost of establishing the underlying TCP connection. With HTTP/1.1, I issue multiple requests per connection, and infer the setup and per-request cost of an HTTP connection. The costs do not correspond exactly with those shown in Table 12.4, since

server	connect ms	request ms/req	σ ms
Apache	4.3	4.6	1.6
remote	3.6	4.5	0.80
Jetty	10	22	2.0
remote	16	28	9.9

Table 12.5: *A setup and per-request experiment that reveals the baseline cost of a single HTTP/1.1 request. By varying the number of requests per connection, a linear regression provides the setup cost of a connection and the cost of each request thereafter.*

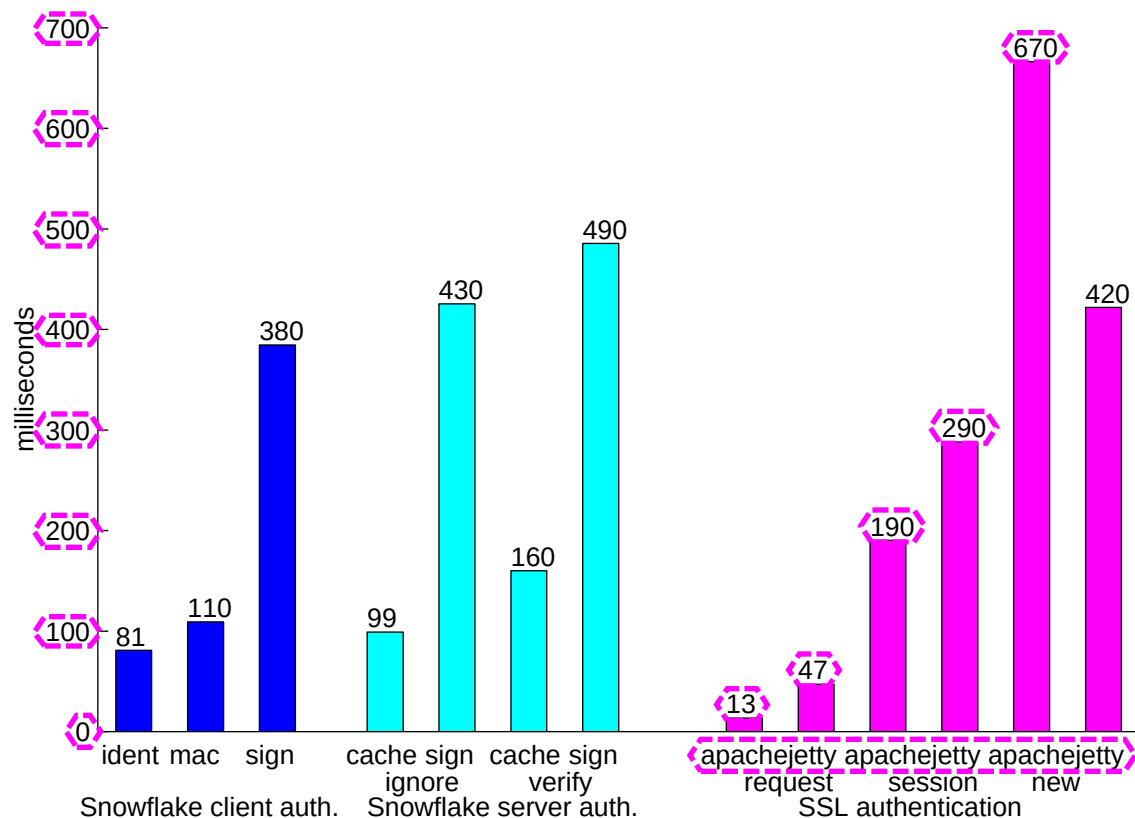


Figure 12.3: This graph compares Snowflake client authorization, server (document) authentication, and standard SSL authentication. The left group draws from Table 12.6, the center group draws from Table 12.7, and the right group draws from Tables 12.8 and 12.9.

client and server are executing HTTP version 1.1 rather than 1.0.

12.3.3 Network baseline

I ran the same tests on a shared segment of 10 Mbps Ethernet, with no intervening router. Not surprisingly, the per-request costs in the remote case are very similar to those in the local case, since a connection setup only incurs a few sub-microsecond latencies. Per-megabyte costs snap to about one second per megabyte, reflecting the 10 Mbps bandwidth of the cable.

12.3.4 Snowflake costs

Now that I have established a baseline, what are the costs of my implementation of Snowflake services over HTTP? As shown in Table 12.6, a fresh connection costs around 380 ms to establish, including making a public-key signature of the outbound request. Replaying the same request costs only about 81 ms, since the **Prover** has the required proof in cache. Using the MAC optimization from Section 10.3.1 brings the cost for authorizing a new

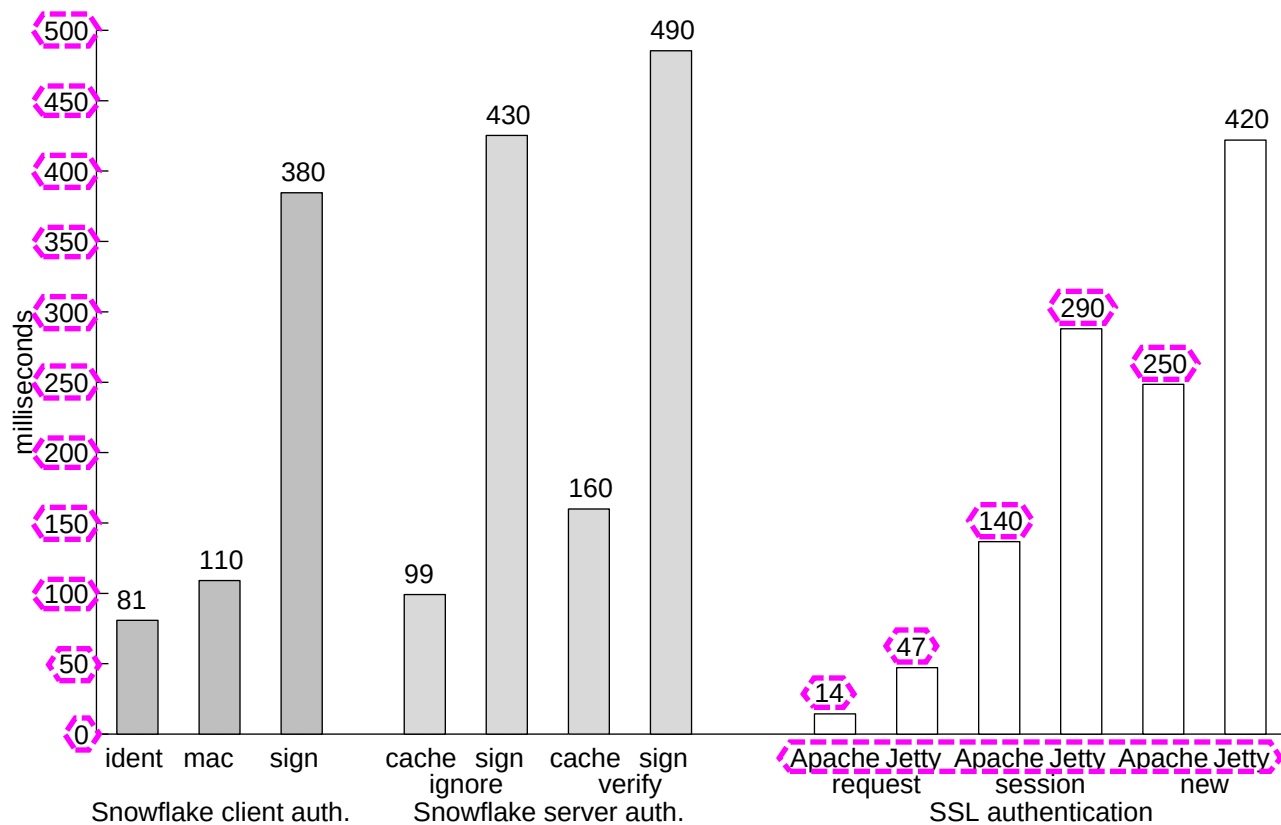


Figure 12.3: *This graph compares Snowflake client authorization, server (document) authentication, and standard SSL authentication. The left group draws from Table 12.6, the center group draws from Table 12.7, and the right group draws from Tables 12.8 and 12.9.*

client and server are executing HTTP version 1.1 rather than 1.0.

12.3.3 Network baseline

I ran the same tests on a shared segment of 10 Mbps Ethernet, with no intervening router. Not surprisingly, the per-request costs in the remote case are very similar to those in the local case, since a connection setup only incurs a few sub-microsecond latencies. Per-megabyte costs snap to about one second per megabyte, reflecting the 10 Mbps bandwidth of the cable.

12.3.4 Snowflake costs

Now that I have established a baseline, what are the costs of my implementation of Snowflake services over HTTP? As shown in Table 12.6, a fresh connection costs around 380 ms to establish, including making a public-key signature of the outbound request. Replaying the same request costs only about 81 ms, since the **Prover** has the required proof in cache.

Using the MAC optimization from Section 10.3.1 brings the cost for authorizing a new request to 110 ms. The extra 28 ms over an identical request reflects the fact that I do all the same work (hashing the request), plus an additional hash to include the MAC, plus parsing and verifying the proof at the server.

requests	request ms	copy ms/MB	σ ms	R^2
signed	380	190	10	0.21
remote	380	980	8.8	0.91
identical	81	300	1.9	0.95
remote	85	920	13	0.81
MAC opt	110	300	1.6	0.97
remote	110	990	1.7	

Table 12.6: *Snowflake HTTP client authorization performance. The three major cases are new requests that require a public-key signature, a repeated request with a cached proof, and new requests that exploit the MAC protocol.*

Table 12.7 presents my measurements of the server document authentication protocol described in Section 10.3.2. In this experiment, the client presented identical requests. Signing a document $\langle \text{sign}, \text{ignore} \rangle$ is dominated by a public-key operation, hence the 430 ms connection cost. It also incurs a hashing operation that applies to the entire document, increasing the copy cost. Caching document signatures $\langle \text{cache}, \text{ignore} \rangle$ eliminates both expenses; the increase from 86 ms to 99 ms is due to formatting the cached proof for inclusion in the reply header.

The final two cases correspond to the previous two, but now the client actually parses and verifies the signature on the document and finds a proof of the server’s authority in its **Prover**. The per-document cost increases by about 60 ms for parsing the server’s proof and querying the **Prover** to complete the proof and yield a name binding. The data copy cost increases since the client must also hash the document. The differences in copy cost in the remote cases appear because hashing operations and network delivery overlap.

12.3.5 Comparison with SSL costs

How do my channels compare in cost with alternative approaches to authorization? The obvious alternative is the Secure Sockets Layer (SSL) or its offspring, the Transport Layer Security (TLS) protocol. I measured the costs associated with SSL in two server configurations: a C implementation, Apache plus OpenSSL; and a pure Java implementation, Jetty plus PureTLS (SSL protocol) plus Cryptix (cryptographic primitive implementation). The client was always my Java client with PureTLS and Cryptix. For posterity, I also measured my simple Java web server with PureTLS and Cryptix, and found that in fact the 15 ms connection setup difference between Jetty and my simple server was lost in the noise of the long SSL connection establishment. I used 1024-bit RSA keys on both the server and client in the SSL experiments so they would compare with the Snowflake measurements.

With the Apache server, an SSL connection takes 230 ms to establish a session to a

request to 110 ms. The extra 28 ms over an identical request reflects the fact that I do all the same work (hashing the request), plus an additional hash to include the MAC, plus parsing and verifying the proof at the server.

requests	request ms	copy ms/MB	σ ms	R^2
signed	380	190	10	0.21
remote	380	980	8.8	0.91
identical	81	300	1.9	0.95
remote	85	920	13	0.81
MAC opt	110	300	1.6	0.97
remote	110	990	1.7	

Table 12.6: *Snowflake HTTP client authorization performance. The three major cases are new requests that require a public-key signature, a repeated request with a cached proof, and new requests that exploit the MAC protocol.*

Table 12.7 presents my measurements of the server document authentication protocol described in Section 10.3.2. In this experiment, the client presented identical requests. Signing a document $\langle sign, ignore \rangle$ is dominated by a public-key operation, hence the 430 ms connection cost. It also incurs a hashing operation that applies to the entire document, increasing the copy cost. Caching document signatures $\langle cache, ignore \rangle$ eliminates both expenses; the increase from 86 ms to 99 ms is due to formatting the cached proof for inclusion in the reply header.

The final two cases correspond to the previous two, but now the client actually parses and verifies the signature on the document and finds a proof of the server’s authority in its **Prover**. The per-document cost increases by about 60 ms for parsing the server’s proof and querying the **Prover** to complete the proof and yield a name binding. The data copy cost increases since the client must also hash the document. The differences in copy cost in the remote cases appear because hashing operations and network delivery overlap.

12.3.5 Comparison with SSL costs

How do my channels compare in cost with alternative approaches to authorization? The obvious alternative is the Secure Sockets Layer (SSL) or its offspring, the Transport Layer Security (TLS) protocol. I measured the costs associated with SSL in two server configurations: a C implementation, Apache plus OpenSSL; and a pure Java implementation, Jetty plus PureTLS (SSL protocol) plus Cryptix (cryptographic primitive implementation). The client was always my Java client with PureTLS and Cryptix. For posterity, I also measured my simple Java web server with PureTLS and Cryptix, and found that in fact the 15 ms connection setup difference between Jetty and my simple server was lost in the noise of the long SSL connection establishment. I used 1024-bit RSA keys on both the server and client in the SSL experiments so they would compare with the Snowflake measurements.

With the Apache server, an SSL connection takes 650 ms to establish a session to a new server once the client’s local context has been initialized, and 190 ms to rebuild using

server behavior	client behavior	request ms	copy ms/MB	σ ms	R^2
none	ignore	86	290	2.2	0.96
	remote	84	960	1.9	
sign	ignore	430	760	7.0	0.93
	remote	430	1400	9.0	0.97
cache	ignore	99	300	1.9	0.97
	remote	100	910	5.5	0.97
sign	verify	490	1200	9.6	0.95
	remote	480	1800	11	0.97
cache	verify	160	560	14	0.68
	remote	150	1300	5.2	

Table 12.7: *Snowflake HTTP server authorization performance. The experiments vary according to whether the server signed the returned document, used a cached proof, or sent no proof at all, and whether the client verified the document’s authenticity.*

new server once the client’s local context has been initialized, and 140 ms to rebuild using a cached session. A cached session reuses secret-key encryption parameters, saving two expensive public-key operations. Jetty takes more time: 390 ms to set up a fresh connection, and 290 ms to reuse an existing session. In Section 12.5, I discuss the relationship between SSL and Snowflake HTTP authorization that explains why these costs are different than those in Tables 12.6 and 12.7.

Once SSL establishes a session and authenticates its endpoints, it makes no authorization decisions per-request. The fixed cost of processing a single HTTP request on an established connection is 14 ms/req and 47 ms/req with Apache and Jetty servers, respectively.

12.4 Gateway authorization

My informal tests of the gateway application show that it has a latency of about 770 ms. This exorbitant cost reflects the fact that the gateway transaction always involves two public-key signatures. I have not optimized the protocol to eliminate public-key encryptions in the common case, for example, by implementing the MAC protocol in the gateway.

12.5 Observations

I hypothesize that the Snowflake authorization model is not prohibitively expensive. In fact, because it can subsume many hop-by-hop authorization models, it allows applications and users to make security-performance tradeoffs freely by selecting alternate hop-by-hop authorization protocols and plugging them into the same authorization framework.

Do my measurements support my hypothesis? Unfortunately, since my implementation is unoptimized and built on top of slow libraries, the numbers do not support my hypothesis unequivocally. By comparing them with baseline experiments, however, I believe I can

server behavior	client behavior	request ms	copy ms/MB	σ ms	R^2
none	ignore	86	290	2.2	0.96
	remote	84	960	1.9	
sign	ignore	430	760	7.0	0.93
	remote	430	1400	9.0	0.97
cache	ignore	99	300	1.9	0.97
	remote	100	910	5.5	0.97
sign	verify	490	1200	9.6	0.95
	remote	480	1800	11	0.97
cache	verify	160	560	14	0.68
	remote	150	1300	5.2	

Table 12.7: *Snowflake HTTP server authorization performance. The experiments vary according to whether the server signed the returned document, used a cached proof, or sent no proof at all, and whether the client verified the document’s authenticity.*

▲ a cached session. A cached session reuses secret-key encryption parameters, saving two expensive public-key operations. Surprisingly, the Jetty server takes less time (390 ms) to set up a new connection. Session reuse with Jetty takes 290 ms. In Section 12.5, I discuss the relationship between SSL and Snowflake HTTP authorization that explains why these costs are different than those in Tables 12.6 and 12.7.

Once SSL establishes a session and authenticates its endpoints, it makes no authorization decisions per-request. The fixed cost of processing a single HTTP request on an established connection is 13 ms/req and 47 ms/req with Apache and Jetty servers, respectively.

12.4 Gateway authorization

My informal tests of the gateway application show that it has a latency of about 770 ms. This exorbitant cost reflects the fact that the gateway transaction always involves two public-key signatures. I have not optimized the protocol to eliminate public-key encryptions in the common case, for example, by implementing the MAC protocol in the gateway.

12.5 Observations

I hypothesize that the Snowflake authorization model is not prohibitively expensive. In fact, because it can subsume many hop-by-hop authorization models, it allows applications and users to make security-performance tradeoffs freely by selecting alternate hop-by-hop authorization protocols and plugging them into the same authorization framework.

Do my measurements support my hypothesis? Unfortunately, since my implementation is unoptimized and built on top of slow libraries, the numbers do not support my hypothesis unequivocally. By comparing them with baseline experiments, however, I believe I can make a strong case for the hypothesis. In the next two sections, I examine the two parts

server	cache	request ms	copy ms/MB	σ ms
Apache	none	250	11000	25
	remote	230	11000	9.0
	context	230	11000	16
	remote	220	11000	21
	session	140	11000	11
Jetty	remote	180	11000	9.8
	none	420	24000	31
	remote	440	12000	23
	context	390	25000	23
	remote	420	12000	37
	session	290	24000	45
	remote	300	13000	46

Table 12.8: The SSL setup and bandwidth experiment. In the context case, I discard cached sessions; in the session case, I allow SSL to cache and reuse sessions.

server	connect ms	request ms/req	σ ms	R^2
Apache	130	14	19	0.96
remote	170	14	26	0.93
Jetty	260	47	45	0.98
remote	240	46	8.5	

Table 12.9: The cost of a single request over SSL. The first column shows cost of establishing a connection with session caching, and the second column is the cost to send a 100-byte request over the connection.

make a strong case for the hypothesis. In the next two sections, I examine the two parts of my hypothesis. In Section 12.5.3, I argue that an optimized Snowflake promises to be competitive with existing hop-by-hop protocols. I summarize my observations with some lessons learned in the process of measuring my implementation.

12.5.1 Comparable operations

Snowflake-enhanced protocols are not inherently more expensive than other protocols with similar guarantees. The measurements displayed in Figure 12.3 indicate that Snowflake performs similar encryption steps as SSL. SSL spends about 400 ms starting up, as does Snowflake. SSL can complete a request over an established channel in about 50 ms. With my MAC optimization, a Snowflake request takes about 110 ms. As I discuss in Section 12.5.3, I have observed that most of the difference is attributable to slow libraries.

Both SSL and Snowflake engage in similar operations. SSL verifies message authenticity with symmetric-key decryption and a CRC; Snowflake does the same with an MD5 hash.

server	cache	request	copy	σ
		ms	ms/MB	ms
Apache	none	670	14000	15
	remote	710	11000	43
	context	650	15000	36
	remote	680	11000	8.5
	session	190	14000	39
Jetty	remote	240	11000	23
	none	420	24000	31
	remote	440	12000	23
	context	390	25000	23
	remote	420	12000	37
	session	290	24000	45
	remote	300	13000	46

Table 12.8: The SSL setup and bandwidth experiment. In the context case, I discard cached sessions; in the session case, I allow SSL to cache and reuse sessions.

server	connect	request	σ	R^2
	ms	ms/req	ms	
Apache	180	13	18	0.97
remote	210	16	38	0.90
Jetty	260	47	45	0.98
remote	240	46	8.5	

Table 12.9: The cost of a single request over SSL. The first column shows cost of establishing a connection with session caching, and the second column is the cost to send a 100-byte request over the connection.

of my hypothesis. In Section 12.5.3, I argue that an optimized Snowflake promises to be competitive with existing hop-by-hop protocols. I summarize my observations with some lessons learned in the process of measuring my implementation.

12.5.1 Comparable operations

Snowflake-enhanced protocols are not inherently more expensive than other protocols with similar guarantees. The measurements displayed in Figure 12.3 indicate that Snowflake performs similar encryption steps as SSL. SSL spends about 400 ms starting up, as does Snowflake. SSL can complete a request over an established channel in about 50 ms. With my MAC optimization, a Snowflake request takes about 110 ms. As I discuss in Section 12.5.3, I have observed that most of the difference is attributable to slow libraries.

Both SSL and Snowflake engage in similar operations. SSL verifies message authenticity with symmetric-key decryption and a CRC; Snowflake does the same with an MD5 hash. Regardless of protocol, the server parses and processes the request and returns the reply.

Regardless of protocol, the server parses and processes the request and returns the reply. SSL checksums and encrypts the reply; Snowflake securely hashes the reply document. In both cases, the client uses a corresponding operation to verify the reply. Because the expensive cryptographic operations are comparable, one expects optimized implementations to perform comparably.

The additional sources of overhead in Snowflake are time spent walking the proof graph and memory consumed maintaining cached proofs. My experiments do not explore that space in depth, but as I describe in Section 10.3.4, proofs are usually constructed incrementally while walking the name graph, an operation driven by the client user or application.

12.5.2 The performance-security tradeoff

By comparing my authorized-request protocol to SSL I somewhat compare apples and oranges, for the protocols make different performance-security tradeoffs. For example, SSL authenticates the client and the server applications and provides secrecy and integrity for the entire communications stream. The Snowflake HTTP protocol I have described, in contrast, shows the chain of authority over requests and over document authentication, but only provides integrity for requests, and only shows integrity of reply documents, not of headers or other information supplied by the server.

That tradeoff, however, is part of my point. With Snowflake, one is free to choose an established hop-by-hop protocol or to develop a new one. By stating in my logic the authorization promises the protocol makes, one can integrate the protocol into Snowflake's end-to-end authorization model. Conceivably, new protocols can be dynamically integrated into existing Snowflake-aware applications; in other cases, a protocol-translating gateway can introduce the new protocol to the distributed system without hiding authorization information from the underlying application.

12.5.3 Slow libraries

My formal measurements and informal tests indicate that a large fraction of Snowflake's cost is needless overhead. As Table 12.4 shows, my baseline HTTP measurements indicate that using Java and the convenient Jetty web server incurs substantial overhead (250%). Furthermore, Table 12.8 shows that the Java encryption library Cryptix imposes a substantial bandwidth overhead.

What surprised me most was the overhead of the SPKI implementation on which I built Snowflake's objects. In informal tests, parsing a 2 KB S-expression from a string takes around 20 ms, and converting the resulting tree into typed Java objects takes another 20 ms. Indeed, even basic operations like `hashCode()` took over 5 ms until I made simple fixes that improved the performance by an order of magnitude. There is no reason a well-implemented library should spend milliseconds parsing short strings in a simple language; and 40+ ms delays such as these explain much of the difference between Snowflake's warm-connection performance and that of simple HTTP transactions. I am considering a full reimplement of the SPKI libraries and adopting native crypto providers to eliminate a large fraction of the inefficiencies in my implementation and make it achieve production-system performance.

SSL checksums and encrypts the reply; Snowflake securely hashes the reply document. In both cases, the client uses a corresponding operation to verify the reply. Because the expensive cryptographic operations are comparable, one expects optimized implementations to perform comparably.

The additional sources of overhead in Snowflake are time spent walking the proof graph and memory consumed maintaining cached proofs. My experiments do not explore that space in depth, but as I describe in Section 10.3.4, proofs are usually constructed incrementally while walking the name graph, an operation driven by the client user or application.

12.5.2 The performance-security tradeoff

By comparing my authorized-request protocol to SSL I somewhat compare apples and oranges, for the protocols make different performance-security tradeoffs. For example, SSL authenticates the client and the server applications and provides secrecy and integrity for the entire communications stream. The Snowflake HTTP protocol I have described, in contrast, shows the chain of authority over requests and over document authentication, but only provides integrity for requests, and only shows integrity of reply documents, not of headers or other information supplied by the server.

That tradeoff, however, is part of my point. With Snowflake, one is free to choose an established hop-by-hop protocol or to develop a new one. By stating in my logic the authorization promises the protocol makes, one can integrate the protocol into Snowflake's end-to-end authorization model. Conceivably, new protocols can be dynamically integrated into existing Snowflake-aware applications; in other cases, a protocol-translating gateway can introduce the new protocol to the distributed system without hiding authorization information from the underlying application.

12.5.3 Slow libraries

My formal measurements and informal tests indicate that a large fraction of Snowflake's cost is needless overhead. As Table 12.4 shows, my baseline HTTP measurements indicate that using Java and the convenient Jetty web server incurs substantial overhead (250%). Furthermore, Table 12.8 shows that the Java encryption library Cryptix imposes a substantial bandwidth overhead.

What surprised me most was the overhead of the SPKI implementation on which I built Snowflake's objects. In informal tests, parsing a 2 KB S-expression from a string takes around 20 ms, and converting the resulting tree into typed Java objects takes another 20 ms. Indeed, even basic operations like `hashCode()` took over 5 ms until I made simple fixes that improved the performance by an order of magnitude. There is no reason a well-implemented library should spend milliseconds parsing short strings in a simple language; and 40+ ms delays such as these explain much of the difference between Snowflake's warm-connection performance and that of simple HTTP transactions. I am considering a full reimplement of the SPKI libraries and adopting native crypto providers to eliminate a large fraction of the inefficiencies in my implementation and make it achieve production-system performance.

12.5.4 Performance lessons

While developing these experiments, I learned two lessons about performance and measurement. First, do not forget to close sockets when they are no longer in use, for extra file descriptors slow things down [BM98]. Second, when latency matters, be careful to disable Nagle's algorithm, which delays delivery of packets in hopes that data that can be aggregated will arrive. Oddly, Nagle's algorithm is even used on connections to the local host. I am also surprised that the interface to Nagle's algorithm is a single bit on the file descriptor. Turning off the algorithm is analogous to turning off buffering in the C `stdio` library: why not supply a `flush` call instead to give the application more control?

12.5.4 Performance lessons

While developing these experiments, I learned two lessons about performance and measurement. First, do not forget to close sockets when they are no longer in use, for extra file descriptors slow things down [BM98]. Second, when latency matters, be careful to disable Nagle's algorithm, which delays delivery of a packet in hopes that the application will soon send more data that can be aggregated into the same packet. Oddly, Nagle's algorithm is even used on connections to the local host. I am also surprised that the interface to Nagle's algorithm is a single bit on the file descriptor. Turning off the algorithm is analogous to turning off buffering in the C `stdio` library; why not supply a `flush` call instead to give the application more control?

Appendix A

Proofs

A.1 Construction of ϕ_T

Definition E12 presupposed the existence of a projection function ϕ_T . We construct such a function now, and show that it satisfies the definition. Let $\overline{W} = 2^T$; that is, $\overline{W}$ are subsets of T . Define

$$\phi_T(w) = \overline{w} \in \overline{W}$$

where $(\sigma \in \overline{w}) \equiv (w \in \mathcal{E}(\sigma)) \forall \sigma \in T$ (Definition E26)

Necessity. Given $\phi_T(w) = \overline{w} = \phi_T(w')$, we know $\forall \sigma \in T, \sigma \in \overline{w}$ iff $w \in \mathcal{E}(\sigma)$, and likewise, $\forall \sigma \in T, \sigma \in \overline{w}$ iff $w' \in \mathcal{E}(\sigma)$. Therefore $\forall \sigma \in T, w \in \mathcal{E}(\sigma)$ iff $w' \in \mathcal{E}(\sigma)$, and we conclude $w \cong_T w'$.

Sufficiency. From the definition of $w \cong_T w'$, we know $\forall \sigma \in T, w \in \mathcal{E}(\sigma)$ iff $w' \in \mathcal{E}(\sigma)$. Let $\overline{w} = \{\sigma \in T | w \in \mathcal{E}(\sigma)\}$ and $\overline{w'} = \{\sigma \in T | w' \in \mathcal{E}(\sigma)\}$. From our hypothesis we know that the conditions on $\overline{w}$ and $\overline{w'}$ are the same, so $\phi_T(w) = \overline{w} = \overline{w'} = \phi_T(w')$.

In the following proofs, I generally use a bar ($\overline{w}$) to indicate a member of an equivalence class constructed as shown here.

A.2 Equivalence of ϕ_T^w and ϕ_T^+ definitions of $\xRightarrow{T}$

I now justify my claim in Section 4.2 that Definition E14 and Definition E16 are equivalent.

Necessity. Assume $\mathcal{B} \xRightarrow{T} \mathcal{A}$ holds according to Definition E14:

$$\forall w'_0 \quad (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

Appendix A

Proofs

A.1 Construction of ϕ_T

Definition E12 presupposed the existence of a projection function ϕ_T . We construct such a function now, and show that it satisfies the definition. Let $\overline{W} = 2^T$; that is, worlds in $\overline{M}$ are subsets of T . Define

$$\phi_T(w) = \overline{w} \in \overline{W}$$

where $(\sigma \in \overline{w}) \equiv (w \in \mathcal{E}(\sigma)) \ \forall \sigma \in T$ (Definition E26)

Necessity. Given $\phi_T(w) = \overline{w} = \phi_T(w')$, we know $\forall \sigma \in T, \sigma \in \overline{w}$ iff $w \in \mathcal{E}(\sigma)$, and likewise, $\forall \sigma \in T, \sigma \in \overline{w}$ iff $w' \in \mathcal{E}(\sigma)$. Therefore $\forall \sigma \in T, w \in \mathcal{E}(\sigma)$ iff $w' \in \mathcal{E}(\sigma)$, and we conclude $w \cong_T w'$.

Sufficiency. From the definition of $w \cong_T w'$, we know $\forall \sigma \in T, w \in \mathcal{E}(\sigma)$ iff $w' \in \mathcal{E}(\sigma)$. Let $\overline{w} = \{\sigma \in T | w \in \mathcal{E}(\sigma)\}$ and $\overline{w'} = \{\sigma \in T | w' \in \mathcal{E}(\sigma)\}$. From our hypothesis we know that the conditions on $\overline{w}$ and $\overline{w'}$ are the same, so $\phi_T(w) = \overline{w} = \overline{w'} = \phi_T(w')$.

In the following proofs, I generally use a bar ($\overline{w}$) to indicate a member of an equivalence class constructed as shown here.

A.2 Equivalence of ϕ_T^w and ϕ_T^+ definitions of $\xRightarrow{T}$

I now justify my claim in Section 4.2 that Definition E14 and Definition E16 are equivalent.

Necessity. Assume $\mathcal{B} \xRightarrow{T} \mathcal{A}$ holds according to Definition E14:

$$\forall w'_0 \ (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

For all $\langle w_0, w_1 \rangle$,

$$\begin{aligned}
\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{A}) &\supset w_1 \in \mathcal{R}(\mathcal{A})(w_0) \\
&\supset \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)), \\
&\quad \bar{w}_1 = \phi_T(w_1) \\
&\supset \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) && \text{(using the assumption)} \\
&\supset \exists w'_1 \cong_T w_1, \\
&\quad \langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})(w_0) \\
&\supset \langle w_0, w_1 \rangle \in \phi_T^+(\mathcal{R}(\mathcal{B})) && \text{(Definition E15)}
\end{aligned}$$

Sufficiency. Assume $\mathcal{B} \xRightarrow{T} \mathcal{A}$ holds according to Definition E16:

$$\mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B}))$$

Given w_0 and $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0))$, we know that there is some $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\bar{w}_1 = \phi_T(w_1)$. We rewrite the statement $\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{A})$, and invoke the assumption to get $\langle w_0, w_1 \rangle \in \phi_T^+(\mathcal{R}(\mathcal{B}))$. Now we know there exists $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ with $w'_1 \cong_T w_1$. Changing notation again, $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$. Since $w'_1 \cong_T w_1$, we know $\bar{w}_1 = \phi_T(w'_1)$, and we may conclude $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$.

Together, the two implications show the equivalence.

A.3 An undesirable semantics for $\xRightarrow{T}$

Notice that ϕ_T^+ projects only the destination world of each edge in a relation. Why do we not project both ends of the relation? Such a definition actually does not preserve our most basic intuition, that $B \xRightarrow{T} A \supset B \xrightarrow{T} A$. In the model in Figure A.1, the grey boxes depict the equivalence classes under T ; projecting both ends of the edges in $\mathcal{R}(A)$ gives $\{\langle T, \emptyset \rangle\}$, as does $\mathcal{R}(B)$. From world w_0 , however, B says s but not A says s .

Given a relation $\langle w_0, w_1 \rangle$, then, the reason we only project w_1 is this: w_0 is affected by what statements are true at w_1 ; substituting other worlds equivalent with respect to T does no harm. Substituting other worlds for w_0 , on the other hand, changes what statements we consider true at w_0 .

A.4 Proof of soundness

In this section, I show that my extension to Lampson's calculus is still a sound axiomatization of the presented semantics. Like Lampson's original logic, mine is based on a conventional Kripke semantics of modal logic. The conventional proofs of soundness for

For all $\langle w_0, w_1 \rangle$,

$$\begin{aligned}
\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{A}) &\supset w_1 \in \mathcal{R}(\mathcal{A})(w_0) \\
&\supset \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \\
&\quad \bar{w}_1 = \phi_T(w_1) \\
&\supset \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) && \text{(using the assumption)} \\
&\supset \exists w'_1 \cong_T w_1 \\
&\quad \langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})(w_0) \\
&\supset \langle w_0, w_1 \rangle \in \phi_T^+(\mathcal{R}(\mathcal{B})) && \text{(by Definition E15)}
\end{aligned}$$

Sufficiency. Assume $\mathcal{B} \xRightarrow{T} \mathcal{A}$ holds according to Definition E16:

$$\mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B}))$$

Given w_0 and $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0))$, we know that there is some $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\bar{w}_1 = \phi_T(w_1)$. We rewrite the statement $\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{A})$, and invoke the assumption to get $\langle w_0, w_1 \rangle \in \phi_T^+(\mathcal{R}(\mathcal{B}))$. Now we know there exists $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ with $w'_1 \cong_T w_1$. Changing notation again, $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$. Since $w'_1 \cong_T w_1$, we know $\bar{w}_1 = \phi_T(w'_1)$, and we may conclude $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$.

Together, the two implications show the equivalence.

A.3 An undesirable semantics for $\xRightarrow{T}$

Notice that ϕ_T^+ projects only the destination world of each edge in a relation. Why do we not project both ends of the relation? Such a definition actually does not preserve our most basic intuition, that $B \xRightarrow{T} A \supset B \xrightarrow{T} A$. In the model in Figure A.1, the grey boxes depict the equivalence classes under T ; projecting both ends of the edges in $\mathcal{R}(A)$ gives $\{\langle T, \emptyset \rangle\}$, as does $\mathcal{R}(B)$. From world w_0 , however, B says s but not A says s .

Given a relation $\langle w_0, w_1 \rangle$, then, the reason we only project w_1 is this: w_0 is affected by what statements are true at w_1 ; substituting other worlds equivalent with respect to T does no harm. Substituting other worlds for w_0 , on the other hand, changes what statements we consider true at w_0 .

A.4 Proof of soundness

In this section, I show that my extension to Lampson's calculus is still a sound axiomatization of the presented semantics. Like Lampson's original logic, mine is based on a conventional Kripke semantics of modal logic. The conventional proofs of soundness for

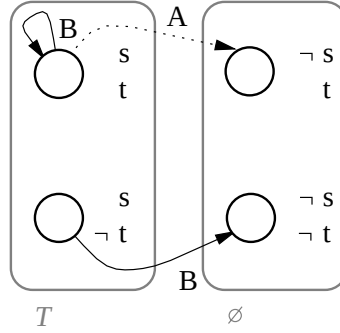


Figure A.1: In this example, $T = \{s\}$. Notice that $B \not\stackrel{T}{\Rightarrow} A$.

Axiom S1, Rule S2, Axiom S3, and Rule S4 apply. My extensions define $\mathcal{E}$ for a new formula ($\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}$) and $\mathcal{R}$ for a new principal ($\mathcal{A} \cdot N$), but do not perturb Abadi's original semantics for the calculus for access control. Because those semantics do not depend on any particular structure in $\mathcal{E}$ or $\mathcal{R}$, the axioms of the calculus remain sound in my extended calculus.

My present task is to show that the axioms of my extensions are sound.

Axiom E1. $\vdash (\mathcal{C} \stackrel{T}{\Rightarrow} \mathcal{B}) \wedge (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{C} \stackrel{T}{\Rightarrow} \mathcal{A})$. This axiom follows easily from Definition E14. For all w_0 ,

$$\begin{aligned} \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) &\subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \\ &\subseteq \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) \end{aligned}$$

□

The following lemma shows that ϕ_T^+ preserves the union operation. Let R_1 and R_2 be relations.

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \phi_T^+(R_1 \cup R_2) \\ &\equiv \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_1 \cup R_2 \\ &\equiv \exists w'_1 \cong_T w_1, \\ &\quad \langle w_0, w'_1 \rangle \in R_1 \vee \langle w_0, w'_1 \rangle \in R_2 \\ &\equiv \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_1 \\ &\quad \vee \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_2 \\ &\equiv \langle w_0, w_1 \rangle \in \phi_T^+(R_1) \vee \langle w_0, w_1 \rangle \in \phi_T^+(R_2) \\ &\equiv \langle w_0, w_1 \rangle \in \phi_T^+(R_1) \cup \phi_T^+(R_2) \end{aligned}$$

From this equivalence we conclude

$$\phi_T^+(R_1 \cup R_2) = \phi_T^+(R_1) \cup \phi_T^+(R_2) \quad (\text{Lemma E27})$$

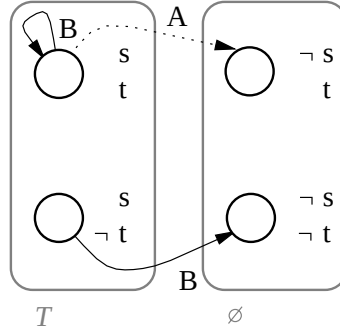


Figure A.1: In this example, $T = \{s\}$. Notice that $B \not\stackrel{T}{\Rightarrow} A$.

Axiom S1, Rule S2, Axiom S3, and Rule S4 apply. My extensions define $\mathcal{E}$ for a new formula ($\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}$) and $\mathcal{R}$ for a new principal ($\mathcal{A} \cdot N$), but do not perturb Abadi's original semantics for the calculus for access control. Because those semantics do not depend on any particular structure in $\mathcal{E}$ or $\mathcal{R}$, the axioms of the calculus remain sound in my extended calculus.

My present task is to show that the axioms of my extensions are sound.

Axiom E1. $\vdash (\mathcal{C} \stackrel{T}{\Rightarrow} \mathcal{B}) \wedge (\mathcal{B} \stackrel{T}{\Rightarrow} \mathcal{A}) \supset (\mathcal{C} \stackrel{T}{\Rightarrow} \mathcal{A})$. This axiom follows easily from Definition E14. For all w_0 ,

$$\begin{aligned} \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) &\subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \\ &\subseteq \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) \end{aligned}$$

□

The following lemma shows that ϕ_T^+ preserves the union operation. Let R_1 and R_2 be relations.

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \phi_T^+(R_1 \cup R_2) \\ &\equiv \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_1 \cup R_2 \\ &\equiv \exists w'_1 \cong_T w_1, \\ &\quad \langle w_0, w'_1 \rangle \in R_1 \vee \langle w_0, w'_1 \rangle \in R_2 \\ &\equiv \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_1 \\ &\quad \vee \exists w'_1 \cong_T w_1, \langle w_0, w'_1 \rangle \in R_2 \\ &\equiv \langle w_0, w_1 \rangle \in \phi_T^+(R_1) \vee \langle w_0, w_1 \rangle \in \phi_T^+(R_2) \\ &\equiv \langle w_0, w_1 \rangle \in \phi_T^+(R_1) \cup \phi_T^+(R_2) \end{aligned}$$

From this equivalence we conclude

$$\phi_T^+(R_1 \cup R_2) = \phi_T^+(R_1) \cup \phi_T^+(R_2) \quad (\text{Lemma E27})$$

Axiom E2. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \wedge \mathcal{C}) \xRightarrow{T} (\mathcal{A} \wedge \mathcal{C})$. We assume the premise in terms of Definition E14:

$$\forall w'_0 \ (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

We can readily reason for all w_0 :

$$\begin{aligned} & \phi_T^w(\mathcal{R}(\mathcal{A} \wedge \mathcal{C})(w_0)) \\ &= \phi_T^w((\mathcal{R}(\mathcal{A}) \cup \mathcal{R}(\mathcal{C}))(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{A})(w_0) \cup \mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \cup \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) && \text{(Lemma E27)} \\ &\subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \cup \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{B})(w_0) \cup \mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w((\mathcal{R}(\mathcal{B}) \cup \mathcal{R}(\mathcal{C}))(w_0)) && \text{(Lemma E27)} \\ &= \phi_T^w(\mathcal{R}(\mathcal{B} \wedge \mathcal{C})(w_0)) \quad \square \end{aligned}$$

Axiom E3. $\vdash (\mathcal{C} \xRightarrow{T} \mathcal{A}) \wedge (\mathcal{C} \xRightarrow{T} \mathcal{B}) \equiv \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B})$. The equivalence depends on Lemma E27:

$$\begin{aligned} & \mathcal{M} \models \mathcal{C} \xRightarrow{T} \mathcal{A} \wedge \mathcal{C} \xRightarrow{T} \mathcal{B} \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ & \quad \wedge \quad \phi_T^+(\mathcal{R}(\mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A})) \cup \phi_T^+(\mathcal{R}(\mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A} \cup \mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) && \text{(Lemma E27)} \\ \text{iff} & \quad \mathcal{M} \models \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B}) \quad \square \end{aligned}$$

To justify Axiom E4 we must first show that for any relation R ,

$$\phi_U^+(R) = R \quad \text{(Lemma E28)}$$

Lemma E28. The lemma holds because we may discard *identical worlds* from a model

without loss of generality. That is, imagine we have a model $\mathcal{M}$ with two worlds w_1 and w_2 where $w_1 \in \mathcal{E}(\sigma)$ iff $w_2 \in \mathcal{E}(\sigma)$ for every formula $\sigma \in \Sigma$. The extra world w_2 appears in every $I(s)$ in which w_1 appears. Any edge in any relation ending in w_1 has a related edge ending in w_2 ($\langle w, w_1 \rangle \in J(\mathcal{A}) \equiv \langle w, w_2 \rangle \in J(\mathcal{A})$); likewise edges starting at w_1 have a related edge starting at w_2 in every relation. The same holds for the relations in the name interpretation function $K(\mathcal{A}, N)$. It is clear that the extension function $\mathcal{R}$, and hence $\mathcal{E}$, have the same overlap with respect to w_1 and w_2 , so that $w_1 \in \mathcal{E}(\sigma) \equiv w_2 \in \mathcal{E}(\sigma)$.

=2=

=2=

Axiom E2. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{B} \wedge \mathcal{C}) \xRightarrow{T} (\mathcal{A} \wedge \mathcal{C})$. We assume the premise in terms of Definition E14:

$$\forall w'_0 \ (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

We can readily reason for all w_0 :

$$\begin{aligned} & \phi_T^w(\mathcal{R}(\mathcal{A} \wedge \mathcal{C})(w_0)) \\ &= \phi_T^w((\mathcal{R}(\mathcal{A}) \cup \mathcal{R}(\mathcal{C}))(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{A})(w_0) \cup \mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \cup \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) && \text{(Lemma E27)} \\ &\subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \cup \phi_T^w(\mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w(\mathcal{R}(\mathcal{B})(w_0) \cup \mathcal{R}(\mathcal{C})(w_0)) \\ &= \phi_T^w((\mathcal{R}(\mathcal{B}) \cup \mathcal{R}(\mathcal{C}))(w_0)) && \text{(Lemma E27)} \\ &= \phi_T^w(\mathcal{R}(\mathcal{B} \wedge \mathcal{C})(w_0)) \quad \square \end{aligned}$$

Axiom E3. $\vdash (\mathcal{C} \xRightarrow{T} \mathcal{A}) \wedge (\mathcal{C} \xRightarrow{T} \mathcal{B}) \equiv \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B})$. The equivalence depends on Lemma E27:

$$\begin{aligned} & \mathcal{M} \models \mathcal{C} \xRightarrow{T} \mathcal{A} \wedge \mathcal{C} \xRightarrow{T} \mathcal{B} \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ & \quad \wedge \quad \phi_T^+(\mathcal{R}(\mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A})) \cup \phi_T^+(\mathcal{R}(\mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) \\ \text{iff} & \quad \phi_T^+(\mathcal{R}(\mathcal{A} \cup \mathcal{B})) \subseteq \phi_T^+(\mathcal{R}(\mathcal{C})) && \text{(Lemma E27)} \\ \text{iff} & \quad \mathcal{M} \models \mathcal{C} \xRightarrow{T} (\mathcal{A} \wedge \mathcal{B}) \quad \square \end{aligned}$$

To justify Axiom E4 we must first show that for any relation R ,

$$\phi_U^+(R) = R \quad \text{(Lemma E28)}$$

Lemma E28. The lemma holds because we may discard *identical worlds* from a model

without loss of generality. That is, imagine we have a model $\mathcal{M}$ with two worlds w_1 and w_2 where $w_1 \in \mathcal{E}(\alpha)$ iff $w_2 \in \mathcal{E}(\sigma)$ for every formula $\sigma \in \Sigma$. The extra world w_2 appears in every $I(s)$ in which w_1 appears. Any edge in any relation ending in w_1 has a related edge ending in w_2 ($\langle w, w_1 \rangle \in J(\mathcal{A}) \equiv \langle w, w_2 \rangle \in J(\mathcal{A})$); likewise edges starting at w_1 have a related edge starting at w_2 in every relation. The same holds for the relations in the name interpretation function $K(\mathcal{A}, N)$. It is clear that the extension function $\mathcal{R}$, and hence $\mathcal{E}$, have the same overlap with respect to w_1 and w_2 , so that $w_1 \in \mathcal{E}(\alpha) \equiv w_2 \in \mathcal{E}(\sigma)$.

=2=

=2=

Given this definition, we can build a model $\mathcal{M}' = \langle W', w'_0, I', J', K' \rangle$ that discards w_2 :

$$\begin{aligned} W' &= W - \{w_2\} \\ w'_0 &= \begin{cases} w_1 & \text{if } w_0 \\ w_0 & \text{otherwise} \end{cases} \\ I'(s) &= I(s) - \{w_2\} \\ J'(\mathcal{A}) &= J(\mathcal{A}) - \{\langle w, w' \rangle \mid w = w_2 \vee w' = w_2\} \\ K'(\mathcal{A}, N) &= K(\mathcal{A}, N) \\ &\quad - \{\langle w, w' \rangle \mid w = w_2 \vee w' = w_2\} \end{aligned}$$

Happily, $\mathcal{M}'$ preserves every consequence of $\mathcal{M}$: $(\mathcal{M} \models \sigma) \equiv (\mathcal{M}' \models \sigma)$. Why? Whenever $w_0 \in \mathcal{E}(\sigma)$, we know $w'_0 \in \mathcal{E}'(\sigma)$, either for exactly the same reasons (when $w_0 \neq w_2$), or because $w_0 = w_2$, so $w_0 = w_2 \in \mathcal{E}(\sigma) \equiv w_1 \in \mathcal{E}(\sigma)$, and then $w'_0 \in \mathcal{E}'(s)$ for the same reasons that $w_1 \in \mathcal{E}(s)$.

Convinced that duplicate worlds do not alter the consequences of a model, we may now assume that no models contain identical worlds, without damaging the semantics. If we know $w_1 \neq w_2$, we can assume the existence of a formula σ with $(w_1 \in \mathcal{E}(\sigma)) \neq (w_2 \in \mathcal{E}(\sigma))$, and conclude that $w_1 \not\approx_{\mathcal{U}} w_2$ (by Definition E11). Therefore, $\phi_{\mathcal{U}}$ is bijective:

$$w_1 \neq w_2 \supset \phi_{\mathcal{U}}(w_1) \neq \phi_{\mathcal{U}}(w_2)$$

By the definition of ϕ_T^+ it is obvious that any relation $R \in \phi_T^+(R)$. But when $T = \mathcal{U}$, the converse is also true:

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \phi_{\mathcal{U}}^+(R) \\ &\supset \exists w'_1 \text{ such that } \langle w_0, w'_1 \rangle \in R, \\ &\quad \phi_{\mathcal{U}}(w'_1) = \phi_{\mathcal{U}}(w_1) \\ &\supset w'_1 = w_1 \\ &\supset \langle w_0, w_1 \rangle \in R \end{aligned}$$

Now we have $\phi_{\mathcal{U}}^+(R) = R$. □

Axiom E4. $\vdash (\mathcal{B} \stackrel{\mathcal{U}}{\Rightarrow} \mathcal{A}) \equiv (\mathcal{B} \Rightarrow \mathcal{A})$. Expanding the definition of $B \stackrel{\mathcal{U}}{\Rightarrow} A$ and applying the previous result gives $\mathcal{R}(A) \subseteq \phi_{\mathcal{U}}^+(\mathcal{R}(B)) = \mathcal{R}(B)$, which satisfies the definition of $B \Rightarrow A$. □

Justifying Axiom E5 requires two lemmas that relate representatives of equivalence classes under different projections.

First, a representative of a projection due to a small set has a “big brother” in any projection due to a superset, and the structure of the brothers is closely related:

$$\begin{aligned} \bar{w}'_1 &\in \phi_{T'}^w(S_w), \quad T' \subseteq T \\ &\supset \exists \bar{w}_1 \in \phi_T^w(S_w), \quad \bar{w}'_1 = \bar{w}_1 \cap T' \end{aligned} \quad (\text{Lemma E29})$$

Given this definition, we can build a model $\mathcal{M}' = \langle W', w'_0, I', J', K' \rangle$ that discards w_2 :

$$\begin{aligned} W' &= W - \{w_2\} \\ w'_0 &= \begin{cases} w_1 & \text{if } w_0 \\ w_0 & \text{otherwise} \end{cases} \\ I'(s) &= I(s) - \{w_2\} \\ J'(\mathcal{A}) &= J(\mathcal{A}) - \{\langle w, w' \rangle \mid w = w_2 \vee w' = w_2\} \\ K'(\mathcal{A}, N) &= K(\mathcal{A}, N) \\ &\quad - \{\langle w, w' \rangle \mid w = w_2 \vee w' = w_2\} \end{aligned}$$

Happily, $\mathcal{M}'$ preserves every consequence of $\mathcal{M}$: $(\mathcal{M} \models \sigma) \equiv (\mathcal{M}' \models \sigma)$. Why? Whenever $w_0 \in \mathcal{E}(\sigma)$, we know $w'_0 \in \mathcal{E}'(\sigma)$, either for exactly the same reasons (when $w_0 \neq w_2$), or because $w_0 = w_2$, so $w_0 = w_2 \in \mathcal{E}(\sigma) \equiv w_1 \in \mathcal{E}(\sigma)$, and then $w'_0 \in \mathcal{E}'(\sigma)$ for the same reasons that $w_1 \in \mathcal{E}(\sigma)$.

Convinced that duplicate worlds do not alter the consequences of a model, we may now assume that no models contain identical worlds, without damaging the semantics. If we know $w_1 \neq w_2$, we can assume the existence of a formula σ with $(w_1 \in \mathcal{E}(\sigma)) \neq (w_2 \in \mathcal{E}(\sigma))$, and conclude that $w_1 \not\equiv_{\mathcal{U}} w_2$ (by Definition E11). Therefore, $\phi_{\mathcal{U}}$ is bijective:

$$w_1 \neq w_2 \supset \phi_{\mathcal{U}}(w_1) \neq \phi_{\mathcal{U}}(w_2)$$

By the definition of ϕ_T^+ it is obvious that any relation $R \in \phi_T^+(R)$. But when $T = \mathcal{U}$, the converse is also true:

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \phi_{\mathcal{U}}^+(R) \\ &\supset \exists w'_1 \text{ such that } \langle w_0, w'_1 \rangle \in R, \\ &\quad \phi_{\mathcal{U}}(w'_1) = \phi_{\mathcal{U}}(w_1) \\ &\supset w'_1 = w_1 \\ &\supset \langle w_0, w_1 \rangle \in R \end{aligned}$$

Now we have $\phi_{\mathcal{U}}^+(R) = R$. □

Axiom E4. $\vdash (\mathcal{B} \stackrel{\mathcal{U}}{\Rightarrow} \mathcal{A}) \equiv (\mathcal{B} \Rightarrow \mathcal{A})$. Expanding the definition of $B \stackrel{\mathcal{U}}{\Rightarrow} A$ and applying the previous result gives $\mathcal{R}(A) \subseteq \phi_{\mathcal{U}}^+(\mathcal{R}(B)) = \mathcal{R}(B)$, which satisfies the definition of $B \Rightarrow A$. □

Justifying Axiom E5 requires two lemmas that relate representatives of equivalence classes under different projections.

First, a representative of a projection due to a small set has a “big brother” in any projection due to a superset, and the structure of the brothers is closely related:

$$\begin{aligned} \bar{w}'_1 &\in \phi_{T'}^w(S_w), \quad T' \subseteq T \\ &\supset \exists \bar{w}_1 \in \phi_T^w(S_w) \cdot \bar{w}'_1 = \bar{w}_1 \cap T' \end{aligned} \quad (\text{Lemma E29})$$

Lemma E29. By the first premise, there is a $w_1 \in S_w$ where $\bar{w}'_1 = \phi_{T'}(w_1)$. From Definition E26 we know

$$(\sigma \in \bar{w}'_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \forall \sigma \in T' \quad (1)$$

Let $\bar{w}_1 = \phi_T(w_1)$; since $w_1 \in S_w$, $\bar{w}_1 \in \phi_T^w(S_w)$. Having exhibited $\bar{w}_1$, we need only show $\bar{w}_1 \cap T = \bar{w}'_1$.

We again invoke Definition E26 to get

$$(\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \forall \sigma \in T \quad (2)$$

First, $\sigma \in \bar{w}_1 \cap T'$ means both $\sigma \in T'$, and because $T' \subseteq T$, $\sigma \in T$. The latter allows us to use (2) to write $w_1 \in \mathcal{E}(\sigma)$, and then we invoke (1) to get $\sigma \in \bar{w}'_1$. Conversely, $\sigma \in \bar{w}'_1$ means $\sigma \in T'$ and hence $\sigma \in T$. We apply (1) to get $w_1 \in \mathcal{E}(\sigma)$, and apply (2) to get $\sigma \in \bar{w}_1$. Now we have shown $\bar{w}_1 \cap T' = \bar{w}'_1$, proving the lemma. $\square$

The second lemma is approximately the converse of the first:

$$\begin{aligned} \bar{w}_1 \in \phi_T^w(S_w), \bar{w}'_1 = \bar{w}_1 \cap T' \quad T' \subseteq T \\ \Rightarrow \bar{w}'_1 \in \phi_{T'}^w(S_w) \end{aligned} \quad (\text{Lemma E30})$$

Lemma E30. The first premise, by Definition E13, implies the existence of a $w_1 \in R$, and Definition E26 lets us write

$$(\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \forall \sigma \in T \quad (1)$$

For every $\sigma \in T'$, all of the following hold:

$$\begin{aligned} & \sigma \in T && (\text{third premise}) \\ & (\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) && (\text{by 1}) \\ & (\sigma \in \bar{w}_1 \cap T') \equiv (w_1 \in \mathcal{E}(\sigma)) && \\ & (\sigma \in \bar{w}'_1) \equiv (w_1 \in \mathcal{E}(\sigma)) && (\text{second premise}) \end{aligned}$$

This last result implies that $\bar{w}'_1 = \phi_{T'}(w_1)$, which is sufficient to prove the conclusion of the lemma. $\square$

Axiom E5. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \Rightarrow (\mathcal{B} \xRightarrow{T'} \mathcal{A})$ when $T' \subseteq T$. We take as our hypothesis $\mathcal{M} \models B \xRightarrow{T} A$, that is:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$$

Given any world w_0 and a set $T' \subseteq T$, we assume $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$ and set out to prove $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. By the assumption and Lemma E29, we know

$$\exists \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)), \bar{w}'_1 = \bar{w}_1 \cap T'$$

Lemma E29. By the first premise, there is a $w_1 \in S_w$ where $\bar{w}'_1 = \phi_{T'}(w_1)$. From Definition E26 we know

$$(\sigma \in \bar{w}'_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \quad \forall \sigma \in T' \quad (1)$$

Let $\bar{w}_1 = \phi_T(w_1)$; since $w_1 \in S_w$, $\bar{w}_1 \in \phi_T^w(S_w)$. Having exhibited $\bar{w}_1$, we need only show $\bar{w}_1 \cap T = \bar{w}'_1$.

We again invoke Definition E26 to get

$$(\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \quad \forall \sigma \in T \quad (2)$$

First, $\sigma \in \bar{w}_1 \cap T'$ means both $\sigma \in T'$, and because $T' \subseteq T$, $\sigma \in T$. The latter allows us to use (2) to write $w_1 \in \mathcal{E}(\sigma)$, and then we invoke (1) to get $\sigma \in \bar{w}'_1$. Conversely, $\sigma \in \bar{w}'_1$ means $\sigma \in T'$ and hence $\sigma \in T$. We apply (1) to get $w_1 \in \mathcal{E}(\sigma)$, and apply (2) to get $\sigma \in \bar{w}_1$. Now we have shown $\bar{w}_1 \cap T' = \bar{w}'_1$, proving the lemma. $\square$

The second lemma is approximately the converse of the first:

$$\begin{aligned} \bar{w}_1 \in \phi_T^w(S_w), \bar{w}'_1 = \bar{w}_1 \cap T' \quad T' \subseteq T \\ \Rightarrow \bar{w}'_1 \in \phi_{T'}^w(S_w) \end{aligned} \quad (\text{Lemma E30})$$

Lemma E30. The first premise, by Definition E13, implies the existence of a $w_1 \in R$, and Definition E26 lets us write

$$(\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) \quad \forall \sigma \in T \quad (1)$$

For every $\sigma \in T'$, all of the following hold:

$$\begin{aligned} & \sigma \in T' && (\text{third premise}) \\ & (\sigma \in \bar{w}_1) \equiv (w_1 \in \mathcal{E}(\sigma)) && (\text{by 1}) \\ & (\sigma \in \bar{w}_1 \cup T') \equiv (w_1 \in \mathcal{E}(\sigma)) && \\ & (\sigma \in \bar{w}'_1) \equiv (w_1 \in \mathcal{E}(\sigma)) && (\text{second premise}) \end{aligned}$$

This last result implies that $\bar{w}'_1 = \phi_{T'}(w_1)$, which is sufficient to prove the conclusion of the lemma. $\square$

Axiom E5. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \Rightarrow (\mathcal{B} \xRightarrow{T'} \mathcal{A})$ when $T' \subseteq T$. We take as our hypothesis $\mathcal{M} \models B \xRightarrow{T} A$, that is:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$$

Given any world w_0 and a set $T' \subseteq T$, we assume $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$ and set out to prove $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. By the assumption and Lemma E29, we know

$$\exists \bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0)), \bar{w}'_1 = \bar{w}_1 \cap T'$$

The hypothesis gives $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, which satisfies the premise for Lemma E30. Hence we know $\bar{w}_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$, and we have proven that

$$\forall w_0, \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w_0)) \quad \square$$

Theorem E6. $\vdash (\mathcal{C} \xRightarrow{S} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{S \cup T} \mathcal{A})$. Apply Axiom E5 twice to the premises to get two relations restricted by $S \cup T$, then apply Axiom E1 to collapse them into the relation in the conclusion. $\square$

Result E7. $(B \xRightarrow{S} A) \wedge (B \xRightarrow{T} A) \not\supset B \xRightarrow{S \cup T} A$. Figure A.2 gives a counterexample that justifies the result. The diagram in the figure models $B \xRightarrow{S} A$ and $B \xRightarrow{T} A$. The statement $B \xRightarrow{S \cup T}$, however, fails. Projecting the model under $S \cup T$ gives the original picture, since each world falls in a separate equivalence class. Notice that B says $\neg(s \wedge \neg t)$: that statement is true in both worlds B considers possible. But A does not believe it, since A can see the lower-left world, where the statement is false.

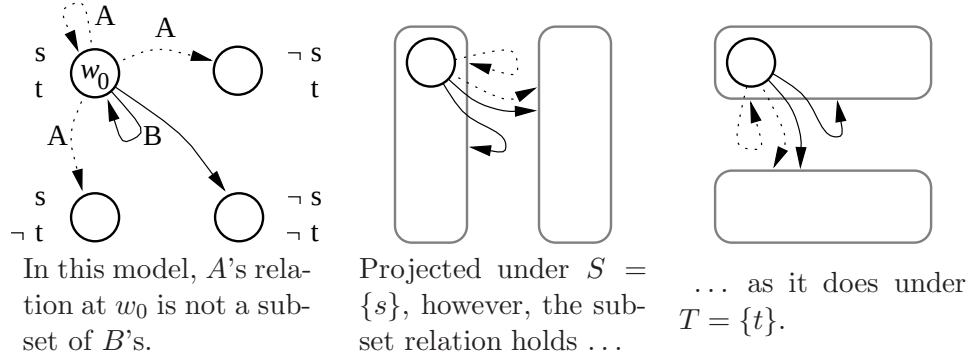


Figure A.2: A counterexample showing why two delegations for sets S and T do not imply a delegation for set $S \cup T$ (Result E7).

Why should this result be intuitive or desirable? Recall from Section 7.9 that the strength of $\xRightarrow{T}$ means that a delegation regarding T may imply a delegation regarding a larger set T^* that includes formulas constructed from the members of T . In our example, B speaks for A regarding formulas composed exclusively with the primitive s or the primitive t , but not regarding formulas combining the two. The closure of the restriction set $S \cup T$ includes formulas such as $\neg(s \wedge \neg t)$.

Axiom E8. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset \mathcal{C} | \mathcal{B} \xRightarrow{T} \mathcal{C} | \mathcal{A}$. Assume the premise in terms of Definition E14:

$$\forall w'_0 (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

Let $\bar{w}$ belong to $\phi_T^w(\mathcal{R}(\mathcal{C} | \mathcal{A})(w_0))$. The semantics for quoting gives $\bar{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{A}))(w_0))$. An edge only exists in a composition if we have w_1 and w_2 such that

The hypothesis gives $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, which satisfies the premise for Lemma E30. Hence we know $\bar{w}_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$, and we have proven that

$$\forall w_0, \text{ (pink circle) } (\phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0)) \subseteq \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))) \quad \square$$

Theorem E6. $\vdash (\mathcal{C} \xRightarrow{S} \mathcal{B}) \wedge (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset (\mathcal{C} \xRightarrow{S \cup T} \mathcal{A})$. Apply Axiom E5 twice to the premises to get two relations restricted by $S \cup T$, then apply Axiom E1 to collapse them into the relation in the conclusion. $\square$

Result E7. $(B \xRightarrow{S} A) \wedge (B \xRightarrow{T} A) \not\vdash B \xRightarrow{S \cup T} A$. Figure A.2 gives a counterexample that justifies the result. The diagram in the figure models $B \xRightarrow{S} A$ and $B \xRightarrow{T} A$. The statement $B \xRightarrow{S \cup T}$, however, fails. Projecting the model under $S \cup T$ gives the original picture, since each world falls in a separate equivalence class. Notice that B says $\neg(s \wedge \neg t)$: that statement is true in both worlds B considers possible. But A does not believe it, since A can see the lower-left world, where the statement is false.

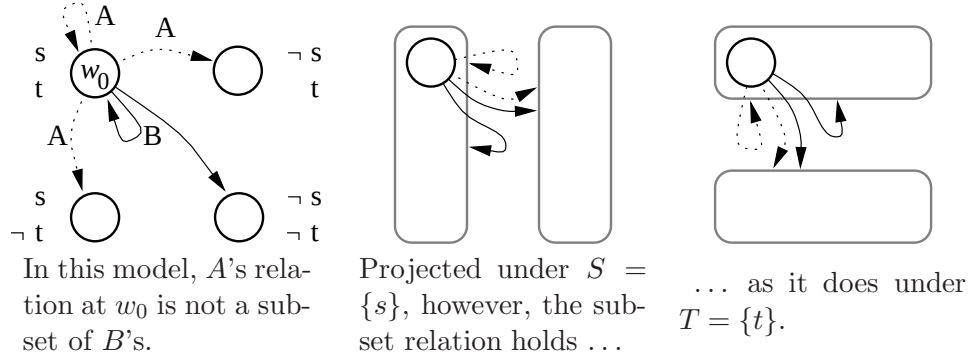


Figure A.2: A counterexample showing why two delegations for sets S and T do not imply a delegation for set $S \cup T$ (Result E7).

Why should this result be intuitive or desirable? Recall from Section 7.9 that the strength of $\xRightarrow{T}$ means that a delegation regarding T may imply a delegation regarding a larger set T^* that includes formulas constructed from the members of T . In our example, B speaks for A regarding formulas composed exclusively with the primitive s or the primitive t , but not regarding formulas combining the two. The closure of the restriction set $S \cup T$ includes formulas such as $\neg(s \wedge \neg t)$.

Axiom E8. $\vdash (\mathcal{B} \xRightarrow{T} \mathcal{A}) \supset \mathcal{C} | \mathcal{B} \xRightarrow{T} \mathcal{C} | \mathcal{A}$. Assume the premise in terms of Definition E14:

$$\forall w'_0 (\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)))$$

Let $\bar{w}$ belong to $\phi_T^w(\mathcal{R}(\mathcal{C} | \mathcal{A})(w_0))$. The semantics for quoting gives $\bar{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{A}))(w_0))$. An edge only exists in a composition if we have w_1 and w_2 such that

$\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{C})$ and $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{A})$; Definition E13 guarantees that we have such w_1, w_2 with $\bar{w} = \phi_T(w_2)$.

Since $w_2 \in \mathcal{R}(\mathcal{A})(w_1)$, we can use the assumption to show the existence of $w'_2 \in \mathcal{R}(\mathcal{B})(w_1)$ with $\phi_T(w'_2) = \phi_T(w_2) = \bar{w}$. That means that $\bar{w} \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_1))$, and hence $\bar{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{B}))(w_0))$. By the definition of quoting, we arrive at $\bar{w} \in \phi_T^w(\mathcal{R}(\mathcal{C}|\mathcal{B})(w_0))$, which proves the conclusion. $\square$

Result E9. $(\mathcal{B} \xRightarrow{T} \mathcal{A}) \not\vdash \mathcal{B}|\mathcal{C} \xRightarrow{T} \mathcal{A}|\mathcal{C}$. The model in Figure A.3 is a counterexample for $T = \{s\}$ that shows the result. Notice that $B \xRightarrow{T} A$: $\mathcal{R}(A)$'s only edge goes from w_0 to the equivalence class of worlds where s is true, and $\mathcal{R}(B)$ also has such an edge (the loop at w_0). When we compose the relations, however, we see that $B|C$ **says** s , but not $A|C$ **says** s . The equivalence classes of $\{C \text{ says } s\}$ are different than the equivalence classes of $\{s\}$.

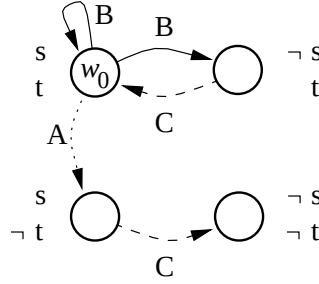


Figure A.3: A model that demonstrates Result E9.

Axiom E10. $\vdash \left(\mathcal{B} \xRightarrow{(T^*)^C} \mathcal{A} \right) \supset \left(\mathcal{B}|\mathcal{C} \xRightarrow{T} \mathcal{A}|\mathcal{C} \right)$. Inductively applying Axiom E25 and Axiom E24 shows as a theorem that $B \xRightarrow{T} A$ implies $B \xRightarrow{T^*} A$. Therefore, we may immediately replace the premise of this axiom with $B \xRightarrow{((T^*)^C)^*} A$, which follows by the theorem from the original premise. Herein we omit the parentheses for the postfix set operators $*$ and C , and simply write T^*C^* .

Hence we begin with the hypothesis that

$$\mathcal{R}(\mathcal{A}) \subseteq \phi_{T^*C^*}^+(\mathcal{R}(\mathcal{B}))$$

We are given some $w_0 \in W$ and the existence of $\bar{w}_2 \in \phi_T^w(\mathcal{R}(\mathcal{A}|\mathcal{C})(w_0))$. The set can be rewritten $\phi_T^w((\mathcal{R}(\mathcal{A}) \circ \mathcal{R}(\mathcal{C}))(w_0))$, so we know that there exist w_1 and w_2 , where

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \mathcal{R}(\mathcal{A}) \\ \langle w_1, w_2 \rangle &\in \mathcal{R}(\mathcal{C}) \\ \bar{w}_2 &= \phi_{T^*C^*}^w(w_2) \end{aligned}$$

The last expression means that for all $\sigma \in T$, $\sigma \in \bar{w}_2$ if and only if $w_2 \in \mathcal{E}(\sigma)$.

$\langle w_0, w_1 \rangle \in \mathcal{R}(\mathcal{C})$ and $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{A})$; Definition E13 guarantees that we have such w_1, w_2 with $\bar{w} = \phi_T(w_2)$.

Since $w_2 \in \mathcal{R}(\mathcal{A})(w_1)$, we can use the assumption to show the existence of $w'_2 \in \mathcal{R}(\mathcal{B})(w_1)$ with $\phi_T(w'_2) = \phi_T(w_2) = \bar{w}$. That means that $\bar{w} \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_1))$, and hence $\bar{w} \in \phi_T^w((\mathcal{R}(\mathcal{C}) \circ \mathcal{R}(\mathcal{B}))(w_0))$. By the definition of quoting, we arrive at $\bar{w} \in \phi_T^w(\mathcal{R}(\mathcal{C}|\mathcal{B})(w_0))$, which proves the conclusion. $\square$

Result E9. $(\mathcal{B} \xRightarrow{T} \mathcal{A}) \not\vdash \mathcal{B}|\mathcal{C} \xRightarrow{T} \mathcal{A}|\mathcal{C}$. The model in Figure A.3 is a counterexample for $T = \{s\}$ that shows the result. Notice that $B \xRightarrow{T} A$: $\mathcal{R}(\mathcal{A})$'s only edge goes from w_0 to the equivalence class of worlds where s is true, and $\mathcal{R}(\mathcal{B})$ also has such an edge (the loop at w_0). When we compose the relations, however, we see that $B|\mathcal{C}$ **says** s , but not $A|\mathcal{C}$ **says** s . The equivalence classes of $\{C \text{ says } s\}$ are different than the equivalence classes of $\{s\}$.

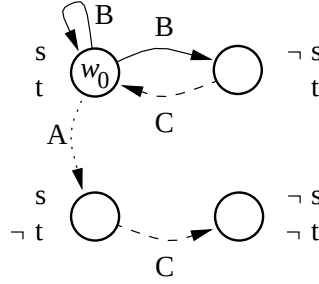


Figure A.3: A model that demonstrates Result E9.

Axiom E10. $\vdash \left(\mathcal{B} \xRightarrow{(T^*)^C} \mathcal{A} \right) \supset \left(\mathcal{B}|\mathcal{C} \xRightarrow{T} \mathcal{A}|\mathcal{C} \right)$. Inductively applying Axiom E25 and Axiom E24 shows as a theorem that $B \xRightarrow{T} A$ implies $B \xRightarrow{T^*} A$. Therefore, we may immediately replace the premise of this axiom with $B \xRightarrow{((T^*)^C)^*} A$, which follows by the theorem from the original premise. Herein we omit the parentheses for the postfix set operators $*$ and C , and simply write T^*C^* .

Hence we begin with the hypothesis that

$$\mathcal{R}(\mathcal{A}) \subseteq \phi_{T^*C^*}^+(\mathcal{R}(\mathcal{B}))$$

We are given some $w_0 \in W$ and the existence of $\bar{w}_2 \in \phi_T^w(\mathcal{R}(\mathcal{A}|\mathcal{C})(w_0))$. The set can be rewritten $\phi_T^w((\mathcal{R}(\mathcal{A}) \circ \mathcal{R}(\mathcal{C}))(w_0))$, so we know that there exist w_1 and w_2 , where

$$\begin{aligned} \langle w_0, w_1 \rangle &\in \mathcal{R}(\mathcal{A}) \\ \langle w_1, w_2 \rangle &\in \mathcal{R}(\mathcal{C}) \\ \bar{w}_2 &= \phi_{T^*C^*}^w(w_2) \end{aligned}$$

The last expression means that for all $\sigma \in T, \sigma \in \bar{w}_2$ if and only if $w_2 \in \mathcal{E}(\sigma)$.

Define the formula

$$\tau_2 = \bigwedge_{\sigma \in T} \left\{ \begin{array}{ll} \sigma & \text{if } \sigma \in \bar{w}_2 \\ \neg \sigma & \text{otherwise} \end{array} \right.$$

Intuitively, τ_2 is true at precisely those worlds that map to $\bar{w}_2$ under ϕ_T . We have constructed τ_2 such that $w_2 \in \mathcal{E}(\tau_2)$.

Since $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{C})$, we know $\mathcal{R}(\mathcal{C}) \not\subseteq \mathcal{E}(\neg \tau_2)$, and therefore $w_1 \notin \mathcal{E}(\mathcal{C} \text{ says } \neg \tau_2)$, and finally $w_1 \in \mathcal{E}(\neg \mathcal{C} \text{ says } \neg \tau_2)$. The propositional closure of T ensures that each conjunct of τ_2 , and thus τ_2 itself and $\neg \tau_2$, appear in T^* . The modal closure over “ $\mathcal{C}$ says” ensures that $(\mathcal{C} \text{ says } \neg \tau_2) \in T^*C$, and therefore $(\neg \mathcal{C} \text{ says } \neg \tau_2) \in T^*C^*$.

Now we may employ the hypothesis to show that there exists a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$ with $w'_1 \cong_{T^*C^*} w_1$. It follows that:

$$\begin{aligned} w'_1 &\in \mathcal{E}(\neg \mathcal{C} \text{ says } \neg \tau_2) \\ &= W - \mathcal{E}(\mathcal{C} \text{ says } \neg \tau_2) \\ &= W - \{w \mid \mathcal{R}(\mathcal{C})(w) \subseteq \mathcal{E}(\neg \tau_2)\} \\ &= \{w \mid \mathcal{R}(\mathcal{C})(w) \not\subseteq \mathcal{E}(\neg \tau_2)\} \\ &= \{w \mid \exists w'_2 \in \mathcal{R}(\mathcal{C})(w), w'_2 \notin \mathcal{E}(\neg \tau_2)\} \\ &= \{w \mid \exists w'_2 \in \mathcal{R}(\mathcal{C})(w), w'_2 \in \mathcal{E}(\tau_2)\} \end{aligned}$$

That is, we know there is a $w'_2 \in \mathcal{E}(\tau_2)$, with $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$.

With both $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ and $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$, we have $\langle w_0, w'_2 \rangle \in \mathcal{R}(\mathcal{B}) \circ \mathcal{R}(\mathcal{C}) = \mathcal{R}(\mathcal{B}|\mathcal{C})$. From the definition of τ_2 , we know that w'_2 is in $\mathcal{E}(\sigma)$ exactly when $\sigma \in \bar{w}_2$ for all $\sigma \in T$, so $\bar{w}_2 = \phi_T(w'_2)$. We have shown that $\bar{w}_2 \in \phi_T^w(\mathcal{R}(\mathcal{B}|\mathcal{C})(w_0))$, and therefore that given the hypothesis, the model supports $\mathcal{B}|\mathcal{C} \xrightarrow{T} \mathcal{A}|\mathcal{C}$. $\square$

Axiom E17. $\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset (\mathcal{B} \cdot N \Rightarrow \mathcal{A} \cdot N)$. This axiom follows from my brute-force semantics for names. Assume the premise:

$$\mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B})$$

We want to show that

$$\mathcal{R}(\mathcal{A} \cdot N) \subseteq \mathcal{R}(\mathcal{B} \cdot N),$$

which, of course, is trivial thanks to requirement (I) of Definition E21.

Theorem E18. $\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N)$. Since $(\mathcal{A} \wedge \mathcal{B}) \Rightarrow \mathcal{A}$, $(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow \mathcal{A} \cdot N$ (by Axiom E17, with $T = \mathcal{U}$). The same is true for $\mathcal{B}$, proving:

$$(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad \square$$

Define the formula

$$\tau_2 = \bigwedge_{\sigma \in T} \left\{ \begin{array}{ll} \sigma & \text{if } \sigma \in \bar{w}_2 \\ \neg \sigma & \text{otherwise} \end{array} \right.$$

Intuitively, τ_2 is true at precisely those worlds that map to $\bar{w}_2$ under ϕ_T . We have constructed τ_2 such that $w_2 \in \mathcal{E}(\tau_2)$.

Since $\langle w_1, w_2 \rangle \in \mathcal{R}(\mathcal{C})$, we know $\mathcal{R}(\mathcal{C}) \not\subseteq \mathcal{E}(\neg\tau_2)$, and therefore $w_1 \notin \mathcal{E}(\mathcal{C} \text{ says } \neg\tau_2)$, and finally $w_1 \in \mathcal{E}(\neg\mathcal{C} \text{ says } \neg\tau_2)$. The propositional closure of T ensures that each conjunct of τ_2 , and thus τ_2 itself and $\neg\tau_2$, appear in T^* . The modal closure over “ $\mathcal{C}$ says” ensures that $(\mathcal{C} \text{ says } \neg\tau_2) \in T^*C$, and therefore $(\neg\mathcal{C} \text{ says } \neg\tau_2) \in T^*C^*$.

Now we may employ the hypothesis to show that there exists a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$ with $w'_1 \cong_{T^*C^*} w_1$. It follows that:

$$\begin{aligned} w'_1 &\in \mathcal{E}(\neg\mathcal{C} \text{ says } \neg\tau_2) \\ &= W - \mathcal{E}(\mathcal{C} \text{ says } \neg\tau_2) \\ &= W - \{w \mid \mathcal{R}(\mathcal{C})(w) \subseteq \mathcal{E}(\neg\tau_2)\} \\ &= \{w \mid \mathcal{R}(\mathcal{C})(w) \not\subseteq \mathcal{E}(\neg\tau_2)\} \\ &= \{w \mid \exists w'_2 \in \mathcal{R}(\mathcal{C})(w), w'_2 \notin \mathcal{E}(\neg\tau_2)\} \\ &= \{w \mid \exists w'_2 \in \mathcal{R}(\mathcal{C})(w), w'_2 \in \mathcal{E}(\tau_2)\} \end{aligned}$$

That is, we know there is a $w'_2 \in \mathcal{E}(\tau_2)$, with $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$.

With both $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ and $\langle w'_1, w'_2 \rangle \in \mathcal{R}(\mathcal{C})$, we have $\langle w_0, w'_2 \rangle \in \mathcal{R}(\mathcal{B}) \circ \mathcal{R}(\mathcal{C}) = \mathcal{R}(\mathcal{B}|\mathcal{C})$. From the definition of τ_2 , we know that w'_2 is in $\mathcal{E}(\tau_2)$ exactly when $\sigma \in \bar{w}_2$ for all $\sigma \in T$, so $\bar{w}_2 = \phi_T(w'_2)$. We have shown that $\bar{w}_2 \in \phi_T^w(\mathcal{R}(\mathcal{B}|\mathcal{C})(w_0))$, and therefore that given the hypothesis, the model supports $\mathcal{B}|\mathcal{C} \xrightarrow{T} \mathcal{A}|\mathcal{C}$. $\square$

Axiom E17. $\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset (\mathcal{B} \cdot N \Rightarrow \mathcal{A} \cdot N)$. This axiom follows from my brute-force semantics for names. Assume the premise:

$$\mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B})$$

We want to show that

$$\mathcal{R}(\mathcal{A} \cdot N) \subseteq \mathcal{R}(\mathcal{B} \cdot N)$$

which, of course, is trivial thanks to requirement (I) of Definition E21.

Theorem E18. $\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N)$. Since $(\mathcal{A} \wedge \mathcal{B}) \Rightarrow \mathcal{A}$, $(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow \mathcal{A} \cdot N$ (by Axiom E17, with $T = \mathcal{U}$). The same is true for $\mathcal{B}$, proving:

$$(\mathcal{A} \wedge \mathcal{B}) \cdot N \Rightarrow (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \quad \square$$

Axiom E19. $\vdash (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \Rightarrow (\mathcal{A} \wedge \mathcal{B}) \cdot N$. Requirement (III) of Definition E21 exists to support this axiom. It says:

$$\mathcal{R}(\mathcal{A} \wedge \mathcal{B}) \cdot N \subseteq \mathcal{R}(\mathcal{A} \cdot N) \cup \mathcal{R}(\mathcal{B} \cdot N)$$

The right-hand side, by the semantics for $\wedge$, is equal to $\mathcal{R}((\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N))$, completing the proof.

Theorem E20. $\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N = (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N)$. Theorem E18 and Axiom E19 together show equality. $\square$

Axiom E23. $(\mathcal{B} \xRightarrow{\{\sigma, \tau\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A})$. Assume $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma') \not\subseteq \mathcal{R}(\mathcal{A}) \subseteq \mathcal{E}(\sigma')$ for $\sigma' \in \{\sigma, \tau\}$. Further, assume that $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma \wedge \tau)$. Using the semantics of $\wedge$, we can write $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)$, and hence $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma)$ and $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\tau)$. By the first assumption, we can replace $\mathcal{B}$ in both statements with $\mathcal{A}$, use the definition of $\cap$ and the semantics of $\wedge$, and conclude that $\mathcal{R}(\mathcal{A}) \subseteq \mathcal{E}(\sigma \wedge \tau)$, justifying the axiom. $\square$

Axiom E24. $(\mathcal{B} \xRightarrow{\{\sigma, \tau\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A})$. Let $T = \{\sigma, \tau\}$ and $T' = \{\sigma \wedge \tau\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)) \quad \forall w'_0 \in W$$

Second, assume we are given w_0 and $\bar{w}'_1$ such that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. We have the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$ with $\bar{w}'_1 = \phi_{T'}(w_1)$.

Let $\bar{w}_1 = \phi_T(w_1)$. By our first assumption, $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, so there is a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$ with $\bar{w}_1 = \phi_T(w'_1)$. We claim that $\phi_{T'}(w'_1) = \bar{w}'_1$, a claim supported by leaning on the definition of ϕ_T :

$$\begin{aligned} \sigma \wedge \tau \in \phi_{T'}(w'_1) &\equiv w'_1 \in \mathcal{E}(\sigma \wedge \tau) \\ &\equiv w'_1 \in \mathcal{E}(\sigma) \wedge w'_1 \in \mathcal{E}(\tau) \\ &\equiv \sigma \in \bar{w}_1 \wedge \tau \in \bar{w}_1 \\ &\equiv w_1 \in \mathcal{E}(\sigma) \wedge w_1 \in \mathcal{E}(\tau) \\ &\equiv w_1 \in \mathcal{E}(\sigma \wedge \tau) \\ &\equiv \sigma \wedge \tau \in \bar{w}_1 \end{aligned}$$

Since $\bar{w}'_1$ is either $T = \{\sigma \wedge \tau\}$ or $\emptyset$, we have shown the equality, and that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. Therefore the model supports $\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A}$. $\square$

Axiom E25. $(\mathcal{B} \xRightarrow{\{\sigma\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\neg \sigma\}} \mathcal{A})$. The structure of this proof parallels that of Axiom E24. Let $T = \{\sigma\}$ and $T' = \{\neg \sigma\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)) \quad \forall w'_0 \in W$$

Axiom E19. $\vdash (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N) \Rightarrow (\mathcal{A} \wedge \mathcal{B}) \cdot N$. Requirement (III) of Definition E21 exists to support this axiom. It says:

$$\mathcal{R}(\mathcal{A} \wedge \mathcal{B}) \cdot N \subseteq \mathcal{R}(\mathcal{A} \cdot N) \cup \mathcal{R}(\mathcal{B} \cdot N)$$

The right-hand side, by the semantics for $\wedge$, is equal to $\mathcal{R}((\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N))$, completing the proof.

Theorem E20. $\vdash (\mathcal{A} \wedge \mathcal{B}) \cdot N = (\mathcal{A} \cdot N) \wedge (\mathcal{B} \cdot N)$. Theorem E18 and Axiom E19 together show equality. $\square$

Axiom E23. $(\mathcal{B} \xRightarrow{\{\sigma, \tau\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A})$. Assume $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma)$ and $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\tau)$ for $\sigma' \in \{\sigma, \tau\}$. Further, assume that $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma \wedge \tau)$. Using the semantics of $\wedge$, we can write $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)$, and hence $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\sigma)$ and $\mathcal{R}(\mathcal{B}) \subseteq \mathcal{E}(\tau)$. By the first assumption, we can replace $\mathcal{B}$ in both statements with $\mathcal{A}$, use the definition of $\cap$ and the semantics of $\wedge$, and conclude that $\mathcal{R}(\mathcal{A}) \subseteq \mathcal{E}(\sigma \wedge \tau)$, justifying the axiom. $\square$

Axiom E24. $(\mathcal{B} \xRightarrow{\{\sigma, \tau\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A})$. Let $T = \{\sigma, \tau\}$ and $T' = \{\sigma \wedge \tau\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)) \quad \forall w'_0 \in W$$

Second, assume we are given w_0 and $\bar{w}'_1$ such that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. We have the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$ with $\bar{w}'_1 = \phi_{T'}(w_1)$.

Let $\bar{w}_1 = \phi_T(w_1)$. By our first assumption, $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$, so there is a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$ with $\bar{w}_1 = \phi_T(w'_1)$. We claim that $\phi_{T'}(w'_1) = \bar{w}'_1$, a claim supported by leaning on the definition of ϕ_T :

$$\begin{aligned} \sigma \wedge \tau \in \phi_{T'}(w'_1) &\equiv w'_1 \in \mathcal{E}(\sigma \wedge \tau) \\ &\equiv w'_1 \in \mathcal{E}(\sigma) \wedge w'_1 \in \mathcal{E}(\tau) \\ &\equiv \sigma \in \bar{w}_1 \wedge \tau \in \bar{w}_1 \\ &\equiv w_1 \in \mathcal{E}(\sigma) \wedge w_1 \in \mathcal{E}(\tau) \\ &\equiv w_1 \in \mathcal{E}(\sigma \wedge \tau) \\ &\equiv \sigma \wedge \tau \in \bar{w}'_1 \end{aligned}$$

Since $\bar{w}'_1$ is either $T = \{\sigma \wedge \tau\}$ or $\emptyset$, we have shown the equality, and that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{B})(w_0))$. Therefore the model supports $\mathcal{B} \xRightarrow{\{\sigma \wedge \tau\}} \mathcal{A}$. $\square$

Axiom E25. $(\mathcal{B} \xRightarrow{\{\sigma\}} \mathcal{A}) \supset (\mathcal{B} \xRightarrow{\{\neg \sigma\}} \mathcal{A})$. The structure of this proof parallels that of Axiom E24. Let $T = \{\sigma\}$ and $T' = \{\neg \sigma\}$. Assume first that:

$$\phi_T^w(\mathcal{R}(\mathcal{A})(w'_0)) \subseteq \phi_T^w(\mathcal{R}(\mathcal{B})(w'_0)) \quad \forall w'_0 \in W$$

Second, assume we are given w_0 and $\bar{w}'_1$ such that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. That implies the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\bar{w}'_1 = \phi_{T'}(w_1)$. By the definition of $\phi_{T'}$ we know $w_1 \in \mathcal{E}(\neg\sigma)$ if and only if $\neg\sigma \in \bar{w}'_1$. Using the semantics of $\neg$, we can rewrite that expression as

$$w_1 \in \mathcal{E}(\sigma) \text{ iff } \neg\sigma \notin \bar{w}'_1$$

Define

$$\bar{w}_1 = \begin{cases} T & \text{if } \bar{w}'_1 = \emptyset \\ \emptyset & \text{otherwise } (\bar{w}'_1 = T') \end{cases}$$

Clearly $\sigma \in \bar{w}_1$ if and only if $\neg\sigma \notin \bar{w}'_1$. Now we can write

$$w_1 \in \mathcal{E}(\sigma) \text{ iff } \sigma \in \bar{w}_1$$

This expression satisfies the definition of ϕ_T , so we have $\phi_T(w_1) = \bar{w}_1$. Because $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, we know $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0))$.

Using the first assumption, we have $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$. Using arguments analogous to those above, we have the existence of a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$, and by the definition of ϕ_T , we can show that $\bar{w}'_1$ is in $\phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$ as well. The model supports $\mathcal{B} \xRightarrow{\sigma} \mathcal{A}$. $\square$

A.5 Relationships among the restricted relations

In each of the counterexamples below, assume $T = \{s\}$.

$\xRightarrow{T}$ **is not stronger than** $\xRightarrow{T}$. The subset relation in the projected model $\bar{\mathcal{M}}$ of $\xRightarrow{T}$ holds with the possible exception of the single world $\bar{w}_T = T$ that represents the equivalence class of worlds in $\mathcal{M}$ in which all statements in T hold. Clearly ϕ_T takes every member of $\cap_{\sigma \in T} \mathcal{E}(\sigma)$ to that representative. The counterexample illustrated in Figure A.4 highlights this exception.

$\xRightarrow{T}$ **is not stronger than** $\xRightarrow{T}$. Although we just showed that $\xRightarrow{T}$ is not quite stronger than $\xRightarrow{T}$, it certainly seems almost so. Indeed, it is easy to construct an example that shows that the mighty relation does not follow from the basic speaks-for-regarding relation. See Figure A.5.

$\xRightarrow{T}$ **implies** $\xRightarrow{T}$. Assume $\mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B}))$. We prove by contradiction that $\mathcal{B} \xRightarrow{T} \mathcal{A}$. To establish a contradiction, we assume there is a statement $\sigma \in T$ and a world w_0 where $\mathcal{B}$ says σ but not $\mathcal{A}$ says σ . That is, $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$ but $\mathcal{R}(\mathcal{A})(w_0) \not\subseteq \mathcal{E}(\sigma)$. The latter means that there is a world $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, but $w_1 \notin \mathcal{E}(\sigma)$.

We can push $\langle w_0, w_1 \rangle$ through our original assumption to find a w'_1 such that $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ and $w'_1 \cong_T w_1$. Definition E11 tells us that $w'_1 \notin \mathcal{E}(\sigma)$, which means $\mathcal{R}(\mathcal{B})(w_0) \not\subseteq \mathcal{E}(\sigma)$.

Second, assume we are given w_0 and $\bar{w}'_1$ such that $\bar{w}'_1 \in \phi_{T'}^w(\mathcal{R}(\mathcal{A})(w_0))$. That implies the existence of a $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, with $\bar{w}'_1 = \phi_{T'}(w_1)$. By the definition of $\phi_{T'}$ we know $w_1 \in \mathcal{E}(\neg\alpha)$ if and only if $\neg\sigma \in \bar{w}'_1$. Using the semantics of $\neg$, we can rewrite that expression as

$$w_1 \in \mathcal{E}(\alpha) \text{ iff } \neg\sigma \notin \bar{w}'_1$$

Define

$$\bar{w}_1 = \begin{cases} T & \text{if } \bar{w}'_1 = \emptyset \\ \emptyset & \text{otherwise } (\bar{w}'_1 = T') \end{cases}$$

Clearly $\sigma \in \bar{w}_1$ if and only if $\neg\sigma \notin \bar{w}'_1$. Now we can write

$$w_1 \in \mathcal{E}(\alpha) \text{ iff } \sigma \in \bar{w}_1$$

This expression satisfies the definition of ϕ_T , so we have $\phi_T(w_1) = \bar{w}_1$. Because $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, we know $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{A})(w_0))$.

Using the first assumption, we have $\bar{w}_1 \in \phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$. Using arguments analogous to those above, we have the existence of a $w'_1 \in \mathcal{R}(\mathcal{B})(w_0)$, and by the definition of ϕ_T , we can show that $\bar{w}'_1$ is in $\phi_T^w(\mathcal{R}(\mathcal{B})(w_0))$ as well. The model supports $\mathcal{B} \xRightarrow{\sigma} \mathcal{A}$. $\square$

A.5 Relationships among the restricted relations

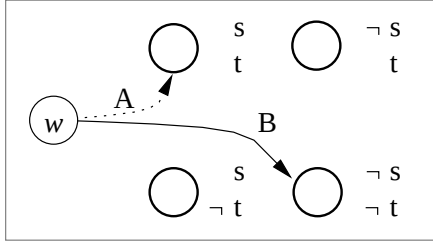
In each of the counterexamples below, assume $T = \{s\}$.

$\xRightarrow{T}$ **is not stronger than** $\xRightarrow{T}$. The subset relation in the projected model $\bar{\mathcal{M}}$ of $\xRightarrow{T}$ holds with the possible exception of the single world $\bar{w}_T = T$ that represents the equivalence class of worlds in $\mathcal{M}$ in which all statements in T hold. Clearly ϕ_T takes every member of $\cap_{\sigma \in T} \mathcal{E}(\sigma)$ to that representative. The counterexample illustrated in Figure A.4 highlights this exception.

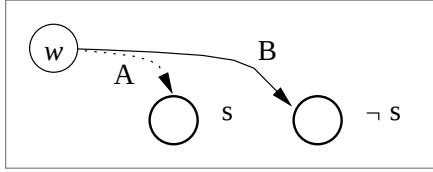
$\xRightarrow{T}$ **is not stronger than** $\xRightarrow{T}$. Although we just showed that $\xRightarrow{T}$ is not quite stronger than $\xRightarrow{T}$, it certainly seems almost so. Indeed, it is easy to construct an example that shows that the mighty relation does not follow from the basic speaks-for-regarding relation. See Figure A.5.

$\xRightarrow{T}$ **implies** $\xRightarrow{T}$. Assume $\mathcal{R}(\mathcal{A}) \subseteq \phi_T^+(\mathcal{R}(\mathcal{B}))$. We prove by contradiction that $\mathcal{B} \xRightarrow{T} \mathcal{A}$. To establish a contradiction, we assume there is a statement $\sigma \in T$ and a world w_0 where $\mathcal{B}$ says σ but not $\mathcal{A}$ says σ . That is, $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$ but $\mathcal{R}(\mathcal{A})(w_0) \not\subseteq \mathcal{E}(\sigma)$. The latter means that there is a world $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$, but $w_1 \notin \mathcal{E}(\sigma)$.

We can push $\langle w_0, w_1 \rangle$ through our original assumption to find a w'_1 such that $\langle w_0, w'_1 \rangle \in \mathcal{R}(\mathcal{B})$ and $w'_1 \cong_T w_1$. Definition E11 tells us that $w'_1 \notin \mathcal{E}(\sigma)$, which means $\mathcal{R}(\mathcal{B})(w_0) \not\subseteq \mathcal{E}(\sigma)$.



The set $\bigcap_{s \in T} \mathcal{E}(s)$ is the left pair of worlds (where s is true); the only edge belonging to $\mathcal{R}(A)$ terminates in one of those worlds. Therefore, in this model, $\mathcal{R}(A)(w) - \bigcap_{s \in T} \mathcal{E}(s) \subseteq \mathcal{R}(B)(w)$, and we conclude that $B \stackrel{T}{\Rightarrow} A$.



The mapping ϕ_T that reduces the worlds above to equivalence classes modulo statements in T will make this model $\mathcal{M}'$. $\phi_T^w(\mathcal{R}(A))$ includes an edge to the equivalence class labeled s , but $\phi_T^w(\mathcal{R}(B)(w))$ does not. Therefore, $B \not\stackrel{T}{\Rightarrow} A$.

Figure A.4: A counterexample that shows $B \stackrel{T}{\Rightarrow} A$ does not imply $B \stackrel{T}{\Rightarrow} A$.

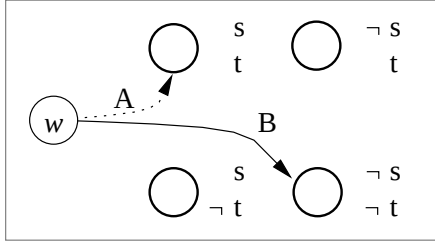
which contradicts our second assumption. We may conclude that for all $w_0 \in W$ and $\sigma \in T$, $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$ implies $\mathcal{R}(\mathcal{A})(w_0) \subseteq \mathcal{E}(\sigma)$. $\square$

$\stackrel{T}{\Rightarrow}$ implies $\stackrel{T}{\rightarrow}$. We assume

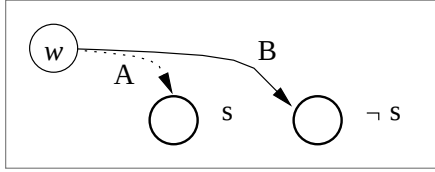
$$\mathcal{R}(\mathcal{A})(w_0) - \bigcap_{\tau \in T} \mathcal{E}(\tau) \subseteq \mathcal{R}(\mathcal{B})(w_0)$$

and that $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$. From the first assumption, any world $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$ is either in $\mathcal{E}(\sigma)$ (let $\tau = \sigma$) or in $\mathcal{R}(\mathcal{B})(w_0)$. The former case trivially guarantees $w_1 \in \mathcal{E}(\sigma)$, and the latter case does so by the second assumption. We conclude that $\mathcal{R}(\mathcal{A})(w_0) \subseteq \mathcal{E}(\sigma)$. $\square$

$\stackrel{T}{\rightarrow}$ is weaker than $\stackrel{T}{\Rightarrow}$ and $\stackrel{T}{\rightarrow}$ is weaker than $\stackrel{T}{\Rightarrow}$. See Figure A.6 for counterexamples that illustrate these relationships.



The set $\bigcap_{s \in T} \mathcal{E}(s)$ is the left pair of worlds (where s is true); the only edge belonging to $\mathcal{R}(A)$ terminates in one of those worlds. Therefore, in this model, $\mathcal{R}(A)(w) - \bigcap_{s \in T} \mathcal{E}(s) \subseteq \mathcal{R}(B)(w)$, and we conclude that $B \stackrel{T}{\Rightarrow} A$.



The mapping ϕ_T that reduces the worlds above to equivalence classes modulo statements in T will make this model $\mathcal{M}'$. $\phi_T^w(\mathcal{R}(A))$ includes an edge to the equivalence class labeled s , but $\phi_T^w(\mathcal{R}(B)(w))$ does not. Therefore, $B \not\stackrel{T}{\Rightarrow} A$.

Figure A.4: A counterexample that shows $B \stackrel{T}{\Rightarrow} A$ does not imply $B \stackrel{T}{\Rightarrow} A$.

which contradicts our second assumption. We may conclude that for all $w_0 \in W$ and $\sigma \in T$, $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$ implies $\mathcal{R}(\mathcal{A})(w_0) \subseteq \mathcal{E}(\sigma)$. $\square$

$\stackrel{T}{\Rightarrow}$ implies $\stackrel{T}{\rightarrow}$. We assume

$$\mathcal{R}(\mathcal{A})(w_0) - \bigcap_{\tau \in T} \mathcal{E}(\tau) \subseteq \mathcal{R}(\mathcal{B})(w_0)$$

and that $\mathcal{R}(\mathcal{B})(w_0) \subseteq \mathcal{E}(\sigma)$. From the first assumption, any world $w_1 \in \mathcal{R}(\mathcal{A})(w_0)$ is either in $\mathcal{E}(\sigma)$ (let $\tau = \sigma$) or in $\mathcal{R}(\mathcal{B})(w_0)$. The former case trivially guarantees $w_1 \in \mathcal{E}(\sigma)$, and the latter case does so by the second assumption. We conclude that $\mathcal{R}(\mathcal{A})(w_0) \subseteq \mathcal{E}(\sigma)$. $\square$

$\stackrel{T}{\rightarrow}$ is weaker than $\stackrel{T}{\Rightarrow}$ and $\stackrel{T}{\rightarrow}$ is weaker than $\stackrel{T}{\Rightarrow}$. See Figure A.6 for counterexamples that illustrate these relationships.

Appendix B

Review: the logic of belief

The Sicilian smiled and stared at the wine goblets. “Now a great fool,” he began, “would place the wine in his own goblet, because he would know that only another great fool would reach first for what he was given. I am clearly not a great fool, so I will clearly not reach for your wine.”

“That’s your final choice?”

“No. Because you knew I was not a great fool, so you would know that I would never fall for such a trick. You would count on it. So I will clearly not reach for mine either.” [Gol73, p. 157]

The Sicilian’s great effort went into reasoning about the beliefs of his opponent, including his opponent’s beliefs about his own beliefs, and so on. His watertight reasoning is an example of modal logic, the logic of belief. One way to reason about permissions and sharing is to reason about who believes what. Let us call participants in a distributed system *agents*, and the symbols that represent agents in logical expressions *principals*. Principals can also represent sets of agents, or one agent quoting another; these are called *compound principals*, and I discuss them in Appendix Section B.1. If Alice believes everything Bob believes (that is, Alice trusts Bob in every matter), then if Bob believes it is good to read a given file, Alice must believe the same. In this appendix, I develop a model for reasoning about logic in the presence of belief.

We begin with propositional logic. Assume there is a set of primitive (uninterpreted, independent) statements Σ .¹ For our purposes of access control, we consider primitive statements such as “it is good to write to file X.” This interpretation turns an imperative command into a declarative proposition. The primitive statements may be connected with *and* ($\wedge$) and *not* ($\neg$) to form arbitrary formulas. The *or* ($\vee$) and *implies* ($\supset$) operators are abbreviations for longer formulas made of $\wedge$ and $\neg$.

¹Figure 4.1 provides a table of sets and variable notation used in this paper.

Appendix B

Review: the logic of belief

The Sicilian smiled and stared at the wine goblets. “Now a great fool,” he began, “would place the wine in his own goblet, because he would know that only another great fool would reach first for what he was given. I am clearly not a great fool, so I will clearly not reach for your wine.”

“That’s your final choice?”

“No. Because you knew I was not a great fool, so you would know that I would never fall for such a trick. You would count on it. So I will clearly not reach for mine either.” [Gol73, p. 157]

The Sicilian’s great effort went into reasoning about the beliefs of his opponent, including his opponent’s beliefs about his own beliefs, and so on. His watertight reasoning is an example of modal logic, the logic of belief. One way to reason about permissions and sharing is to reason about who believes what. Let us call participants in a distributed system *agents*, and the symbols that represent agents in logical expressions *principals*. Principals can also represent sets of agents, or one agent quoting another; these are called *compound principals*, and I discuss them in Section B.1. If Alice believes everything Bob believes (that is, Alice trusts Bob in every matter), then if Bob believes it is good to read a given file, Alice must believe the same. In this appendix, I develop a model for reasoning about logic in the presence of belief.

We begin with propositional logic. Assume there is a set of primitive (uninterpreted, independent) statements Σ .¹ For our purposes of access control, we consider primitive statements such as “it is good to write to file X.” This interpretation turns an imperative command into a declarative proposition. The primitive statements may be connected with *and* ($\wedge$) and *not* ($\neg$) to form arbitrary formulas. The *or* ($\vee$) and *implies* ($\supset$) operators are abbreviations for longer formulas made of $\wedge$ and $\neg$.

¹Figure 4.1 provides a table of sets and variable notation used in this dissertation.

Next we introduce a *modal* operator **believes**.² If σ is a formula and principal A represents agent Alice, A **believes** σ is a formula that can be read “Alice believes σ is true.” In time, we will introduce multiple **believes** operators, one per principal. For now, we build a *model* that helps us understand which formulas A believes; that is, for which σ do we have A **believes** σ ?

To model this logic, we build a *Kripke structure*. A Kripke structure is a tuple of sets $\mathcal{M} = \langle W, I, J \rangle$. The members of set W represent *possible worlds*. The function I maps a primitive proposition (s) to the set of worlds where it is true, and the function J maps a principal to a relation on worlds in W . Together, I and J determine the truth value of every formula in every world in W ; we discuss them in more detail shortly.

First, some intuition: A principal Alice (A) living in world w_0 considers some other set of worlds possible. If a formula σ is true in each of those other worlds, then A (in w_0) believes the formula. Since each world is a parallel universe, principal A exists in every world. In any given world, A believes different propositions depending on which worlds the principal considers possible from the given world. An interesting fact about possible worlds is that the set of worlds A considers possible captures what she does not know: if a statement σ appears in one possible world and $\neg\sigma$ appears in another, then A knows neither σ nor $\neg\sigma$. As far as she is concerned, σ could go either way, because A cannot tell in which of the possible worlds she actually is.

When we write $\mathcal{M}, w_0 \models \sigma$ (pronounced “ $\mathcal{M}$ at w_0 models σ ”), we mean that in model $\mathcal{M}$ at world w_0 , the formula σ is true.³ The mapping I tells us immediately about the truth of primitive propositions at different worlds, but we wish to determine the truth of arbitrary statements σ , including propositional connectives and our modal operators ($\sigma \equiv A$ **believes** τ). I illustrate with an example structure, shown in Figure B.1.

The model contains three primitive statements, a , b , and p . The statement a means that our agent Alice (A) is in the produce department of a grocery store. Its negation, $\neg a$, means that Alice is in the meat department (it’s a small store). The b primitive means that the store’s bananas are yellow, and the p primitive means that the store’s pork is fresh.

Recall the three parts of a model, $\langle W, I, J \rangle$. W is the set of possible worlds; in our case, since there are three primitive statements, there are at most eight: $W = \{w_0, w_1, \dots, w_7\}$. I is a relation that defines which primitive statements are true at which worlds. In our example, $I(b) = \{w_0, w_1, w_4, w_5\}$, since the bananas are only yellow in those four worlds. Finally, J is a function that maps principals to relations. Because we have only one principal (Alice), J has only one mapping, written $J(A)$. The relation $J(A)$ is depicted with arrows in the diagram. For example, $\langle w_0, w_1 \rangle \in J(A)$; that is, when the actual world is w_0 , w_1 is a world Alice considers possible. In our example, it happens that Alice considers two worlds possible from each world.

Assume for a moment that the actual world is in fact w_0 : Alice is in the produce

²In conventional modal logic, A **believes** σ is written $\Box_A \sigma$.

³Abadi typically distinguishes the world w_0 that defines ground truth in a model itself, so that a model tuple becomes $\langle W, w_0, I, J \rangle$. I use the same notation in Section 4.2.

Next we introduce a *modal* operator **believes**.² If σ is a formula and principal A represents agent Alice, A **believes** σ is a formula that can be read “Alice believes σ is true.” In time, we will introduce multiple **believes** operators, one per principal. For now, we build a *model* that helps us understand which formulas A believes; that is, for which σ do we have A **believes** σ ?

To model this logic, we build a *Kripke structure*. A Kripke structure is a tuple of sets $\mathcal{M} = \langle W, I, J \rangle$. The members of set W represent *possible worlds*. The function I maps a primitive proposition (s) to the set of worlds where it is true, and the function J maps a principal to a relation on worlds in W . Together, I and J determine the truth value of every formula in every world in W ; we discuss them in more detail shortly.

First, some intuition: A principal Alice (A) living in world w_0 considers some other set of worlds possible. If a formula σ is true in each of those other worlds, then A (in w_0) believes the formula. Since each world is a parallel universe, principal A exists in every world. In any given world, A believes different propositions depending on which worlds the principal considers possible from the given world. An interesting fact about possible worlds is that the set of worlds A considers possible captures what she does not know: if a statement σ appears in one possible world and $\neg\sigma$ appears in another, then A knows neither σ nor $\neg\sigma$. As far as she is concerned, σ could go either way, because A cannot tell in which of the possible worlds she actually is.

When we write $\mathcal{M}, w_0 \models \sigma$ (pronounced “ $\mathcal{M}$ at w_0 models σ ”), we mean that in model $\mathcal{M}$ at world w_0 , the formula σ is true.³ The mapping I tells us immediately about the truth of primitive propositions at different worlds, but we wish to determine the truth of arbitrary statements σ including propositional connectives and our modal operators ($\sigma \equiv A$ **believes** τ). I illustrate with an example structure, shown in Figure B.1.

The model contains three primitive statements, a , b , and p . The statement a means that our agent Alice (A) is in the produce department of a grocery store. Its negation, $\neg a$, means that Alice is in the meat department (it’s a small store). The b primitive means that the store’s bananas are yellow, and the p primitive means that the store’s pork is fresh.

Recall the three parts of a model, $\langle W, I, J \rangle$. W is the set of possible worlds; in our case, since there are three primitive statements, there are at most eight: $W = \{w_0, w_1, \dots, w_7\}$. I is a relation that defines which primitive statements are true at which worlds. In our example, $I(b) = \{w_0, w_1, w_4, w_5\}$, since the bananas are only yellow in those four worlds. Finally, J is a function that maps principals to relations. Because we have only one principal (Alice), J has only one mapping, written $J(A)$. The relation $J(A)$ is depicted with arrows in the diagram. For example, $\langle w_0, w_1 \rangle \in J(A)$; that is, when the actual world is w_0 , w_1 is a world Alice considers possible. In our example, it happens that Alice considers two worlds possible from each world.

Assume for a moment that the actual world is in fact w_0 : Alice is in the produce

²In conventional modal logic, A **believes** σ is written $\Box_A \sigma$.

³Abadi typically distinguishes the world w_0 that defines ground truth in a model itself, so that a model tuple becomes $\langle W, w_0, I, J \rangle$. I use the same notation in Section 4.2.

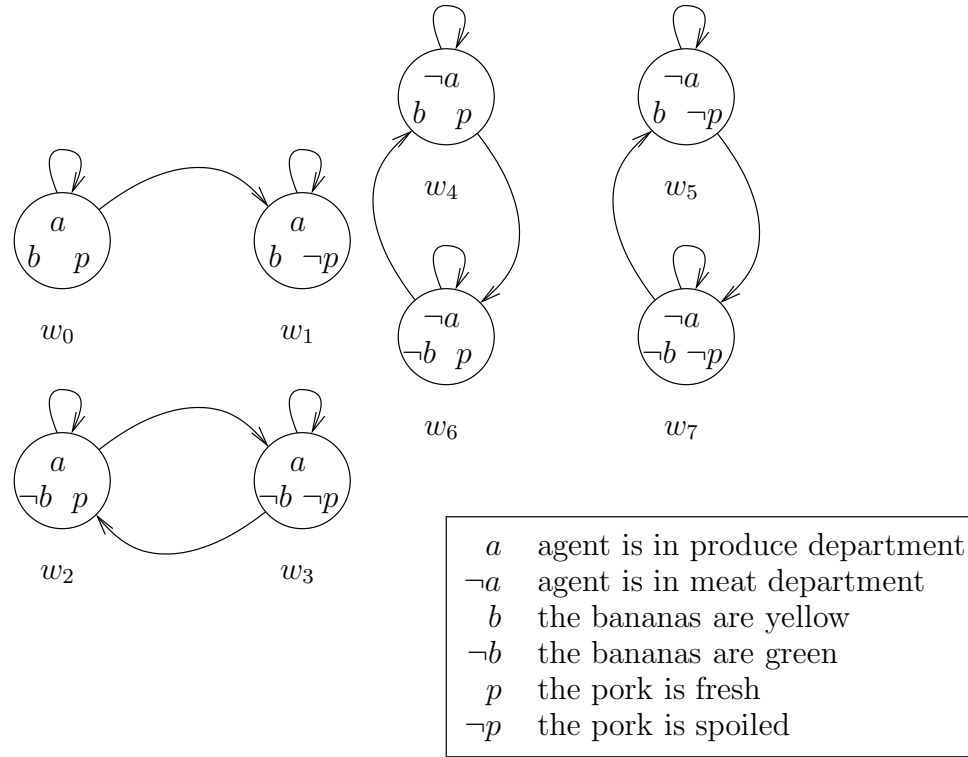


Figure B.1: A model of eight worlds (circles), illustrating the relationship between the accessibility relation for A (arrows) and the modal operator (A **believes**). At world w_0 , A **believes** b , but $\neg(A$ **believes** p), and $\neg(A$ **believes** $\neg p$).

department, the bananas are yellow and the pork is fresh. If Alice were omniscient, she would consider only w_0 possible, for that is indeed the state of things. Alice, however, is merely a shopper. She cannot see from the produce department what is going on in the meat department, and thus she cannot tell if the pork is fresh. She must also consider possible world w_1 , where the pork is spoiled. She knows for certain her own location, though, so she can ignore worlds $w_4 \cdots w_7$. Because she is in the produce department and can see the bananas, she can also ignore worlds w_2 and w_3 in which the bananas are green.

I have explained the two arrows emanating from world w_0 . These two, with the other arrows in the diagram, constitute the relation $J(A)$. Having arrows leaving different worlds provides information about what Alice *would* believe if things were different. For example, if the actual world were w_1 (the pork is in fact spoiled), Alice considers just the same worlds w_0 and w_1 possible, and for the same reasons.

Now that we have the intuition behind the Kripke structure, we can formally define when various statements are true. Primitive propositions are easy: the casual definition of I above becomes:

$$\mathcal{M}, w_0 \models s \quad \text{when } w_0 \in I(s)$$

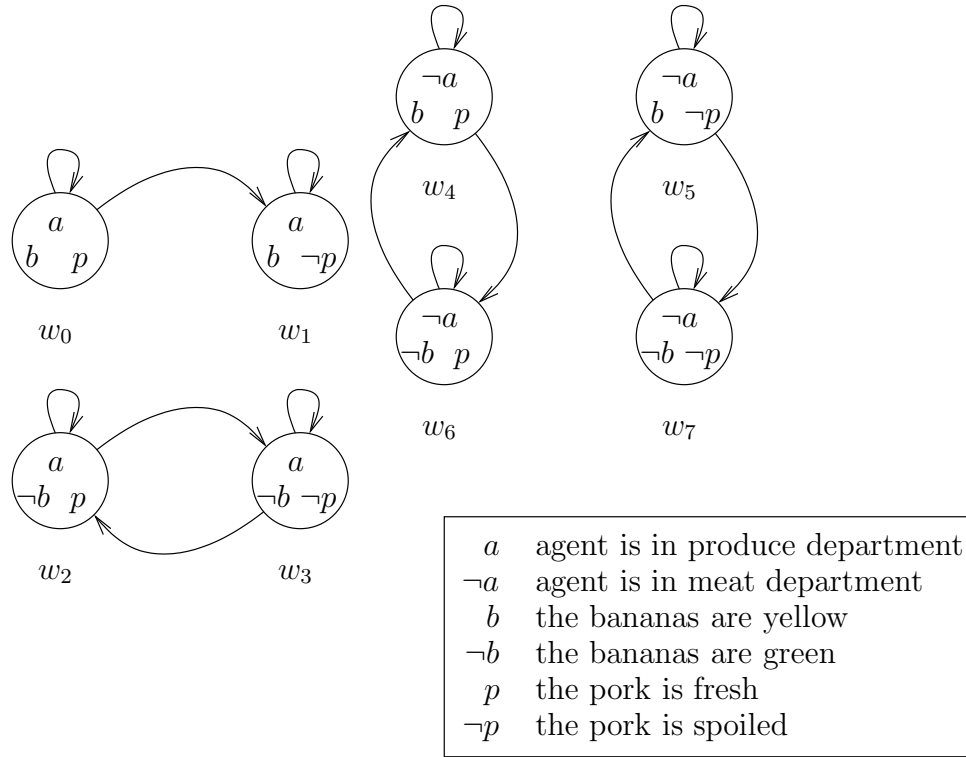


Figure B.1: A model of eight worlds (circles), illustrating the relationship between the accessibility relation for A (arrows) and the modal operator (A **believes**). At world w_0 , A **believes** b , but $\neg(A$ **believes** p), and $\neg(A$ **believes** $\neg p$).

department, the bananas are yellow and the pork is fresh. If Alice were omniscient, she would consider only w_0 possible, for that is indeed the state of things. Alice, however, is merely a shopper. She cannot see from the produce department what is going on in the meat department, and thus she cannot tell if the pork is fresh. She must also consider possible world w_1 , where the pork is spoiled. She knows for certain her own location, though, so she can ignore worlds $w_4 \cdots w_7$. Because she is in the produce department and can see the bananas, she can also ignore worlds w_2 and w_3 in which the bananas are green.

I have explained the two arrows emanating from world w_0 . These two, with the other arrows in the diagram, constitute the relation $J(A)$. The arrows leaving worlds other than w_0 provide information about what Alice *would* believe if things were different. For example, if the actual world were w_1 (the pork is in fact spoiled), Alice considers just the same worlds w_0 and w_1 possible, and for the same reasons.

Now that we have the intuition behind the Kripke structure, we can formally define when various statements are true. Primitive propositions are easy: the casual definition of I above becomes:

$$\mathcal{M}, w_0 \models s \quad \text{when } w_0 \in I(s)$$

This definition can be read “Statement s is true at world w_0 in model M when w_0 is in the set $I(s)$.”

What about formulas constructed from the propositional connectives $\wedge$ and $\neg$? The truth of some complex formula σ in a world is completely determined by the truth of its primitive propositions, which the model defines by the mapping I . So we can formally define an *extension* function $\mathcal{E}$ to extend the definition of I to arbitrary formulas. $\mathcal{E}$ is defined recursively starting with I , and extends as you would expect for the propositional connectives:

$$\begin{aligned}\mathcal{E}(s) &= I(s) \\ \mathcal{E}(\neg\sigma) &= W - \mathcal{E}(\sigma) \\ \mathcal{E}(\sigma \wedge \tau) &= \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)\end{aligned}$$

Not surprisingly, $\neg\sigma$ holds in exactly those worlds where σ does not, and $\sigma \wedge \tau$ holds in exactly those worlds where both subformulas hold. Take a look at the example structure and convince yourself that $\mathcal{E}(b \wedge \neg p) = \{w_1, w_5\}$.

We embarked on this journey to discover when Alice believes various statements, so we need to find out when the model supports formulas including our modal belief operator. The natural intuition is that Alice should believe a statement whenever it is true in *every* world Alice considers possible. To recall our example, b is true (the bananas are yellow) in every world Alice considers possible from w_0 , so $\mathcal{M}, w_0 \models A \text{ believes } b$. But because Alice considers w_0 and w_1 possible, she considers both p and $\neg p$ possible; and so she can believe neither; hence we have $\neg(A \text{ believes } p)$ and $\neg(A \text{ believes } \neg p)$ at world w_0 . (You can think of this situation as representing Alice’s “silence” on the matter of p . Even though Alice asserts neither p nor $\neg p$, every formula is assigned a truth value. It is just that both $A \text{ believes } p$ and $A \text{ believes } \neg p$ are false.)

With this intuition, we fill out the definition of $\mathcal{E}$ to mention formulas containing our modal operator $A \text{ believes}$:

$$\mathcal{E}(A \text{ believes } \sigma) = \{w \mid J(A)(w) \subseteq \mathcal{E}(\sigma)\}$$

$J(A)(w)$ denotes the set of worlds that A considers possible from w .⁴ So when σ is true in every one of these worlds (i.e., $J(A)(w) \subseteq \mathcal{E}(\sigma)$), then $A \text{ believes } \sigma$.

Of course, security is not very interesting in a world with only one agent. To introduce a second principal, we simply add a new relation $J(B)$ to our model. Now we can reason about what Bob believes ($B \text{ believes } \sigma$), and even about what Alice believes about what Bob believes ($A \text{ believes } B \text{ believes } \sigma$). (In our example, we could certainly discuss Alice’s beliefs about her own beliefs, but for our application to access control, that is uninteresting.)

⁴Formally, $J(A)(w) = \{w' \mid \langle w, w' \rangle \in J(A)\}$.

This definition can be read “Statement s is true at world w_0 in model M when w_0 is in the set $I(s)$.”

What about formulas constructed from the propositional connectives $\wedge$ and $\neg$? The truth of some complex formula σ in a world is completely determined by the truth of its primitive propositions, which the model defines by the mapping I . So we can formally define an *extension* function $\mathcal{E}$ to extend the definition of I to arbitrary formulas. $\mathcal{E}$ is defined recursively starting with I , and extends as you would expect for the propositional connectives:

$$\begin{aligned}\mathcal{E}(s) &= I(s) \\ \mathcal{E}(\neg\sigma) &= W - \mathcal{E}(\sigma) \\ \mathcal{E}(\sigma \wedge \tau) &= \mathcal{E}(\sigma) \cap \mathcal{E}(\tau)\end{aligned}$$

Not surprisingly, $\neg\sigma$ holds in exactly those worlds where σ does not, and $\sigma \wedge \tau$ holds in exactly those worlds where both subformulas hold. Take a look at the example structure and convince yourself that $\mathcal{E}(b \wedge \neg p) = \{w_1, w_5\}$.

We embarked on this journey to discover when Alice believes various statements, so we need to find out when the model supports formulas including our modal belief operator. The natural intuition is that Alice should believe a statement whenever it is true in *every* world Alice considers possible. To recall our example, b is true (the bananas are yellow) in every world Alice considers possible from w_0 , so $\mathcal{M}, w_0 \models A \text{ believes } b$. But because Alice considers w_0 and w_1 possible, she considers both p and $\neg p$ possible; and so she can believe neither; hence we have $\neg(A \text{ believes } p)$ and $\neg(A \text{ believes } \neg p)$ at world w_0 . (You can think of this situation as representing Alice’s “silence” on the matter of p . Even though Alice asserts neither p nor $\neg p$, every formula is assigned a truth value. It is just that both $A \text{ believes } p$ and $A \text{ believes } \neg p$ are false.)

With this intuition, we fill out the definition of $\mathcal{E}$ to mention formulas containing our modal operator $A \text{ believes}$:

$$\mathcal{E}(A \text{ believes } \sigma) = \{w \mid J(A)(w) \subseteq \mathcal{E}(\sigma)\}$$

$J(A)(w)$ denotes the set of worlds that A considers possible from w .⁴ So when σ is true in every one of these worlds (i.e., $J(A)(w) \subseteq \mathcal{E}(\sigma)$), then $A \text{ believes } \sigma$.

Of course, security is not very interesting in a world with only one agent. To introduce a second principal, we simply add a new relation $J(B)$ to our model. Now we can reason about what Bob believes ($B \text{ believes } \sigma$), and even about what Alice believes about what Bob believes ($A \text{ believes } B \text{ believes } \sigma$). (In our example, we could certainly discuss Alice’s beliefs about her own beliefs, but for our application to access control, that is uninteresting.)

⁴Formally, $J(A)(w) = \{w' \mid \langle w, w' \rangle \in J(A)\}$.

B.1 Compound principals

It is also possible to talk about *compound principals*. Lampson et al. define two operators on principals that can be used to make new compound principals. The first is fairly easy to describe: the principal $A \wedge B$ believes only things that both A and B believe. We can define a new possible-worlds relation for the compound principal in terms of the relations for A and B . To do this, we extend the mapping J to a new mapping $\mathcal{R}$ whose domain includes compound principals. Like the definition of $\mathcal{E}$, $\mathcal{R}$ is defined recursively starting with J :

$$\begin{aligned}\mathcal{R}(A) &= J(A) \\ &\quad \forall \text{ primitive principals } A \\ \mathcal{R}(A \wedge B) &= \mathcal{R}(A) \cup \mathcal{R}(B) \\ &\quad \forall \text{ arbitrary principals } A, B\end{aligned}$$

And $\mathcal{R}$ replaces J 's role in the definition of $\mathcal{E}$:

$$\mathcal{E}(A \text{ believes } \sigma) = \{w \mid \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma)\}$$

That set union operation is surprising! What's going on? Recall that the more worlds an agent considers possible, the less the agent believes. In our example structure, Alice could not believe p because she considered world w_1 possible, where p was false. Likewise, by taking the union of the relations for principals A and B to get the relation for the compound principal $A \wedge B$, we ensure that the compound principal is at least as ignorant as either of A or B . If A and B disagree on any statement σ , then $A \wedge B$ can see both worlds where σ is true and worlds where it is false, so $A \wedge B$ can have neither belief.

The second operator for forming compound principals is written $B|A$, and pronounced “ B quoting A .” (“Quoting” may seem an odd choice of words when talking about belief; however, when we translate our terminology into that of Lampson et al., it reads more naturally.) This principal captures B 's beliefs about A 's beliefs: $(B|A) \text{ believes } \sigma$ should be synonymous with $B \text{ believes } (A \text{ believes } \sigma)$.

The relation for the compound principal $B|A$ is the composition of the relations of B and A :

$$\mathcal{R}(B|A) = \mathcal{R}(B) \circ \mathcal{R}(A)$$

What is the intuition for using composition? Suppose we have $\mathcal{M}, w_0 \models B|A \text{ believes } \sigma$. At world w_0 , Bob believes Alice believes σ . That means that at every world Bob considers possible from w_0 ($\mathcal{R}(B)(w_0)$), Alice believes σ . But Alice only believes σ at those worlds if σ is true at every world Alice can see from those worlds:

$$\bigcup_{w' \in \mathcal{R}(B)(w_0)} \mathcal{R}(A)(w')$$

The composition $\mathcal{R}(B) \circ \mathcal{R}(A)$ relates w_0 to just this set. So $B|A \text{ believes } \sigma$ is true at w_0 exactly when σ is true in every world reachable from w_0 by the composited relation given above as $\mathcal{R}(B|A)$.

B.1 Compound principals

It is also possible to talk about *compound principals*. Lampson et al. define two operators on principals that can be used to make new compound principals. The first is fairly easy to describe: the principal $A \wedge B$ believes only things that both A and B believe. We can define a new possible-worlds relation for the compound principal in terms of the relations for A and B . To do this, we extend the mapping J to a new mapping $\mathcal{R}$ whose domain includes compound principals. Like the definition of $\mathcal{E}$, $\mathcal{R}$ is defined recursively starting with J :

$$\begin{aligned}\mathcal{R}(A) &= J(A) \\ &\quad \forall \text{ primitive principals } A \\ \mathcal{R}(A \wedge B) &= \mathcal{R}(A) \cup \mathcal{R}(B) \\ &\quad \forall \text{ arbitrary principals } A, B\end{aligned}$$

And $\mathcal{R}$ replaces J 's role in the definition of $\mathcal{E}$:

$$\mathcal{E}(A \text{ believes } \phi) = \{w \mid \mathcal{R}(A)(w) \subseteq \mathcal{E}(\phi)\}$$

That set union operation is surprising! What's going on? Recall that the more worlds an agent considers possible, the less the agent believes. In our example structure, Alice could not believe p because she considered world w_1 possible, where p was false. Likewise, by taking the union of the relations for principals A and B to get the relation for the compound principal $A \wedge B$, we ensure that the compound principal is at least as ignorant as either of A or B . If A and B disagree on any statement ϕ then $A \wedge B$ can see both worlds where ϕ is true and worlds where it is false, so $A \wedge B$ can have neither belief.

The second operator for forming compound principals is written $B|A$, and pronounced “ B quoting A .” (“Quoting” may seem an odd choice of words when talking about belief; however, when we translate our terminology into that of Lampson et al., it reads more naturally.) This principal captures B 's beliefs about A 's beliefs: $(B|A) \text{ believes } \phi$ should be synonymous with $B \text{ believes } (A \text{ believes } \phi)$.

The relation for the compound principal $B|A$ is the composition of the relations of B and A :

$$\mathcal{R}(B|A) = \mathcal{R}(B) \circ \mathcal{R}(A)$$

What is the intuition for using composition? Suppose we have $\mathcal{M}, w_0 \models B|A \text{ believes } \phi$. At world w_0 , Bob believes Alice believes ϕ . That means that at every world Bob considers possible from w_0 ($\mathcal{R}(B)(w_0)$), Alice believes ϕ . But Alice only believes ϕ at those worlds if ϕ is true at every world Alice can see from those worlds:

$$\bigcup_{w' \in \mathcal{R}(B)(w_0)} \mathcal{R}(A)(w')$$

The composition $\mathcal{R}(B) \circ \mathcal{R}(A)$ relates w_0 to just this set. So $B|A \text{ believes } \phi$ is true at w_0 exactly when ϕ is true in every world reachable from w_0 by the composited relation given above as $\mathcal{R}(B|A)$.

B.1.1 The nature of principal relations

Now that we have a formal structure for discussing the beliefs of principals, let us consider what kinds of beliefs are reasonable, and how one principal's beliefs should be related to another's.

Recall our example structure, where in any world, Alice was either ignorant (had no belief) about the pork or ignorant about the bananas. The first observation is that agents do not need to believe every true thing; statements about which they have neither a positive nor a negative belief represent a fact about which the agent is ignorant.

Furthermore, observe that Alice never believed anything false: in every world, if A believes σ , σ also held in that world. In the parlance of modal logic, we would say Alice's belief is actually *knowledge*: although she does not have all knowledge, everything she believes is in fact true. Why was this the case? Notice that Alice's possible-worlds relation is reflexive: for every world Alice's relation includes an edge pointing back to that world. That is why Alice cannot believe anything false. If σ is not true in a given world, Alice cannot believe σ there, because the definition

$$\begin{aligned} \mathcal{M}, w \models A \text{ believes } \sigma & \text{ iff } w \in \mathcal{E}(A \text{ believes } \sigma) \\ & \text{ iff } \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \end{aligned}$$

precludes it.

In modeling access control in the presence of arbitrary principals, however, we should certainly expect that some principals will believe (or at least claim to believe) untrue things. So we make no restriction of reflexivity on the relation that defines a principal's beliefs. Indeed, a principal may have an empty relation at a world: it may consider *no* worlds possible! In that case, at that world, the agent considers every statement true, since every statement is true in all of the zero worlds the agent considers possible. Indeed, the agent believes *false*. The agent's reasoning has become inconsistent; other agents would be wise not to follow this agent's beliefs.

B.1.2 Trust

Agents following one another's beliefs is exactly how we model trust. If Alice establishes that she believes everything Bob believes, then Alice does not have to be present for Bob to read one of her files: if Bob claims that reading the file would be good, Alice must agree, and the file server grants the request. To capture this trust, we observe that Alice is "less ignorant" than Bob: she believes everything Bob believes, and then perhaps more (on which Bob may remain silent). Therefore, from any actual world, Alice should consider possible a subset of the worlds Bob considers possible. If Bob says a statement, it is because he has ruled out the alternatives; he has no arrows to worlds where the statement is false. If $\mathcal{R}(A) \subseteq \mathcal{R}(B)$, then A has ruled out the same worlds, and must believe the same statement. Thus Alice says everything Bob says; if she says even more, it is because she disregards some

B.1.1 The nature of principal relations

Now that we have a formal structure for discussing the beliefs of principals, let us consider what kinds of beliefs are reasonable, and how one principal's beliefs should be related to another's.

Recall our example structure, where in any world, Alice was either ignorant (had no belief) about the pork or ignorant about the bananas. The first observation is that agents do not need to believe every true thing; statements about which they have neither a positive nor a negative belief represent a fact about which the agent is ignorant.

Furthermore, observe that Alice never believed anything false: in every world, if A believes σ also held in that world. In the parlance of modal logic, we would say Alice's belief is actually *knowledge*: although she does not have all knowledge, everything she believes is in fact true. Why was this the case? Notice that Alice's possible-worlds relation is reflexive: for every world Alice's relation includes an edge pointing back to that world. That is why Alice cannot believe anything false. If σ is not true in a given world, Alice cannot believe σ there, because the definition

$$\begin{aligned} \mathcal{M}, w \models A \text{ believes } \sigma & \text{ iff } w \in \mathcal{E}(A \text{ believes } \sigma) \\ & \text{ iff } \mathcal{R}(A)(w) \subseteq \mathcal{E}(\sigma) \end{aligned}$$

precludes it.

In modeling access control in the presence of arbitrary principals, however, we should certainly expect that some principals will believe (or at least claim to believe) untrue things. So we make no restriction of reflexivity on the relation that defines a principal's beliefs. Indeed, a principal may have an empty relation at a world: it may consider *no* worlds possible! In that case, at that world, the agent considers every statement true, since every statement is true in all of the zero worlds the agent considers possible. Indeed, the agent believes *false*. The agent's reasoning has become inconsistent; other agents would be wise not to follow this agent's beliefs.

B.1.2 Trust

Agents following one another's beliefs is exactly how we model trust. If Alice establishes that she believes everything Bob believes, then Alice does not have to be present for Bob to read one of her files: if Bob claims that reading the file would be good, Alice must agree, and the file server grants the request. To capture this trust, we observe that Alice is "less ignorant" than Bob: she believes everything Bob believes, and then perhaps more (on which Bob may remain silent). Therefore, from any actual world, Alice should consider possible a subset of the worlds Bob considers possible. If Bob says a statement, it is because he has ruled out the alternatives; he has no arrows to worlds where the statement is false. If $\mathcal{R}(A) \subseteq \mathcal{R}(B)$, then A has ruled out the same worlds, and must believe the same statement. Thus Alice says everything Bob says; if she says even more, it is because she disregards some

possible world that leaves Bob's belief ambiguous. You should convince yourself that if Bob believes σ , Alice has to believe the same thing, for she considers possible only a subset of the worlds Bob considers possible.

B.2 Further reading

Hughes and Cresswell provide the canonical, concise introduction to modal logic [HC96]. Fagin et al. provide a gentler introduction with motivating examples [FHMV95].

possible world that leaves Bob's belief ambiguous. You should convince yourself that if Bob believes σ , Alice has to believe the same thing, for she considers possible only a subset of the worlds Bob considers possible.

B.2 Further reading

Hughes and Cresswell provide the canonical, concise introduction to modal logic [HC96]. Fagin et al. provide a gentler introduction with motivating examples [FHMV95].

Appendix C

Review: the original Calculus for Access Control

This appendix contains an introduction to the Calculus for Access Control due to Abadi, Lampson, et al. [ABLP93, LABW92]. I have preserved here the names used for formulas in [LABW92]. I explicitly name formulas L1–L3, which are mentioned in passing in [LABW92, p. 273], and formulas A1–A4, which are mentioned in [ABLP93, pp. 712, 714, and 718]. The typographic conventions defined in Table 4.1 are used here, as well.

In the preceding appendix, I introduced an instance of modal logic: propositional logic plus the modal operators to capture the possibly ignorant, possibly false beliefs of fallible principals. The semantics I presented, based on Kripke structures, is exactly that used by Abadi to justify the calculus for access control. I introduced the semantics first, though, because conventionally the semantics is the “intuitive model” of the world, and the logic is a system for discovering theorems (statements that are true in every model) and reasoning from premises to conclusions that must appear in the model.

To apply modal logic to access control, Abadi et al. rename the operators. First, “believes” is renamed “says.” This is meant to capture the notion that the logic is *performative*: sometimes when a principal says something, that something becomes true. The act of saying to a file server that a file should be modified, given that the file server believes you, causes that file to indeed be modified. This renaming makes the quoting operator sound more natural: $B|A$ is Bob quoting Alice. $B|A$ **says** s is a synonym for B **says** A **says** s . “Belief” is still useful intuition, however. The operator is the same; Bob’s belief in σ can be inherited by Alice without Alice actually uttering σ .

A logic is a system of axioms and proof rules that let one reason from premises to conclusions: if the premise holds in a model, the conclusion holds as well. The logic of the Calculus is *sound* in that any conclusion proven in the logic holds in the model, but it is not *complete*: there are statements that are true in every model that cannot be proven in the logic. Abadi suggests that in fact the model may be *undecidable*: no logic system is

Appendix C

Review: the original Calculus for Access Control

This appendix contains an introduction to the Calculus for Access Control due to Abadi, Lampson, et al. [ABLP93, LABW92]. I have preserved here the names used for formulas in [LABW92]. I explicitly name formulas L1–L3, which are mentioned in passing in [LABW92, p. 273], and formulas A1–A4, which are mentioned in [ABLP93, pp. 712, 714, and 718]. The typographic conventions defined in Table 4.1 are used here, as well.

In the preceding appendix, I introduced an instance of modal logic: propositional logic plus the modal operators to capture the possibly ignorant, possibly false beliefs of fallible principals. The semantics I presented, based on Kripke structures, is exactly that used by Abadi to justify the calculus for access control. I introduced the semantics first, though, because conventionally the semantics is the “intuitive model” of the world, and the logic is a system for discovering theorems (statements that are true in every model) and reasoning from premises to conclusions that must appear in the model.

To apply modal logic to access control, Abadi et al. rename the operators. First, “believes” is renamed “says.” This is meant to capture the notion that the logic is *performative*: sometimes when a principal says something, that something becomes true. The act of saying to a file server that a file should be modified, given that the file server believes you, causes that file to indeed be modified. This renaming makes the quoting operator sound more natural: $B|A$ is Bob quoting Alice. $B|A$ **says** s is a synonym for B **says** A **says** s . “Belief” is still useful intuition, however. The operator is the same; Bob’s belief in σ can be inherited by Alice without Alice actually uttering σ .

A logic is a system of axioms and proof rules that let one reason from premises to conclusions: if the premise holds in a model, the conclusion holds as well. The logic of the Calculus is *sound* in that any conclusion proven in the logic holds in the model, but it is not *complete*: there are statements that are true in every model that cannot be proven in the logic. Abadi suggests that in fact the model may be *undecidable*: no logic system is

adequate to prove every valid statement of the model.

The logic of access control is the same (up to variations in notation) as the conventional modal logic system K_n . The subscript n indicates that there are multiple modal operators [HC96, FHMV95, p. 51]. I present that system here.

First, we write $\vdash \sigma$ if a statement σ is valid in the logic: either taken as an axiom, or provable as a theorem from other axioms and the proof rules. We prove theorems using the following:

If σ is a tautology of propositional calculus,
then $\vdash \sigma$ (Axiom S1)

The axiom lets us pull in the theorems of propositional calculus without explicitly mentioning the axioms and proof rules that produce them.

$$\frac{\vdash \sigma \quad \vdash \sigma \Rightarrow \tau}{\vdash \tau} \quad \text{(Rule S2)}$$

The proof rule (modus ponens) says that if both σ and the implication $\sigma \Rightarrow \tau$ are valid (provable), then τ is provable as well. It lets us prove theorems about formulas that include the modal operators (**says**) by reasoning from premises to conclusions.

We also have the *Distribution Axiom* (known in modal logic as the axiom **K**, from which the name of the system K_n derives):

$$\vdash \mathcal{A} \text{ says } (\sigma \Rightarrow \tau) \Rightarrow (\mathcal{A} \text{ says } \sigma \Rightarrow \mathcal{A} \text{ says } \tau) \quad \text{(Axiom S3)}$$

Intuitively it means that agents understand and believe all of the consequences of their beliefs. Furthermore, they believe every theorem:

$$\forall \mathcal{A}, \frac{\vdash \sigma}{\vdash \mathcal{A} \text{ says } \sigma} \quad \text{(Rule S4)}$$

That is, agents know all of the theorems of the logic.

There is a subtle but important distinction between implication in the metalogic (the proof rule above) and implication in the logic. The logical symbol $\vdash$ means that the premises on its left prove the conclusions on its right. The proof rule condition $\vdash \sigma$ means that no premises are required to prove σ ; that is, σ is a theorem. When that is true, we may conclude $\vdash \mathcal{A} \text{ says } \sigma$: it is proven that $\mathcal{A} \text{ says } \sigma$.

In contrast, the corresponding statement in the logic (not the metalogic) does not hold. The statement $\not\vdash \sigma \Rightarrow \mathcal{A} \text{ says } \sigma$ is read “it is not provable that σ implies $\mathcal{A} \text{ says } \sigma$ ”. The premise of the implication is an arbitrary statement σ (unlike the theorem $\vdash \sigma$ in the proof rule); it is not true that principals say every true statement. They say every theorem (those statements true in every world), but not every true statement (those statements true in the actual world from which the statement is being uttered).

adequate to prove every valid statement of the model.

The logic of access control is the same (up to variations in notation) as the conventional modal logic system K_n . The subscript n indicates that there are multiple modal operators [HC96, FHMV95, p. 51]. I present that system here.

First, we write $\vdash \sigma$ if a statement σ is valid in the logic: either taken as an axiom, or provable as a theorem from other axioms and the proof rules. We prove theorems using the following:

If σ is a tautology of propositional calculus,
then $\vdash \sigma$ (Axiom S1)

The axiom lets us pull in the theorems of propositional calculus without explicitly mentioning the axioms and proof rules that produce them.

$$\frac{\vdash \sigma \quad \vdash \sigma \Rightarrow \tau}{\vdash \tau} \quad \text{(Rule S2)}$$

The proof rule (modus ponens) says that if both σ and the implication $\sigma \Rightarrow \tau$ are valid (provable), then τ is provable as well. It lets us prove theorems about formulas that include the modal operators (**says**) by reasoning from premises to conclusions.

We also have the *Distribution Axiom* (known in modal logic as the axiom **K**, from which the name of the system K_n derives):

$$\vdash \mathcal{A} \text{ says } (\sigma \Rightarrow \tau) \Rightarrow (\mathcal{A} \text{ says } \sigma \Rightarrow \mathcal{A} \text{ says } \tau) \quad \text{(Axiom S3)}$$

Intuitively it means that agents understand and believe all of the consequences of their beliefs. Furthermore, they believe every theorem:

$$\forall \mathcal{A}. \frac{\vdash \sigma}{\vdash \mathcal{A} \text{ says } \sigma} \quad \text{(Rule S4)}$$

That is, agents know all of the theorems of the logic.

There is a subtle but important distinction between implication in the metalogic (the proof rule above) and implication in the logic. The logical symbol $\vdash$ means that the premises on its left prove the conclusions on its right. The proof rule condition $\vdash \sigma$ means that no premises are required to prove σ that is, σ is a theorem. When that is true, we may conclude $\vdash \mathcal{A} \text{ says } \sigma$: it is proven that $\mathcal{A} \text{ says } \sigma$.

In contrast, the corresponding statement in the logic (not the metalogic) does not hold. The statement $\not\vdash \sigma \Rightarrow \mathcal{A} \text{ says } \sigma$ is read “it is not provable that σ implies $\mathcal{A} \text{ says } \sigma$ ”. The premise of the implication is an arbitrary statement σ (unlike the theorem $\vdash \sigma$ in the proof rule); it is not true that principals say every true statement. They say every theorem (those statements true in every world), but not every true statement (those statements true in the actual world from which the statement is being uttered).

C.1 The calculus of principals

The symbol $=$ is an equivalence relation on principals; by $\mathcal{A} = \mathcal{B}$ we mean that $\mathcal{A}$ and $\mathcal{B}$ have the same relation and therefore the same beliefs.¹ (Elsewhere in the dissertation I also use $=$ to denote set equality; its use should be clear from context.)

I have presented the logical tools for reasoning about formulas of statements. Recall that we can also combine principals into principal formulas. For example, $\mathcal{A} \wedge \mathcal{B}$ is the principal that believes (says) only things that $\mathcal{A}$ and $\mathcal{B}$ agree upon. In the logic, $\mathcal{A} \wedge \mathcal{B}$ is defined in terms of its relationship to statements:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \text{ says } \sigma \equiv (\mathcal{A} \text{ says } \sigma) \wedge (\mathcal{B} \text{ says } \sigma) \quad (\text{Definition P1})$$

Principal conjunction is associative, commutative, and idempotent:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \wedge \mathcal{C} = \mathcal{A} \wedge (\mathcal{B} \wedge \mathcal{C}) \quad (\text{Axiom P4})$$

$$\vdash \mathcal{A} \wedge \mathcal{B} = \mathcal{B} \wedge \mathcal{A} \quad (\text{Axiom P4})$$

$$\vdash \mathcal{A} \wedge \mathcal{A} = \mathcal{A} \quad (\text{Axiom P4})$$

Quoting $(\mathcal{B}|\mathcal{A})$ is defined as:

$$\vdash (\mathcal{B}|\mathcal{A}) \text{ says } \sigma \equiv \mathcal{B} \text{ says } (\mathcal{A} \text{ says } \sigma) \quad (\text{Definition P2})$$

In a sense, the quoting operator “curries” a **says** operation from the propositional formula into the principal formula, so that one can talk about a principal quoting another without yet mentioning the specific statement being quoted.

Quoting is associative and distributes over conjunction in both arguments:

$$\vdash (\mathcal{A}|\mathcal{B})|\mathcal{C} = \mathcal{A}|(\mathcal{B}|\mathcal{C}) \quad (\text{Axiom P5})$$

$$\vdash \mathcal{A}|(\mathcal{B} \wedge \mathcal{C}) = (\mathcal{A}|\mathcal{B}) \wedge (\mathcal{A}|\mathcal{C}) \quad (\text{Axiom P6})$$

$$\vdash (\mathcal{A} \wedge \mathcal{B})|\mathcal{C} = (\mathcal{A}|\mathcal{C}) \wedge (\mathcal{B}|\mathcal{C})$$

C.2 The “speaks for” relation

A central concept of the calculus is the “speaks for” relation ($\Rightarrow$), which defines a partial order over all principals. This relation encodes the notion of one principal trusting another that I introduced in Section B.1.2. The statement $\mathcal{B} \Rightarrow \mathcal{A}$ is read “ $\mathcal{B}$ speaks for $\mathcal{A}$,” and means that whenever $\mathcal{B}$ says something, $\mathcal{A}$ certainly agrees. Formally, we define

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \equiv (\mathcal{B} = \mathcal{B} \wedge \mathcal{A}) \quad (\text{Definition P7})$$

¹Abadi et al. “note that $\mathcal{A}$ and $\mathcal{B}$ can have the same beliefs without having the same possible worlds relation; however, because principals are identified by their relations in the semantics, we define equality in terms of relations.” This situation is only possible if the model has two distinct worlds in W that belong to all the same I sets; that is, the model has two separate but indistinguishable worlds.

C.1 The calculus of principals

The symbol $=$ is an equivalence relation on principals; by $\mathcal{A} = \mathcal{B}$ we mean that $\mathcal{A}$ and $\mathcal{B}$ have the same relation and therefore the same beliefs.¹ (Elsewhere in the dissertation I also use $=$ to denote set equality; its use should be clear from context.)

We have just seen the logical tools for reasoning about formulas of statements. We can also combine principals into principal formulas. For example, $\mathcal{A} \wedge \mathcal{B}$ is the principal that believes (says) only things that $\mathcal{A}$ and $\mathcal{B}$ agree upon. In the logic, $\mathcal{A} \wedge \mathcal{B}$ is defined in terms of its relationship to statements:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \text{ says } \sigma \equiv (\mathcal{A} \text{ says } \sigma) \wedge (\mathcal{B} \text{ says } \sigma) \quad (\text{Definition P1})$$

Principal conjunction is associative, commutative, and idempotent:

$$\vdash (\mathcal{A} \wedge \mathcal{B}) \wedge \mathcal{C} = \mathcal{A} \wedge (\mathcal{B} \wedge \mathcal{C}) \quad (\text{Axiom P4})$$

$$\vdash \mathcal{A} \wedge \mathcal{B} = \mathcal{B} \wedge \mathcal{A} \quad (\text{Axiom P4})$$

$$\vdash \mathcal{A} \wedge \mathcal{A} = \mathcal{A} \quad (\text{Axiom P4})$$

Quoting $(\mathcal{B}|\mathcal{A})$ is defined as:

$$\vdash (\mathcal{B}|\mathcal{A}) \text{ says } \sigma \equiv \mathcal{B} \text{ says } (\mathcal{A} \text{ says } \sigma) \quad (\text{Definition P2})$$

In a sense, the quoting operator “curries” a **says** operation from the propositional formula into the principal formula, so that one can talk about a principal quoting another without yet mentioning the specific statement being quoted.

Quoting is associative and distributes over conjunction in both arguments:

$$\vdash (\mathcal{A}|\mathcal{B})|\mathcal{C} = \mathcal{A}|(\mathcal{B}|\mathcal{C}) \quad (\text{Axiom P5})$$

$$\vdash \mathcal{A}|(\mathcal{B} \wedge \mathcal{C}) = (\mathcal{A}|\mathcal{B}) \wedge (\mathcal{A}|\mathcal{C}) \quad (\text{Axiom P6})$$

$$\vdash (\mathcal{A} \wedge \mathcal{B})|\mathcal{C} = (\mathcal{A}|\mathcal{C}) \wedge (\mathcal{B}|\mathcal{C})$$

C.2 The “speaks for” relation

A central concept of the calculus is the “speaks for” relation ($\Rightarrow$), which defines a partial order over all principals. This relation encodes the notion of one principal trusting another that I introduced in Section B.1.2. The statement $\mathcal{B} \Rightarrow \mathcal{A}$ is read “ $\mathcal{B}$ speaks for $\mathcal{A}$,” and means that whenever $\mathcal{B}$ says something, $\mathcal{A}$ certainly agrees. Formally, we define

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \equiv (\mathcal{B} = \mathcal{B} \wedge \mathcal{A}) \quad (\text{Definition P7})$$

¹Abadi et al. “note that $\mathcal{A}$ and $\mathcal{B}$ can have the same beliefs without having the same possible worlds relation; however, because principals are identified by their relations in the semantics, we define equality in terms of relations.” This situation is only possible if the model has two distinct worlds in W that belong to all the same I sets; that is, the model has two separate but indistinguishable worlds.

Why is this the case? If $\mathcal{A}$ trusts $\mathcal{B}$, then $\mathcal{A}$ says everything $\mathcal{B}$ says. So the set of things $\mathcal{B} \wedge \mathcal{A}$ say must be the same as the set of things $\mathcal{B}$ says. It cannot be greater, by its semantic definition in Section B.1, and it cannot be less, or else there is something $\mathcal{B}$ says that $\mathcal{A}$ does not.

From the definition we can derive:²

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset ((\mathcal{B} \text{ says } \sigma) \supset (\mathcal{A} \text{ says } \sigma)) \quad (\text{Theorem P8})$$

When $\mathcal{B} \Rightarrow \mathcal{A}$, $\mathcal{B}$ is a stronger principal than $\mathcal{A}$ in the sense that $\mathcal{B}$ can do everything $\mathcal{A}$ can do (by making $\mathcal{A}$ believe the appropriate performative statement), and perhaps more.

Using the associativity of $\wedge$ for principals, it is clear that $\Rightarrow$ is a transitive relation:

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \wedge (\mathcal{C} \Rightarrow \mathcal{B}) \supset \mathcal{C} \Rightarrow \mathcal{A} \quad (\text{Theorem L1})$$

(The $\wedge$ in the theorem is that for statements. I would like to use a different symbol for clarity, but I stick with the notation of Abadi et al. here.) Both the $\wedge$ and $|$ operators on principals are monotonic with respect to $\Rightarrow$:

$$\vdash (\mathcal{A} \Rightarrow \mathcal{B}) \supset ((\mathcal{A} \wedge \mathcal{C}) \Rightarrow (\mathcal{B} \wedge \mathcal{C})) \quad (\text{Axiom L2})$$

$$\begin{aligned} \vdash (\mathcal{A} \Rightarrow \mathcal{B}) \supset ((\mathcal{A}|\mathcal{C}) \Rightarrow (\mathcal{B}|\mathcal{C})) \\ \vdash (\mathcal{A} \Rightarrow \mathcal{B}) \supset ((\mathcal{C}|\mathcal{A}) \Rightarrow (\mathcal{C}|\mathcal{B})) \end{aligned} \quad (\text{Axiom L3})$$

With the speaks-for relation, we can finally see why quoting is a useful operation. One can let $\mathcal{C}|\mathcal{B} \Rightarrow \mathcal{A}$, so that $\mathcal{C}$ can only speak for $\mathcal{A}$ when it quotes $\mathcal{B}$. Without quoting, we would need a formal accounting for universal quantification over formulas: $\forall \sigma, \mathcal{C} \text{ says } \mathcal{B} \text{ says } \sigma \supset \mathcal{A} \text{ says } \sigma$.

The semantics of $\Rightarrow$ falls out fairly directly. Definition P7 requires that

$$\begin{aligned} \mathcal{M}, w \models \mathcal{B} \Rightarrow \mathcal{A} \\ \text{iff } \mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{B} \wedge \mathcal{A}) = \mathcal{R}(\mathcal{B}) \cup \mathcal{R}(\mathcal{A}) \\ \text{iff } \mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B}) \end{aligned}$$

Notice that the condition on the $\mathcal{R}$ relations is independent of the world w . So the extension function $\mathcal{E}$ is all-or-nothing for speaks-for formulas:

$$\mathcal{E}(\mathcal{B} \Rightarrow \mathcal{A}) = \begin{cases} W & \text{if } \mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B}) \\ \emptyset & \text{otherwise} \end{cases} \quad (\text{Definition A1})$$

²Surprisingly, Abadi et al. drop Definition P7 and instead treat Theorem P8 as an axiom. Doing so leaves the logic with no axioms with $\Rightarrow$ in the conclusion, and hence precludes theorems whose conclusions establish $\Rightarrow$ relationships. In fact, Theorem P8 requires only the weaker operator $\rightarrow$ in its premise, which I discuss in Section 4.2.1.

Why is this the case? If $\mathcal{A}$ trusts $\mathcal{B}$, then $\mathcal{A}$ says everything $\mathcal{B}$ says. So the set of things $\mathcal{B} \wedge \mathcal{A}$ say must be the same as the set of things $\mathcal{B}$ says. It cannot be greater, by its semantic definition in Section B.1, and it cannot be less, or else there is something $\mathcal{B}$ says that $\mathcal{A}$ does not.

From the definition we can derive:²

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \supset ((\mathcal{B} \text{ says } \sigma) \supset (\mathcal{A} \text{ says } \sigma)) \quad (\text{Theorem P8})$$

When $\mathcal{B} \Rightarrow \mathcal{A}$, $\mathcal{B}$ is a stronger principal than $\mathcal{A}$ in the sense that $\mathcal{B}$ can do everything $\mathcal{A}$ can do (by making $\mathcal{A}$ believe the appropriate performative statement), and perhaps more.

Using the associativity of $\wedge$ for principals, it is clear that $\Rightarrow$ is a transitive relation:

$$\vdash (\mathcal{B} \Rightarrow \mathcal{A}) \wedge (\mathcal{C} \Rightarrow \mathcal{B}) \supset \mathcal{C} \Rightarrow \mathcal{A} \quad (\text{Theorem L1})$$

(The $\wedge$ in the theorem is that for statements. I would like to use a different symbol for clarity, but I stick with the notation of Abadi et al. here.) Both the $\wedge$ and $|$ operators on principals are monotonic with respect to $\Rightarrow$:

$$\vdash (\mathcal{A} \Rightarrow \mathcal{B}) \supset ((\mathcal{A} \wedge \mathcal{C}) \Rightarrow (\mathcal{B} \wedge \mathcal{C})) \quad (\text{Axiom L2})$$

$$\begin{aligned} \vdash (\mathcal{A} \Rightarrow \mathcal{B}) &\supset ((\mathcal{A}|\mathcal{C}) \Rightarrow (\mathcal{B}|\mathcal{C})) \\ \vdash (\mathcal{A} \Rightarrow \mathcal{B}) &\supset ((\mathcal{C}|\mathcal{A}) \Rightarrow (\mathcal{C}|\mathcal{B})) \end{aligned} \quad (\text{Axiom L3})$$

With the speaks-for relation, we can finally see why quoting is a useful operation. One can let $\mathcal{C}|\mathcal{B} \Rightarrow \mathcal{A}$, so that $\mathcal{C}$ can only speak for $\mathcal{A}$ when it quotes $\mathcal{B}$. Without quoting, we would need a formal accounting for universal quantification over formulas: $\forall \sigma, \mathcal{C} \text{ says } \mathcal{B} \text{ says } \sigma \supset \mathcal{A} \text{ says } \sigma$.

The semantics of $\Rightarrow$ falls out fairly directly. Definition P7 requires that

$$\begin{aligned} \mathcal{M}, w \models \mathcal{B} \Rightarrow \mathcal{A} \\ \text{iff } \mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{B} \wedge \mathcal{A}) = \mathcal{R}(\mathcal{B}) \cup \mathcal{R}(\mathcal{A}) \\ \text{iff } \mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B}) \end{aligned}$$

Notice that the condition on the $\mathcal{R}$ relations is independent of the world w . So the extension function $\mathcal{E}$ is all-or-nothing for speaks-for formulas:

$$\mathcal{E}(\mathcal{B} \Rightarrow \mathcal{A}) = \begin{cases} W & \text{if } \mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\mathcal{B}) \\ \emptyset & \text{otherwise} \end{cases} \quad (\text{Definition A1})$$

²Surprisingly, Abadi et al. drop Definition P7 and instead treat Theorem P8 as an axiom. Doing so leaves the logic with no axioms with $\Rightarrow$ in the conclusion, and hence precludes theorems whose conclusions establish $\Rightarrow$ relationships. In fact, Theorem P8 requires only the weaker operator $\rightarrow$ in its premise, which I discuss in Section 4.2.1.

C.3 Access Control Lists

The speaks-for relation, because it is transitive, lets us reason broadly about how principals' beliefs affect one another. In the end, however, the server wants to convince itself that some primitive proposition s , perhaps to be interpreted “it is okay to change the contents of the file,” is true. To support this, Abadi, Lampson et al. use the construct $\mathcal{A} \text{ controls } s$ to indicate that principal $\mathcal{A}$'s beliefs about s are taken to be truth. It is defined as:

$$\mathcal{A} \text{ controls } s \equiv ((\mathcal{A} \text{ says } s) \supset s) \quad (\text{Definition A2})$$

Now suppose $\mathcal{B}$ wants to write to the file that s describes, and the assumptions $\vdash \mathcal{B} \Rightarrow \mathcal{A}$ and $\vdash \mathcal{A} \text{ controls } s$ hold. Then the file server will be able to verify a proof of $\vdash s$, convincing itself that “it is okay to change the contents of the file.”

Lampson et al. encode access control lists (ACLs) using *controls* assumptions:

$$\text{ACL}(O_1) = \left\{ \begin{array}{l} \vdash \mathcal{A} \text{ controls } s_{\text{read}}, \\ \vdash \mathcal{A} \text{ controls } s_{\text{write}}, \\ \vdash \mathcal{B} \text{ controls } s_{\text{read}} \end{array} \right\}$$

By adjusting which principals' assertions are believed, the ACLs allow or disallow agents to effect action.

C.4 Higher-level operators

The operating system that instantiates the calculus requires resource servers to construct and then verify all necessary proofs [WABL94]. Wobber calls it a *pull* model: it is the servers' job to pull in necessary assumptions and proof components needed to verify an agent's access. Building such proofs, when assumptions include speaks-for formulas with arbitrary combinations of $\wedge$ and $\mid$ operators, takes exponential time. To make the decision problem tractable, Lampson et al. define two high-level operators, **as** and **for**, in terms of the lower-level operators. Each operator is designed to reflect an idiomatic usage pattern of the calculus. The higher-level operators can combine in fewer ways than the lower-level operators, allowing an implementation to exploit characteristics such as associativity and idempotence. In the abstract, the operators can be treated as abbreviations and replaced by their definitions, and they do not affect the calculus. I cover them here to demonstrate the idioms they represent.

C.5 Roles and the “as” operator

Abadi et al. define a distinguished, disjoint set of principals called roles. By quoting a role, a principal restricts its own authority. For example, define the roles R_{user} and R_{admin} representing a person acting as a user and as an administrator, respectively. Suppose the

C.3 Access Control Lists

The speaks-for relation, because it is transitive, lets us reason broadly about how principals' beliefs affect one another. In the end, however, the server wants to convince itself that some primitive proposition s , perhaps to be interpreted “it is okay to change the contents of the file,” is true. To support this, Abadi, Lampson et al. use the construct $\mathcal{A} \text{ controls } s$ to indicate that principal $\mathcal{A}$'s beliefs about s are taken to be truth. It is defined as:

$$\mathcal{A} \text{ controls } s \equiv ((\mathcal{A} \text{ says } s) \supset s) \quad (\text{Definition A2})$$

Now suppose $\mathcal{B}$ wants to write to the file that s describes, and the assumptions $\vdash \mathcal{B} \Rightarrow \mathcal{A}$ and $\vdash \mathcal{A} \text{ controls } s$ hold. Then the file server will be able to verify a proof of $\vdash s$, convincing itself that “it is okay to change the contents of the file.”

Lampson et al. encode access control lists (ACLs) using *controls* assumptions:

$$\text{ACL}(O_1) = \left\{ \begin{array}{l} \vdash \mathcal{A} \text{ controls } s_{read}, \\ \vdash \mathcal{A} \text{ controls } s_{write}, \\ \vdash \mathcal{B} \text{ controls } s_{read} \end{array} \right\}$$

By adjusting which principals' assertions are believed, the ACLs allow or disallow agents to effect action.

C.4 Higher-level operators

The operating system that instantiates the calculus requires resource servers to construct and then verify all necessary proofs [WABL94]. Wobber calls it a *pull* model: it is the servers' job to pull in necessary assumptions and proof components needed to verify an agent's access. Building such proofs, when assumptions include speaks-for formulas with arbitrary combinations of $\wedge$ and $\mid$ operators, takes exponential time. To make the decision problem tractable, Lampson et al. define two high-level operators, **as** and **for**, in terms of the lower-level operators. Each operator is designed to reflect an idiomatic usage pattern of the calculus. The higher-level operators can combine in fewer ways than the lower-level operators, allowing an implementation to exploit characteristics such as associativity and idempotence. In the abstract, the operators can be treated as abbreviations and replaced by their definitions, and they do not affect the calculus. I cover them here to demonstrate the idioms they represent.

C.5 Roles and the “as” operator

Abadi et al. define a distinguished, disjoint set of principals called roles. By quoting a role, a principal restricts its own authority. For example, define the roles R_{user} and R_{admin} representing a person acting as a user and as an administrator, respectively. Suppose the

Appendix D

Review: The Simple Public Key Infrastructure

The Simple Public Key Infrastructure 2.0 (SPKI, pronounced “spooky”) is an Internet Experimental Protocol created by Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen [EFL⁺99]. As its name suggests, it is designed to be a unifying standard for supporting public key authorization across the global Internet. I highlight here some of the features of SPKI relevant to my work.

First, SPKI’s primary goal is to provide a server with evidence that the holder of a given cryptographic key is ultimately authorized for a request signed by that key. This goal contrasts with that of other public-key infrastructure efforts that attempt to bind keys to identities, and leave authorization to be handled in the conventional fashion by ACLs that map identity to authorization.

In this section, I review the types of certificates that SPKI supports, and outline the procedure used to determine whether a given certificate chain supports a requested operation.

D.1 Certificate types

SPKI defines its own certificate format, as well as an internal representation of certificates to which it can map other inputs, such as PGP certificates, X.509 certificates, or locally maintained ACL entries. Authorization results can be constructed from inputs providing information in one of three forms:

- $\langle \text{authorization}, \text{key} \rangle$
- $\langle \text{authorization}, \text{name} \rangle$

Appendix D

Review: The Simple Public Key Infrastructure

The Simple Public Key Infrastructure 2.0 (SPKI, pronounced “spooky”) is an Internet Experimental Protocol created by Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen [EFL⁺99]. As its name suggests, it is designed to be a unifying standard for supporting public key authorization across the global Internet. I highlight here some of the features of SPKI relevant to my work.

First, SPKI’s primary goal is to provide a server with evidence that the holder of a given cryptographic key is ultimately authorized for a request signed by that key. This goal contrasts with that of other public-key infrastructure efforts that attempt to bind keys to identities, and leave authorization to be handled in the conventional fashion by ACLs that map identity to authorization.

In this section, I review the types of certificates that SPKI supports, and outline the procedure used to determine whether a given certificate chain supports a requested operation.

D.1 Certificate types

SPKI defines its own certificate format, as well as an internal representation of certificates to which it can map other inputs, such as PGP certificates, X.509 certificates, or locally maintained ACL entries. Authorization results can be constructed from inputs providing information in one of three forms:

- $\langle \text{authorization}, \text{key} \rangle$
- $\langle \text{authorization}, \text{name} \rangle$

- $\langle \text{name}, \text{key} \rangle$

The first form coincides with SPKI's design philosophy of mapping keys directly to authorizations. Inputs of the latter two forms must ultimately be combined to form a $\langle \text{authorization}, \text{key} \rangle$ mapping to become useful.

Inputs of the first two forms are mapped into a data structure called a *5-tuple* for internal processing; inputs of the latter form are mapped into a data structure called a *4-tuple*.

D.2 The SPKI 5-tuple

A 5-tuple has the following fields:

- issuer: the public key granting the permission defined by the 5-tuple
- subject: a public key or name to which the permission is being granted
- delegation-control: a boolean value indicating whether this permission may be further delegated
- authorization: a set of primitive permissions being granted
- validity dates: a date range limiting the validity of this delegation

The intended meaning is that the issuer grants the subject the permission described in the authorization field for the duration of the validity dates. If the delegation-control bit is set, the subject may further delegate any or all of the permission to another subject.

The subject in a 5-tuple (or a 4-tuple, which I present shortly) may be a *k-of-n* threshold function. In this case, the permission is delegated to any principal that can prove it is authorized to speak for any *k* of the *n* “subordinate” subjects listed in the threshold function.

The authorization fields contain primitive permissions whose interpretation is left to the application employing the SPKI authorization engine. These permissions are represented using *tags*. Tags encode infinitely large sets of primitive statements in a form that permits a compact representation of certain subsets. Notably, a tag can represent only a set of primitive symbols; never a formula made from the negation or conjunction of primitive symbols. Tags admit a simple intersection algorithm that always yields a compact representation of the intersected set.

SPKI certificates may also indicate an on-line mechanism for verifying that the issuer considers a certificate still valid. Two of the checks, the certificate revocation list (CRL, a negative list of revoked certificates) and the timed revalidation (a positive list of still-valid certificates), are performed by consulting a list revised more frequently than the original

- $\langle \text{name}, \text{key} \rangle$

The first form coincides with SPKI's design philosophy of mapping keys directly to authorizations. Inputs of the latter two forms must ultimately be combined to form a $\langle \text{authorization}, \text{key} \rangle$ mapping to become useful.

Inputs of the first two forms are mapped into a data structure called a *5-tuple* for internal processing; inputs of the latter form are mapped into a data structure called a *4-tuple*.

D.2 The SPKI 5-tuple

A 5-tuple has the following fields:

- issuer: the public key granting the permission defined by the 5-tuple
- subject: a public key or name to which the permission is being granted
- delegation-control: a boolean value indicating whether this permission may be further delegated
- authorization: a set of primitive permissions being granted
- validity dates: a date range limiting the validity of this delegation

The intended meaning is that the issuer grants the subject the permission described in the authorization field for the duration of the validity dates. If the delegation-control bit is set, the subject may further delegate any or all of the permission to another subject.

The subject in a 5-tuple (or a 4-tuple, which I present shortly) may be a *k-of-n* threshold function. In this case, the permission is delegated to any principal that can prove it is authorized to speak for any *k* of the *n* “subordinate” subjects listed in the threshold function.

The authorization fields contain primitive permissions whose interpretation is left to the application employing the SPKI authorization engine. These permissions are represented using *tags*. Tags encode infinitely large sets of primitive statements in a form that permits a compact representation of certain subsets. Notably, a tag can represent only a set of primitive symbols; never a formula made from the negation or conjunction of primitive symbols. Tags admit a simple intersection algorithm that always yields a compact representation of the intersected set.

SPKI certificates may also indicate an on-line mechanism for verifying that the issuer considers a certificate still valid. Two of the checks, the certificate revocation list (CRL, a negative list of revoked certificates) and the timed revalidation (a positive list of still-valid certificates), are performed by consulting a list revised more frequently than the original

Reduction proceeds as follows. First, 4-tuples are reduced to resolve names. 4-tuples that define a name in terms of another grounded chain of names are reduced using 4-tuples that define a name in terms of a key. Eventually, 4-tuples of the former form are reduced to 4-tuples of the latter form. The validity date stored in the outcome of each reduction is the intersection of the validity dates of the 4-tuples being reduced.

Then the $\langle \text{name}, \text{key} \rangle$ bindings formed by the reduced 4-tuples are applied to resolve names in 5-tuples back to keys, again carrying validity dates through with intersection operations. This operation turns $\langle \text{authorization}, \text{name} \rangle$ 5-tuples into $\langle \text{authorization}, \text{key} \rangle$ tuples.

At this point, each 5-tuple represents a subject key (or threshold subject defined as a set of keys) with authorization to perform some set of actions on behalf of the issuer key. When two 5-tuples form a chain of delegation (the issuer of the second is the subject of the first, and the first tuple allows further delegation), the 5-tuples are reduced to a new tuple whose subject is the subject of the second tuple and whose issuer is the issuer of the first. The reduced tuple carries the intersection of the authorizations of the source tuples as its authorization, and the intersection of the validity dates of the source tuples as its validity dates. Finally, the reduced tuple carries the same delegation control bit as the second tuple did. Think of the delegation control bit as the coupling on the back of a boxcar; if the first tuple lacks it, the cars cannot couple; if the second tuple lacks it, the cars may couple, but the resulting “super-car” will also lack a rear coupling.

Reduction proceeds as follows. First, 4-tuples are reduced to resolve names. 4-tuples that define a name in terms of another grounded chain of names are reduced using 4-tuples that define a name in terms of a key. Eventually, 4-tuples of the former form are reduced to 4-tuples of the latter form. The validity date stored in the outcome of each reduction is the intersection of the validity dates of the 4-tuples being reduced.

Then the $\langle \text{name}, \text{key} \rangle$ bindings formed by the reduced 4-tuples are applied to resolve names in 5-tuples back to keys, again carrying validity dates through with intersection operations. This operation turns $\langle \text{authorization}, \text{name} \rangle$ 5-tuples into $\langle \text{authorization}, \text{key} \rangle$ tuples.

At this point, each 5-tuple represents a subject key (or threshold subject defined as a set of keys) with authorization to perform some set of actions on behalf of the issuer key. When two 5-tuples form a chain of delegation (the issuer of the second is the subject of the first, and the first tuple allows further delegation), the 5-tuples are reduced to a new tuple whose subject is the subject of the second tuple and whose issuer is the issuer of the first. The reduced tuple carries the intersection of the authorizations of the source tuples as its authorization, and the intersection of the validity dates of the source tuples as its validity dates. Finally, the reduced tuple carries the same delegation control bit as the second tuple did. Think of the delegation control bit as the coupling on the back of a boxcar; if the first tuple lacks it, the cars cannot couple; if the second tuple lacks it, the cars may couple, but the resulting “super-car” will also lack a rear coupling.