A logical reconstruction of SPKI*

Joseph Y. Halpern†
Cornell University
Dept. of Computer Science
Ithaca, NY 14853
halpern@cs.cornell.edu
http://www.cs.cornell.edu/home/halpern

Ron van der Meyden
University of New South Wales
Australia
meyden@cse.unsw.edu.au
http://www.cse.unsw.edu.au/∼meyden

Abstract

SPKI/SDSI is a proposed public key infrastructure standard that incorporates the SDSI public key infrastructure. SDSI's key innovation was the use of *local names*. We previously introduced a Logic of Local Name Containment that has a clear semantics and was shown to completely characterize SDSI name resolution. Here we show how our earlier approach can be extended to deal with a number of key features of SPKI, including revocation, expiry dates, and tuple reduction. We show that these extensions add relatively little complexity to the logic. In particular, we do not need a nonmonotonic logic to capture revocation. We then use our semantics to examine SPKI's tuple reduction rules. Our analysis highlights places where SPKI's informal description of tuple reduction is somewhat vague, and shows that extra reduction rules are necessary in order to capture general information about binding and authorization.

*A preliminary version of this appeared in the *Proceedings of the 14th IEEE Computer Security Foundations Workshop*, 2001, pp. 59–70.

†Supported in part by NSF under grants IRI-96-25901 and IIS-0090145, by ONR under grants N00014-00-1-0341, N00014-01-1-0511, and N00014-02-1-0455, by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grants N00014-97-0505 and N00014-01-1-0795, by AFOSR under grant F49620-02-1-0101, and by a Guggenheim Fellowship and a Fulbright Fellowship. Sabbatical support from CWI and the Hebrew University of Jerusalem is also gratefully acknowledged.

1 Introduction

Rivest and Lampson (1996) introduced SDSI—a Simple Distributed Security Infrastructure—to facilitate the construction of secure systems. In SDSI, principals (agents) are identified with public keys. In addition to principals, SDSI allows other names, such as `poker-buddies`. Rather than having a global name space, these names are interpreted *locally*, by each principal. That is, each principal associates with each name a set of principals. Of course, the interpretation of a name such as `poker-buddies` may be different for each agent. However, a principal can "export" his bindings to other principals using signed certificates. Thus, Ron may receive a signed certificate from the principal he names `Joe` describing a set of principals Joe associates with `poker-buddies`. Ron may then refer to this set of principals by the expression `Joe's poker-buddies`. Rivest and Lampson (1996) give an operational account of local names; they provide a name-resolution algorithm that, given a principal `k` and a name `n`, computes the set of principals associated with `n` according to `k`. In (Halpern and van der Meyden 2001), building on earlier work of Abadi (1998), we give a logic LLNC, the Logic of Local Name Containment, with clean semantics that precisely captures SDSI's operational name resolution algorithm.

However, our earlier work made a number of simplifying assumptions to bring out what we saw as the main issues of name spaces. In particular, we (along with Abadi) assumed that certificates never expired and were not revoked. SDSI has been incorporated into SPKI (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a; Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999b), which allows expiry dates for certificates and revocation, and deals with authorization and delegation in addition to naming. In this paper, we show how our earlier approach can be extended to deal with these features of SPKI.

By not having expiry dates and not allowing revocation, we get a *monotonicity* property: having more certificates can never mean that fewer keys are bound to a given name. Heavy use seems to be made of monotonicity in our earlier work.[1] A number of authors have developed logical accounts of authorization based on *nonmonotonic* logics (Woo and Lam 1993; Jajodia, Samarati, and Subramanian 1997; Li, Grosof, and Feigenbaum 1999; Li, Grosof, and Feigenbaum 2000). These are logics where conclusions can be retracted in the presence of more information (so that C may follow from A but not from $A \wedge B$). It has been suggested that revocation should be modeled using a nonmonotonic logic (Li, Grosof, and Feigenbaum 1999).[2] Dealing with nonmonotonicity adds significant compli-

[1]Ninghui Li (2000) has erroneously claimed that LLNC is nonmonotonic. See (Halpern and van der Meyden 2001) for a rebuttal and discussion of this claim.

[2]Although it does not go into details about the nonmonotonic features, (Li, Grosof, and Feigenbaum 1999) mentions a logic DL, whose notable features are said to include "The ability to handle non-monotonic policies. These are policies that deal explicitly with 'negative evidence' and specify types of requests that do not comply. Important examples include hot-lists of 'revoked' credentials .... 'Non-monotonic' here means in the sense of logic-based knowledge representation (KR)." Some of the authors

cations to a logic, both conceptually and from a complexity-theoretic point of view (see, for example, (Cadoli and Lenzerini 1994)). Thus, there is a benefit to using a monotonic logic where possible. We show that there is no difficulty capturing expiry dates and revocation (at least as they appear in SPKI) using a monotonic logic. Interestingly, SPKI's semantics maintains monotonicity even in the presence of revocation, in the sense that having more certificates (even more revocation certificates) still allows us to draw more conclusions about both name bindings and authorizations. (Roughly speaking, this is because, in SPKI, a certificate is ignored unless it is known *not* to have been revoked. Having a revocation certificate issued by `k` covering a certain time `t` ensures that there are no other revocation certificates issued by `k` covering `t`, and thus allows us to conclude that certain certificates have not been revoked at time `t`. Thus, by having more revocation certificates, we can draw more conclusions.)

We remark that, although SPKI is monotonic with respect to adding more certificates, it is not monotonic with respect to time. Keys that are bound to a name at time t may no longer be bound to that name at time $t' > t$. (Indeed, this does not require revocation; it suffices that certificates have intervals of validity.)

SPKI gives semantics to certificates by first converting them to tuples, and then providing tuple reduction rules, which are used to reduce the tuples to a particularly simple form (corresponding to basic name binding and authorization decisions). We associate with each SPKI certificate a formula in our logic. Thus, we have two ways of giving semantics to SPKI certificates: through tuple reduction and through the logic. The focus of this paper is on examining the connection between these two approaches. Our analysis highlights places where SPKI's informal description of tuple reduction is somewhat vague, and shows that extra reduction rules are necessary in order to capture general information about binding and authorization. Besides clarifying ambiguities, the logic allows for reasoning about the consequences of certain certifications and general reasoning about naming and authorization. (See Section 7 for further discussion of the potential uses of the logic.)

The rest of this paper is organized as follows. In the next section, we briefly describe the syntax of SPKI. In Section 3, we describe SPKI's reduction rules. Section 4 describes the syntax and semantics of our logic for reasoning about SPKI, which extends LLNC. In Section 5, we prove our main results, which involve characterizing the power of SPKI reduction rules in terms of our logic. In Section 6, we compare our work to several other recent approaches to giving semantics to SPKI. We conclude in Section 7 with further discussion of the logic.

---

of this paper have also taken an alternate position: Li and Feigenbaum (2001) recommend that "a PKI should provide an interface that is monotonic".

2 SPKI syntax

SPKI views authority as being associated with *principals*, which it identifies with public keys. Instead of global names, SPKI has incorporated SDSI's notion of *local name space*. In SDSI/SPKI, a local name such as `Joe` is interpreted with respect to a principal, and its meaning may vary from principal to principal. There is no requirement that a local name refer to a unique principal. For example, a local name such as `poker-buddies` may refer to a set of principals. Within its name space, a principal may refer to the interpretation of names in another principal's name space by means of compound names. For example, a principal may use the expression `Joe`'s `poker-buddies` to refer to the principals that the principal he refers to as `Joe` refers to as `poker-buddies`. SPKI calls such expressions *compound (SDSI) names*, and uses the syntax $(\mathtt{name}\ \mathtt{n}_1\ \mathtt{n}_2 \ldots \mathtt{n}_k)$, where the $\mathtt{n}_i$ are local names (called *basic SDSI names* in the SPKI document) for $i > 1$ and $\mathtt{n}_1$ is either a local name or a key. Such an expression may also be represented as $\mathtt{n}_1$'s $\mathtt{n}_2$'s $\ldots \mathtt{n}_k$. A *fully-qualified* name is one where $\mathtt{n}_1$ is a key and $\mathtt{n}_2, \ldots, \mathtt{n}_k$ are local names. While, in general, the interpretation of a compound name depends on the principal (so that the interpretation of `Joe`'s `poker-buddies` by key $\mathtt{k}_1$ depends on $\mathtt{k}_1$'s interpretation of `Joe`, and may be different from $\mathtt{k}_2$'s interpretation of `Joe` and `Joe's poker-buddies`), the interpretation of a fully-qualified name is independent of the principal.

SPKI has other ways of identifying principals. For example, SPKI principals may also be the hash of a key, a threshold subject (an expression representing "any m out of N of the following subjects", used to capture requirements for joint signatures), or the reserved word "Self", representing the entity doing the verification. For simplicity, in this paper, the only principals we consider are those defined by compound names.

There are two types of certificates in SPKI, *naming certificates*, *authorization certificates*. SPKI also has *certificate revocation lists* (CRLs); for uniformity, we treat these as certificates as well. Again, this seems completely consistent with the SPKI treatment. A naming certificate has the form of a cryptographically signed message with contents

$$(\mathtt{cert}\ (\mathtt{issuer}\ (\mathtt{name}\ \mathtt{k}\ \mathtt{n}))\ (\mathtt{subject}\ \mathtt{p})\ \langle\mathtt{valid}\rangle),$$

where `k` is a key (representing the issuer, whose signature should be on the certificate), `n` is a local name, `p` is a fully-qualified SDSI name,[3] and $\langle\mathtt{valid}\rangle$ is an optional section describing validity constraints on the certificate. The $\langle\mathtt{valid}\rangle$ section may describe an interval during which the certificate is valid, expressed by means of a "not-before date" (expressed in the syntax as $(\mathtt{not\text{-}before}\ \langle\mathtt{date}\rangle)$) and/or a "not-after date" (expressed as $(\mathtt{not\text{-}after}\ \langle\mathtt{date}\rangle)$). It may also describe a sequence of "online test" expressions, which specify that the certificate should be verified either by checking a *certificate revocation list* (CRL) (intuitively, a list of certificates that have been revoked), by checking

[3]SPKI also allows `p` to be an unqualified name (that is, a string of local names), but notes that in this case it is to be interpreted as `k`'s `p`, which is a fully qualified name. For simplicity, we insist upon fully qualified names here.

a revalidation list (a list of currently valid certificates), or by performing an online test. Each of these components is optional.

In this paper, we assume that the $\langle\texttt{valid}\rangle$ field contains only validity intervals and a key authorized to sign revocation lists relevant to the certificate; the treatment for revalidation lists and other online tests is similar and does not add new subtleties.[4] We represent dates as natural numbers. From the `not-before` and `not-after` sections of a certificate we may obtain a validity interval $\texttt{V} = [\texttt{t}_1, \texttt{t}_2]$ where $\texttt{t}_1, \texttt{t}_2 \in I\!N \cup \{\infty\}$ are respectively the not-before time and the not-after times indicated. If no `not-before` time is given, we take $\texttt{t}_1 = 0$ and, similarly, if no `not-after` time is given, we take $\texttt{t}_2 = \infty$. We assume that $\texttt{t}_1 \leq \texttt{t}_2$, so that the validity interval is nonempty. We also allow the empty interval, which we denote $\emptyset$.

For simplicity, we abbreviate naming certificates as $(\texttt{cert k n p V k}_r)$, where V has the form $[t_1, t_2]$ and $\texttt{k}_r$ is the key which is authorized to sign CRLs relevant to the certificate. The $\texttt{k}_r$ component may not be present: if it is, then we say that the certificate is *revocable by* $\texttt{k}_r$. We occasionally write $(\texttt{cert k n p V } \langle\texttt{k}_r\rangle)$ to denote a generic naming certificate where $\texttt{k}_r$ may or may not be present. A naming certificate binds the fully-qualified name `p` to the local name `n` in `k`'s local name space during the period `V`, provided that certificate does not appear in any CRLs signed by $\texttt{k}_r$. Binding `p` to `n` means that the interpretation of `n` with respect to `k` includes the meaning of `p`. For example, if Ron and Joe are principals, then the certificate (`cert Ron doctor (Joe's doctor)` $[1, 3]$) binds `Joe's doctor` to the local name `doctor` in `Ron`'s local name space from time 1 to time 3; moreover, this certificate is irrevocable.

Authorization certificates have the form

$$(\texttt{cert (issuer k) (subject p) (propagate) A } \langle\texttt{valid}\rangle),$$

where `k` is a key, `p` is a fully-qualified name,[5] `A` is what the SPKI document calls an *authorization* and we call an *action expression*, since it represents a set of actions, and $\langle\texttt{valid}\rangle$ is a validity section, as described above. The "(propagate)" section is optional. Intuitively, the issuer uses such a certificate to grant the subject the authority to perform the actions in `A`. Moreover, if "(propagate)" is present, then the subject is further authorized to propagate this authority to others.[6] We abbreviate authorization certificates as $(\texttt{cert k p D A V k}_r)$, where `D` is a a Boolean (which stands for *delegate*) indicating

[4]SPKI also allows a certificate to specify a list of locations where the CRL may be obtained (rather than requiring that the actual CRL be sent), and to provide `http` requests to these locations with extra parameters, but we ignore these components since they do not interact with the semantic issues we address.

[5]SPKI also allows the subject to be an unqualified name. We make the simplifying assumption of qualified names just as we did above.

[6]Note that if "(propagate)" is not present, then we treat this as there being no indication of whether the subject is authorized to propagate the authority to others, rather than it being the case that the subject is *not* permitted to propagate the authority. This allows it to be consistent for `k` to issue two certificates that are identical except that one contains "(propagate)" while the other does not. Under our interpretation, the former supersedes the latter. (This seems particularly reasonable if we assume that

whether or not propagation is permitted. Again, the $\mathtt{k}_r$ component is optional, and we use (`cert k p D A V` $\langle \mathtt{k}_r \rangle$) to denote a generic authorization certificate where the $\mathtt{k}_r$ component may or may not be present.

SPKI takes an action expression `A` to be an *S-expression*—a list of strings or sublists. It uses `AIntersect` to denote the intersection of action expressions.[7] As we suggested above, action expressions are best thought of as sets of actions. We abstract this by assuming that there is some set *Act* of actions and a set $\mathcal{A}$ of action expressions that, intuitively, represent sets of actions in *Act*. We assume that $\mathcal{A}$ includes all finite subsets $\{\mathtt{a}_1, \ldots, \mathtt{a}_n\}$ of *Act*. Moreover, we assume that, given two action expressions $\mathtt{A}_1$ and $\mathtt{A}_2$ in $\mathcal{A}$, we can easily compute a third action expression in $\mathcal{A}$, denoted $\mathtt{A}_1 \cap \mathtt{A}_2$, which intuitively represents the intersection of the sets represented by $\mathtt{A}_1$ and $\mathtt{A}_2$. (We capture this intuition by a semantic constraint below.) The reason that we allow expressions in $\mathcal{A}$, rather than just finite subsets of *Act*, is that SPKI allows (some) expressions that represent infinite sets of actions. For example, SPKI allows an action expression of the form (`ftp ftp.clark.net /pub/cme/*`), which allows access to directories ftp.clark.net that start with /pub/cme/ (see (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6.3.1)).

We assume that there is a fixed *action interpretation* $\alpha_{\mathcal{A}}$ that maps action expressions in $\mathcal{A}$ to subsets of *Act*. We require that if `A` is a finite subset of *Act*, then $\alpha_{\mathcal{A}}(\mathtt{A}) = \mathtt{A}$, and that $\alpha_{\mathcal{A}}(\mathtt{A}_1 \cap \mathtt{A}_2) = \alpha_{\mathcal{A}}(\mathtt{A}_1) \cap \alpha_{\mathcal{A}}(\mathtt{A}_2)$. We also assume that we can decide (given $\mathtt{A}_1$ and $\mathtt{A}_2$) whether $\alpha_{\mathcal{A}}(\mathtt{A}_1 \cap \mathtt{A}_2) = \alpha_{\mathcal{A}}(\mathtt{A}_2)$ (intuitively, whether $\mathtt{A}_1$ denotes a subset of $\mathtt{A}_2$).

A CRL has the form

$$(\mathtt{crl}\ (\mathtt{canceled}\ \mathtt{c}_1, \ldots, \mathtt{c}_\mathtt{n})\ \mathtt{V})$$

where the $\mathtt{c}_i$ are hashes of certificates.[8] It is left implicit that the CRL needs to be signed by some key `k`. For simplicity, we will assume that CRLs contain certificates themselves rather than hashes. We abbreviate a CRL as $(\mathtt{crl}\ \mathtt{k}\ (\mathtt{canceled}\ \mathtt{c}_1, \ldots, \mathtt{c}_\mathtt{n})\ \mathtt{V})$. Intuitively, this says that, according to the issuer `k`, the certificates $\mathtt{c}_1, \ldots, \mathtt{c}_\mathtt{n}$ are revoked during the interval `V`. We require that each of the certificates $\mathtt{c}_i$ be revocable by `k` (otherwise `k` is attempting to revoke a certificate that it is not entitled to revoke).

Let $\mathcal{C}^+(K, N, \mathcal{A})$ consist of all certificates over $(K, N, \mathcal{A})$ (i.e., where all the keys are in K, all the names used are in N, and all the action expressions are in $\mathcal{A}$); let $\mathcal{C}(K, N, \mathcal{A})$ be the subset of $\mathcal{C}^+(K, N, \mathcal{A})$ consisting of all naming and authorization certificates; and let $\mathcal{C}_R(K, N, \mathcal{A})$ be the subset of $\mathcal{C}^+(K, N, \mathcal{A})$ consisting of all CRLs.

someone who is seeking permission to perform an action will present those certificates that maximize his/her rights.) The SPKI document is silent on this issue.

[7]Howell and Kotz (Howell and Kotz 2000) have noted some problems with intersection for SPKI's action expressions (or *tags*), namely that not all intersections of tags can be represented as a tag. The problem can be eliminated by extending the set of tags. We simply avoid the issue here by treating action expressions very abstractly, and assuming that they can always be intersected.

[8]SPKI also allows delta-CRLs. We omit these since they do not introduce essentially new semantic issues.

3 SPKI's tuple reduction rules

The semantics of SPKI certificates is characterized by a description of the algorithm invoked to verify that a sequence of credentials supports an authorization decision (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6). It is left to the prover (the agent presenting a set of credentials) to construct an appropriate sequence before submitting a request to the verifier.

Given a set C of naming and authorization certificates and a set C_R of CRLs, the algorithm first converts these certificates to a set of *tuples*, and reduces these tuples according to certain rules.

There are two types of tuples: 4-tuples, related to name binding certificates, and 5-tuples, related to authorization certificates. A 4-tuple has the form $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}\rangle$ where the components are exactly as in the first four components of a naming certificate. Similarly, a 5-tuple has the form $\langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}\rangle$, where the components are exactly as in an authorization certificate. Note that neither the 4-tuples nor the 5-tuples mention the $\mathtt{k}_r$ component of authorization certificates. We use $\tau_\mathtt{c}$ to denote the 4- or 5-tuple corresponding to certificate $\mathtt{c}$.

The first step in the conversion is to check each certificate in C to see if it has been revoked. Given a naming certificate $\mathtt{c} = (\mathtt{cert\ k\ n\ p}\ [\mathtt{t}_0, \mathtt{t}_1]\ \mathtt{k}_r)$ and a CRL $\mathtt{c}_R = (\mathtt{crl}\ \mathtt{k}'_r\ (\mathtt{canceled}\ \mathtt{c}_1, \ldots, \mathtt{c_n})\ [\mathtt{t}'_0, \mathtt{t}'_1])$, say that $\mathtt{c}$ is *live* with respect to $\mathtt{c}_R$ if

1. $\mathtt{c}$ is signed by $\mathtt{k}$, and
2. the following four conditions all hold:
   (a) $\mathtt{k}_r = \mathtt{k}'_r$,
   (b) $\mathtt{c}_R$ is signed by $\mathtt{k}'_r$,
   (c) $[\mathtt{t}_0, \mathtt{t}_1] \cap [\mathtt{t}'_0, \mathtt{t}'_1] \neq \emptyset$,
   (d) $\mathtt{c} \notin \{\mathtt{c}_1, \ldots, \mathtt{c}_n\}$.

Intuitively, $\mathtt{c}$ is live with respect to $\mathtt{c}_R$ if $\mathtt{c}$ is properly signed, the validity component in $\mathtt{c}$ requires checking a CRL, $\mathtt{c}_R$ is the certificate appropriate for the CRL and, according to the CRL, $\mathtt{c}$ has not been revoked. If $\mathtt{c}$ is live with respect to $\mathtt{c}_R$, define $\tau(\mathtt{c}, \mathtt{c}_R)$ to be the 4-tuple $(\mathtt{k\ n\ p}\ [\mathtt{t}_0, \mathtt{t}_1] \cap [\mathtt{t}'_0, \mathtt{t}'_1])$. If $\mathtt{c}$ is an authorization certificate, there is an essentially identical notion of liveness with respect to $\mathtt{c}_R$ and corresponding 5-tuples $\tau(\mathtt{c}, \mathtt{c}_R)$. We leave details to the reader. Define $\mathtt{Tuples}(C, C_R)$ to be the set of tuples $\tau(\mathtt{c}, \mathtt{c}_R)$ where $\mathtt{c} \in C$, $\mathtt{c}_R \in C_R$, and $\mathtt{c}$ is live with respect to $\mathtt{c}_R$, together with the set of tuples $\tau_\mathtt{c}$ where $\mathtt{c} \in C$ is irrevocable.

The mapping $\tau(\mathtt{c}, \mathtt{c}_R)$ is our attempt to capture the mapping described in (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6), which says:

> Individual certificates are verified by checking their signatures and possibly performing other work. They are then mapped to intermediate forms, called "tuples" here. The other work for SPKI or SDSI certificates might include processing of on-line test results (CRL, re-validation or one-time validation). ... If on-line tests are involved in the certificate processing, then the validity dates of those on-line test results are intersected ... with the validity dates of the certificate to yield the dates in the certificate's tuple(s).

Note that the mapping $\texttt{Tuples}(C, C_R)$ is monotonic: if $C' \supseteq C$, $C'_R \supseteq C_R$, then $\texttt{Tuples}(C', C'_R) \supseteq \texttt{Tuples}(C, C_R)$. Intuitively, a certificate $\texttt{c} \in C$ that is revocable by $\texttt{k}_r$ is considered to be valid at time $\texttt{t}$ if there is clear evidence that $\texttt{c}$ has not been revoked by $\texttt{k}_r$ at time $\texttt{t}$, where the "evidence" is that there is a CRL issued by $\texttt{k}_r$ that covers time $\texttt{t}$ that does not mention $\texttt{c}$. The absence of a certificate $\texttt{c}$ from *any* relevant CRL $\texttt{c}'$ ensures that the statement being made by that certificate applies during the intersection of the intervals of $\texttt{c}$ and $\texttt{c}'$.

The intuition that $\texttt{Tuples}(C, C_R)$ consists of the still valid certificates does not hold up so well when there can be more than one CRL relevant to the validity of a certificate $\texttt{c}$ at a time $\texttt{t}$. For suppose that $\texttt{c}$ is revoked according to one certificate and not revoked according to another. One could reasonably argue that, in this situation, the two CRLs are in conflict about whether the certificate has been revoked, and either could apply. In particular, the outcome of an authorization decision would depend on which CRL is presented in support of a request. To avoid such nondeterminism, SPKI (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 5.2) assumes that at any time, at most one CRL applies. We say a set C_R of CRLs is *consistent* if it is not the case that there exist CRLs $\texttt{c}, \texttt{c}' \in C_R$, both issued by $\texttt{k}$, with validity periods $\texttt{V}$, $\texttt{V}'$, respectively, such that $\texttt{V} \cap \texttt{V}' \neq \emptyset$. Restricting to consistent sets of CRLs ensures that it is safe to take any relevant CRL not containing a certificate as evidence for the validity of that certificate, since there cannot exist a CRL contradicting this conclusion. This assumption supports the monotonicity of $\texttt{Tuples}(C, C_R)$, and is essentially what allows us to use a monotonic logic, even in the presence of revocation.[9]

The semantics of SPKI given in (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a) is in terms of tuple reduction. However, the presentation of how the tuples are intended to be used to make an authorization decision is not completely formal. (SPKI Working Group 1998, Section 6) states that "Uses of names are replaced with simple definitions (keys ...), based on the name definitions available from reducing name 4-tuples" and that "Authorization 5-tuples are then reduced to a final authorization decision". The rule for 5-tuple reduction required for the latter step is explicitly described (in (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6.3)); it combines two 5-tuples to produce another 5-tuple:

[9] Of course, in practice, it may well be that a set of CRLs is inconsistent. Both the SPKI document and our paper are silent on what to do in this case.

R1. $\langle \mathtt{k}_1, \mathtt{k}_2, \mathit{true}, \mathtt{A}_1, \mathtt{V}_1\rangle + \langle \mathtt{k}_2, \mathtt{p}, \mathtt{D}_2, \mathtt{A}_2, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}_1, \mathtt{p}, \mathtt{D}_2, \mathtt{A}_1 \cap \mathtt{A}_2, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.[10]

Intuitively, if $\mathtt{k}_1$ permits $\mathtt{k}_2$ to delegate authority to the actions in $\mathtt{A}_1$ during $\mathtt{V}_1$ and $\mathtt{k}_2$ gives $\mathtt{p}$ authority over the actions in $\mathtt{A}_2$ (and to further delegate authority, if $\mathtt{D}_2$ is *true*) for the interval $\mathtt{V}_2$, then this is tantamount to $\mathtt{k}_1$ giving authority to $\mathtt{p}$ over the actions in $\mathtt{A}_1 \cap \mathtt{A}_2$ (and to further delegate authority, if $\mathtt{D}_2$ is *true*) for the interval $\mathtt{V}_1 \cap \mathtt{V}_2$.

The way that 4-tuples are to be reduced is slightly less transparent. The discussion of 4-tuple reduction in (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6.4) does not describe rules by which 4-tuples may be reduced, but rather shows how fully qualified names may be rewritten using 4-tuples. However, the discussion suggests the following rule for 4-tuple reduction:

R2. $\langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}_2\text{'s}\,\mathtt{m}\text{'s}\,\mathtt{p}, \mathtt{V}_1\rangle + \langle \mathtt{k}_2, \mathtt{m}, \mathtt{k}_3, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}_3\text{'s}\,\mathtt{p}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

We allow $\mathtt{p}$ to be the empty string in this rule, treating an expression of the form $\mathtt{r}\text{'s}\,\mathtt{p}$ as equal to $\mathtt{r}$ in this case, to avoid the need for stating the rule that results from replacing $\mathtt{k}_2\text{'s}\,\mathtt{m}\text{'s}\,\mathtt{p}$ by $\mathtt{k}_2\text{'s}\,\mathtt{m}$ and replacing $\mathtt{k}_3\text{'s}\,\mathtt{p}$ by $\mathtt{k}_3$. We use this convention in stating other rules too. Intuitively, this rule says that if $\mathtt{k}_1\text{'s}\,\mathtt{n}$ is bound to $\mathtt{k}_2\text{'s}\,\mathtt{m}\text{'s}\,\mathtt{p}$ for the interval $\mathtt{V}_1$, and $\mathtt{k}_2\text{'s}\,\mathtt{m}$ is bound to $\mathtt{k}_3$ for the interval $\mathtt{V}_2$, then $\mathtt{k}_1\text{'s}\,\mathtt{n}$ will be bound to $\mathtt{k}_3\text{'s}\,\mathtt{p}$ for the interval $\mathtt{V}_1 \cap \mathtt{V}_2$.

The SPKI document also considers a generalization of this rule:

R2′. $\langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}_2\text{'s}\,\mathtt{m}\text{'s}\,\mathtt{p}, \mathtt{V}_1\rangle + \langle \mathtt{k}_2, \mathtt{m}, \mathtt{q}, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}_1, \mathtt{n}, \mathtt{q}\text{'s}\,\mathtt{p}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

We consider in Section 5 the role of R2 vs. R2′.

The step of the authorization decision process described as "Uses of names are replaced with simple definitions (keys ...), based on the name definitions available from reducing name 4-tuples" is not further formalized. However, the following rule seems to capture this intuition:

R3. $\langle \mathtt{k}_1, \mathtt{k}_2\text{'s}\,\mathtt{n}\text{'s}\,\mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1\rangle + \langle \mathtt{k}_2, \mathtt{n}, \mathtt{k}_3, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}_1, \mathtt{k}_3\text{'s}\,\mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

Again, it is possible to generalize R3 much the same way as R2′ generalizes R2.

R3′. $\langle \mathtt{k}_1, \mathtt{k}_2\text{'s}\,\mathtt{n}\text{'s}\,\mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1\rangle + \langle \mathtt{k}_2, \mathtt{n}, \mathtt{q}, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}_1, \mathtt{q}\text{'s}\,\mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

As we shall see, the question of whether we use R2/R3 or R2′/R3′ has a nontrivial impact on the type of conclusions we can draw using tuple reduction; see, for example, Theorems 5.2 and 5.6.

[10]SPKI uses `Vintersect` to denote the intersection of timing expressions; we use the simple $\cap$ symbol here. The intersection of timing intervals is defined in the obvious way. If $\mathtt{V}_1 \cap \mathtt{V}_2$ is empty, then $\mathtt{V}_1 \cap \mathtt{V}_2 = \emptyset$, since we are using $\emptyset$ to denote the empty interval.

SPKI intends that the tuple reduction rules play several different roles. Besides being used for specific authorization decisions, the tuple reduction process is intended to serve as a means of derivation of consequences of a set of certificates that can form the basis of a "certificate result certificate" that captures a set of authorizations that may be derived from a collection of certificates (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a, Section 6.6). As we shall see, in a precise sense, R1–3 suffice for making specific authorization decisions at a given time. However, to derive more general consequences of a set of certificates, we need to consider R2′ and R3′, as well as other rules discussed in Section 5. In the next section, we provide a semantics for SPKI that lets us provide semantics for the reduction rules; we then use that semantics in Section 5 to carefully examine the rules.

4 A logic for reasoning about SPKI

We now define a formal language $\mathcal{L}_{SPKI}(K, N, \mathcal{A})$ for reasoning about SPKI. $\mathcal{L}_{SPKI}(K, N, \mathcal{A})$ is an extension of the language LLNC defined in (Halpern and van der Meyden 2001). The parameters N and $\mathcal{A}$ (the set of names and the set of action expressions) do not play a significant role. However, for some of our results, the cardinality of the set K does play a role. To simplify the notation, we often omit the parameters that play no significant role, and write, for example, $\mathcal{L}_{SPKI}$ or $\mathcal{L}_{SPKI}(K)$. We do the same in all other contexts where these parameters are used.

4.1 Syntax

Following (Halpern and van der Meyden 2001), given a set K of keys and a set N of local names, we define a *principal expression* (over K and N) to be either a key in K, a local name in N, or an expression of the form $\mathtt{p}$'s $\mathtt{q}$ where $\mathtt{p}$ and $\mathtt{q}$ are principal expressions. The compound names of SPKI/SDSI can be viewed as principal expressions. Note that parenthesization matters for principal expressions; for example, $(\mathtt{n}_1$'s $\mathtt{n}_2)$'s $\mathtt{n}_3$ is different from $\mathtt{n}_1$'s $(\mathtt{n}_2$'s $\mathtt{n}_3)$. However, our semantics guarantees that the combination of names is associative, so that, in fact, the two principal expressions are equivalent (see Lemma 4.2. For definiteness, we assume that all principal expressions that arise in naming and authorization certificates are parenthesized to the right.[11]

The primitives of $\mathcal{L}_{SPKI}(K, N, \mathcal{A})$ consist of

- principal expressions over K and N;
- the set $\mathcal{C}^+(K, N, \mathcal{A})$ of naming, authorization and revocation certificates that can be formed from K, N, and $\mathcal{A}$;

[11] Recall that, in SPKI, names are just written as `(name n1 ... nk)`, so there is no parenthesization involved at all.

- a special constant $\texttt{now}$;
- *validity intervals* $\texttt{V}$ consisting of pairs $[\texttt{t}_1, \texttt{t}_2]$ of times in $I\!N \cup \{\infty\}$ with $\texttt{t}_1 \leq \texttt{t}_2$, together with the empty interval $\emptyset$.

The set of formulas of $\mathcal{L}_{SPKI}(K, N, \mathcal{A})$ is the smallest set such that

1. if $\texttt{p}$ and $\texttt{q}$ are principal expressions then $\texttt{p} \longmapsto \texttt{q}$ is a formula;
2. if $\texttt{c} \in \mathcal{C}(K, N, \mathcal{A})$, then $\texttt{c}$ and $applic(\texttt{c})$ are formulas;
3. if $\texttt{k}$ is a key, $\texttt{p}$ is a principal expression, and $A \in \mathcal{A}$, then $Perm(\texttt{k}, \texttt{p}, A)$ and $Del(\texttt{k}, \texttt{p}, A)$ are formulas;
4. $\texttt{now} \in V$ is a formula;
5. if ϕ, ψ are formulas then $\neg\phi$ and $\phi \wedge \psi$ are formulas.

Intuitively, $\texttt{p} \longmapsto \texttt{q}$ says that all the keys in $\texttt{q}$ are bound to $\texttt{p}$. Since principal expressions are associated with sets of keys, this just says that the keys associated with $\texttt{q}$ are a subset of those associated with $\texttt{p}$. The formula $\texttt{c}$ is true at a time $\texttt{t}$ if the certificate $\texttt{c}$ was issued before $\texttt{t}$. (To make sense of this, the semantic object which determines whether formulas are true must include a list of the certificates that have been issued and the current time.) The formula $applic(\texttt{c})$ is true at a time $\texttt{t}$ if $\texttt{c}$ is either irrevocable or if it is revocable, but is known not to have been revoked at time $\texttt{t}$. Although we read $applic(\texttt{c})$ as "$\texttt{c}$ is applicable", it is worth noting that $applic(\texttt{c})$ could be true at time $\texttt{t}$ even if $\texttt{c}$ was not issued before time $\texttt{t}$ or its validity interval does not include $\texttt{t}$. There are other formulas in the logic that enable us to say that $\texttt{c}$ has been issued (namely, the formula $\texttt{c}$) and that the current time is in a given validity interval (namely, $\texttt{now} \in V$). Finally, as the notation suggests, the formula $Perm(\texttt{k}, \texttt{p}, A)$ says that $\texttt{k}$ permits $\texttt{p}$ to perform the actions in A and $Del(\texttt{k}, \texttt{p}, A)$ says that $\texttt{k}$ permits $\texttt{p}$ to delegate authority over the actions in A.

LLNC can be viewed as the fragment of $\mathcal{L}_{SPKI}$ where the only certificates allowed are those of the form $(\texttt{cert k n p})$, which corresponds to the LLNC formula $\texttt{k}\ cert\ \texttt{n} \longmapsto \texttt{p}$. There are no formulas in LLNC of the form $\texttt{now} \in \texttt{V}$, $Perm(\texttt{k}, \texttt{p}, A)$, $Del(\texttt{k}, \texttt{p}, A)$, or $applic(\texttt{c})$ (since there is no notion of time in LLNC, and permission, delegation, and revocation are not treated).[12]

[12]LLNC does allow formulas of the form $\texttt{k}\ cert\ \phi$ for arbitrary formulas ϕ. However, if ϕ is not of the form $\texttt{n} \longmapsto \texttt{p}$, then such formulas do not interact with the other constructs under the semantics of (Halpern and van der Meyden 2001). Thus, LLNC does not gain additional expressive power from such formulas.

Following SDSI, LLNC also has a notion of a *global name*. Since global names have been omitted in SPKI, we omit them in $\mathcal{L}_{SPKI}$ as well.

4.2 Semantics

The semantics for $\mathcal{L}_{SPKI}$ extends that of LLNC. We begin by outlining the main components of the semantic model.

In LLNC, there is a notion of a *world*. A world essentially describes which certificates have been issued. Since now we have time in the picture, we need a temporal analogue of a world. This is a run. Formally, a $(K, N, \mathcal{A})$-*run* is a function $r : \mathbb{N} \rightarrow \mathcal{P}(\mathcal{C}^+(K, N, \mathcal{A}))$. (We use $\mathcal{P}(X)$ to denote the set of subsets of X here and elsewhere). We are implicitly assuming a global clock and are taking time with respect to that global clock. Intuitively, $r(\mathtt{t})$ is the set of appropriately signed certificates issued at time $\mathtt{t}$. That is, if $\mathtt{c}$ is a certificate, then $\mathtt{c} \in r(\mathtt{t})$ if a certificate with contents $\mathtt{c}$ is issued at time $\mathtt{t}$ in run r. For compatibility with the SPKI document, we assume that the set of revocation certificates issued in r (that is, the set of revocation certificates in $\cup_{\mathtt{t} \in \mathbb{N}} r(\mathtt{t})$) is consistent: there cannot be two CRLs with the same issuer whose validity intervals overlap.

As we said before, we assume that there is a set $\mathcal{A}$ of action expressions, which represent sets of actions in a set *Act*, and a fixed action interpretation $\alpha_{\mathcal{A}}$ that maps expressions in $\mathcal{A}$ to subsets of *Act*.

To interpret local names, LLNC has a construct called a *local name assignment* that associates with each key $\mathtt{k}$ and local name $\mathtt{n}$ the set of keys bound to $\mathtt{n}$ by $\mathtt{k}$. There is an analogous function here, but it now takes a time as an argument, since the association may vary over time. In addition, to take into account the new constructs in $\mathcal{L}_{SPKI}$, there is a function that associates with each key $\mathtt{k}$, local name $\mathtt{n}$, and time $\mathtt{t}$ the set of actions that $\mathtt{k}$ has granted each other principal permission to perform, and describes whether or not that permission can be delegated. These functions can be extended from local names to all principal expressions; see below.

Formally,

- a *(temporal) local name assignment* (for K and N) is a function $L : K \times N \times \mathbb{N} \rightarrow \mathcal{P}(K)$. Intuitively, for $\mathtt{k} \in K$, $\mathtt{n} \in N$ and $\mathtt{t} \in \mathbb{N}$, the set $L(\mathtt{k}, \mathtt{n}, \mathtt{t})$ contains the keys associated at time $\mathtt{t}$ with the name $\mathtt{n}$ in $\mathtt{k}$'s name space.

- a *(temporal) permission/delegation assignment* (for K and *Act*) is a function $P : K \times \mathbb{N} \times K \times Act \rightarrow \{0, 1, 2\}$ such that if $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_2, \mathtt{a}) = 2$ and $P(\mathtt{k}_2, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) = i$, then $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) \geq i$. Intuitively, $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a})$ is 0 if at time $\mathtt{t}$, $\mathtt{k}$ has not granted $\mathtt{k}'$ the right to perform or delegate $\mathtt{a}$; it is 1 if principal $\mathtt{k}$ has granted permission to principal $\mathtt{k}'$ to perform action $\mathtt{a}$; it is 2 if, in addition, principal $\mathtt{k}$ has delegated authority to principal $\mathtt{k}'$ to propagate the right to perform action $\mathtt{a}$.[13] The meaning of the right to propagate is captured by the condition above: if, at time $\mathtt{t}$, $\mathtt{k}_1$ has granted $\mathtt{k}_2$ the right to propagate permission to perform $\mathtt{a}$, and $\mathtt{k}_2$ has granted

[13] As noted in (SPKI Working Group 1998), there is not much point to having a principal able to propagate the right to perform an action without having the right to perform it, since the principal may always grant itself that right.

permission to perform (propagate) $\mathtt{a}$, then, according to $\mathtt{k}_1$, principal $\mathtt{k}_3$ has the right to perform (propagate) $\mathtt{a}$. The reason that the condition says $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) \geq i$ rather than $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) = i$ is that if $i = 1$, for example, it is possible that $\mathtt{k}_1$ independently granted $\mathtt{k}_3$ the right to delegate $\mathtt{a}$, so that $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) = 2$.

A (K, N, Act) *interpretation* π is a tuple $\langle L, P \rangle$ consisting of a local name assignment L for K and N and a permission/delegation assignment P for K and Act. We omit the modifier (K, N, Act) when it is not relevant to the discussion. However, it is important to note that the parameters that characterize the language also characterize the interpretations.

Given a local name assignment L, a key $\mathtt{k}$, and a time $\mathtt{t} \in I\!N$, we can assign to each principal expression $\mathtt{p}$ a set of keys $[\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. This set is defined by the following recursion:

- $[\![\mathtt{k}']\!]_{L,\mathtt{k},\mathtt{t}} = \{\mathtt{k}'\}$, if $\mathtt{k}' \in K$ is a key,
- $[\![\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}} = L(\mathtt{k}, \mathtt{n}, \mathtt{t})$, if $\mathtt{n} \in N$ is a local name,
- $[\![\mathtt{p}\text{'s}\, \mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}} = \bigcup\{[\![\mathtt{q}]\!]_{L,\mathtt{k}',\mathtt{t}} \mid \mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}\}$.

This definition is essentially identical to that in (Abadi 1998; Halpern and van der Meyden 2001), except that we have made the interpretation of local names depend on the time of evaluation.

It is now easy to prove some basic facts about principal expressions. First, we can show that a fully-qualified name is independent of the key.

Lemma 4.1: *If* $\mathtt{p}$ *is a fully qualified principal expression, then* $[\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}} = [\![\mathtt{p}]\!]_{L,\mathtt{k}',\mathtt{t}}$ *for all keys* $\mathtt{k}$ *and* $\mathtt{k}'$*.*

Proof: By an easy induction on the structure of $\mathtt{p}$. ∎

We also make precise our claim that the combination of names is associative.

Lemma 4.2: *For all principal expressions* $\mathtt{p}_1$*,* $\mathtt{p}_2$*, and* $\mathtt{p}_3$*, keys* $\mathtt{k}$*, and local name assignment* L*,*

$$[\![\mathtt{p}_1\text{'s}\,(\mathtt{p}_2\text{'s}\,\mathtt{p}_3)]\!]_{L,\mathtt{k},\mathtt{t}} = [\![(\mathtt{p}_1\text{'s}\,\mathtt{p}_2)\text{'s}\,\mathtt{p}_3]\!]_{L,\mathtt{k},\mathtt{t}}.$$

Proof: By unwinding the definitions, it immediately follows that both $[\![\mathtt{p}_1\text{'s}\,(\mathtt{p}_2\text{'s}\,\mathtt{p}_3)]\!]_{L,\mathtt{k},\mathtt{t}}$ and $[\![(\mathtt{p}_1\text{'s}\,\mathtt{p}_2)\text{'s}\,\mathtt{p}_3]\!]_{L,\mathtt{k},\mathtt{t}}$ are equal to

$$\cup\{[\![\mathtt{p}_3]\!]_{L,\mathtt{k}_2,\mathtt{t}} : \mathtt{k}_2 \in [\![\mathtt{p}_2]\!]_{L,\mathtt{k}_1,\mathtt{t}}, \mathtt{k}_1 \in [\![\mathtt{p}_1]\!]_{L,\mathtt{k},\mathtt{t}}\}.$$

∎

In order to capture the impact that CRLs have on the interpretation of naming and authorization certificates, we say that a certificate $\mathtt{c}$ is *applicable* at time $\mathtt{t}$ in a run r

if either $\mathtt{c}$ is not revocable, or $\mathtt{c}$ is revocable by a key $\mathtt{k}_r$ and for some $\mathtt{t}' \leq \mathtt{t}$ we have $(\mathtt{crl\ k}_r\ (\mathtt{canceled\ c_1, \ldots, c_n})\ \mathtt{V}) \in r(\mathtt{t}')$ for some $(\mathtt{c_1, \ldots, c_n})$ such that $\mathtt{c}$ is not one of the $\mathtt{c}_i$, and $\mathtt{t} \in \mathtt{V}$. Roughly speaking, this says that $\mathtt{c}$ is applicable at time $\mathtt{t}$ if has been declared not to have been revoked at $\mathtt{t}$ (under the assumption that at most one CRL is applicable at any given time). Note that a certificate $\mathtt{c}$ may be applicable at a time $\mathtt{t}$ outside its validity interval.

We now define what it means for a formula ϕ to be true at a run r with respect to an interpretation $\pi = \langle L, P\rangle$, a key $\mathtt{k}$, and a time $\mathtt{t}$, written $r, \pi, \mathtt{k}, \mathtt{t} \models \phi$, by induction on the structure of ϕ:

- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathtt{p} \longmapsto \mathtt{q}$ if $[\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}}$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathtt{c}$ if $\mathtt{c} \in r(\mathtt{t}')$ for some $\mathtt{t}' \leq \mathtt{t}$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathit{Perm}(\mathtt{k}_1, \mathtt{p}, A)$ if for all $\mathtt{k}_2 \in [\![\mathtt{p}]\!]_{L,\mathtt{k}_1,\mathtt{t}}$ and all $\mathtt{a} \in \alpha_{\mathcal{A}}(A)$, we have $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_2, \mathtt{a}) \geq 1$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathit{Del}(\mathtt{k}_1, \mathtt{p}, A)$ if for all $\mathtt{k}_2 \in [\![\mathtt{p}]\!]_{L,\mathtt{k}_1,\mathtt{t}}$ and all $\mathtt{a} \in \alpha_{\mathcal{A}}(A)$, we have $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_2, \mathtt{a}) = 2$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathtt{now} \in V$ if $\mathtt{t} \in V$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \mathit{applic}(\mathtt{c})$ if $\mathtt{c}$ is applicable at time $\mathtt{t}$ in r,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \phi \wedge \psi$ if $r, \pi, \mathtt{k}, \mathtt{t} \models \phi$ and $r, \pi, \mathtt{k}, \mathtt{t} \models \psi$,
- $r, \pi, \mathtt{k}, \mathtt{t} \models \neg\phi$ if not $r, \pi, \mathtt{k}, \mathtt{t} \models \phi$.

We write $r, \pi \models \phi$ if $r, \pi, \mathtt{k}, \mathtt{t} \models \phi$ for all principals $\mathtt{k} \in K$ and all times $\mathtt{t} \in I\!N$.

In the definitions so far, there is no connection between the run and the interpretation. Intuitively, we would like the interpretation, which contains information about the meaning of local names and permissions and delegations, to be determined from the information about the certificates that have been issued at each point in time that is represented in the run. We connect these ideas by means of the following definition. The interpretation $\pi = \langle L, P\rangle$ is *consistent* with a run r if, for all times $\mathtt{t} \in I\!N$,

1. for all naming certificates $\mathtt{c} = (\mathtt{cert\ k\ n\ p\ V}\ \langle \mathtt{k}_r\rangle)$ in $\cup_{\mathtt{t}' \leq \mathtt{t}}\, r(\mathtt{t}')$, if $\mathtt{t} \in V$ and $\mathtt{c}$ is applicable at $\mathtt{t}$ in r, then $[\![\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$;
2. for all authorization certificates $\mathtt{c} = (\mathtt{cert\ k\ p\ D\ A\ V}\ \langle \mathtt{k}_r\rangle)$ in $\cup_{\mathtt{t}' \leq \mathtt{t}}\, r(\mathtt{t}')$, if $\mathtt{t} \in V$ and $\mathtt{c}$ is applicable at $\mathtt{t}$ in r, then for all $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ and all keys $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$, we have
   (a) $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) \geq 1$,
   (b) if $\mathtt{D} = \mathtt{true}$ then $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) = 2$.

In general, an interpretation can be consistent with a run while allowing facts to hold that do not follow from the certificates issued in the run. For an extreme example of this, let r be the run in which no certificates are ever issued, and suppose that π is the *maximal interpretation*, where $L(\mathtt{k},\mathtt{n},\mathtt{t}) = K$ and $P(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) = 2$ for all keys $\mathtt{k}$, $\mathtt{k}'$, local names $\mathtt{n}$, actions $\mathtt{a}$, and times $\mathtt{t}$. Then π is consistent with r. This is undesirable. Intuitively, we would like facts concerning rights and the meaning of local names to hold only if they are forced to do so by some certificate. To enforce this, we restrict the interpretation to be the minimal one consistent with the run. Our technique for doing so extends that used in (Halpern and van der Meyden 2001).

Formally, define an order $\leq$ on (K, N, Act) interpretations by $\langle L, P, \rangle \leq \langle L', P' \rangle$ if, for all keys $\mathtt{k}$, local names $\mathtt{n}$, and times $\mathtt{t}$, we have $L(\mathtt{k},\mathtt{n},\mathtt{t}) \subseteq L'(\mathtt{k},\mathtt{n},\mathtt{t})$ and for all keys $\mathtt{k}'$ and actions $\mathtt{a}$, we have $P(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) \leq P'(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a})$. Thus, $\langle L, P \rangle \leq \langle L', P' \rangle$ if, for all $\mathtt{n}$, $\mathtt{k}$, and $\mathtt{t}$, at least as many keys are bound to $\mathtt{n}$ by $\mathtt{k}$ at time $\mathtt{t}$ in L' as in L. In addition, at least as many keys are authorized to perform action $\mathtt{a}$ by $\mathtt{k}$ at time $\mathtt{t}$ in P' as in P and, of those keys authorized to perform the action $\mathtt{a}$, at least as many can delegate that authority in P' as in P.

This order can easily be seen to give the set of (K, N, Act) interpretations the structure of a lattice. Say that an element π of a set S of interpretations is *minimal in S* if $\pi \leq \pi'$ for all $\pi' \in S$.

Proposition 4.3: *For every run r, there exists a unique (K, N, Act) interpretation minimal in the set of (K, N, Act) interpretations consistent with r.*

Proof: See the appendix. ∎

We write π_r for the minimal interpretation consistent with r. Intuitively, in the minimal interpretation consistent with r, there are no name bindings, permissions, or delegations that are not forced by r. Enforcing the requirement that the interpretation should be the minimal one consistent with the run leads to a variant of the semantics discussed above. We write $r, \mathtt{k}, \mathtt{t} \models_c \phi$ if $r, \pi_r, \mathtt{k}, \mathtt{t} \models \phi$. We say that a formula ϕ is *cl-valid* (with respect to K, N, Act) in $\mathcal{L}_{SPKI}(K, N, Act)$, written $\models_{cl,K,N,Act} \phi$, if $r, \mathtt{k}, \mathtt{t} \models_c \phi$ for all (K, N, Act)-runs r, keys $\mathtt{k} \in K$, and times $\mathtt{t}$. (The "cl" stands for closed, since in (Halpern and van der Meyden 2001) the semantics corresponding to $\models_{cl}$ is termed the *closed semantics*, while the semantics corresponding to $\models$ was termed the *open semantics*. We use "cl" here rather than "c" as in (Halpern and van der Meyden 2001) to denote the closed semantics, to avoid confusion with $\mathtt{c}$, which ranges over certificates.) The closed semantics is the one of most interest to us here, since it enforces the desired close connection between the certificates actually issued and the name bindings, permissions, and delegations. The open semantics is mainly used as a stepping-stone to defining the closed semantics. Validity with respect to the open semantics can also be viewed as capturing what is guaranteed to hold, no matter what additional certificates are issued. Interestingly, as shown in (Halpern and van der Meyden 2001), validity with respect to the open and closed semantics coincide for the logic LLNC. We believe that they also coincide for the logic $\mathcal{L}_{SPKI}$, although we have not checked carefully.

It is not hard to check that the subscripts N and Act play no role in $\models_{cl,K,N,\mathcal{A}}$; if a formula is valid for some choice of N and Act (for a fixed K), then it is valid for all choices of N and Act. However, as observed in (Halpern and van der Meyden 2001) (where a complete axiomatization for LLNC was provided), the axioms do depend on the choice of K; in particular, they depend on the cardinality of K. For example, suppose K has just one element, say $\mathtt{k}$. Then it is easy to see that $\models_{cl,K,N,Act} \mathtt{n} \longmapsto \mathtt{k} \Rightarrow \mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}\text{'s}\,\mathtt{m}$, for all $\mathtt{n}, \mathtt{m} \in N$. Since $\mathtt{n}$ is bound to $\mathtt{k}$, it follows that the interpretation of $\mathtt{k}\text{'s}\,\mathtt{n}$ must be $\{\mathtt{k}\}$, since $\mathtt{k}$ is the only key. Thus, the interpretation of $\mathtt{k}\text{'s}\,\mathtt{n}$ must be a superset of interpretation of $\mathtt{k}\text{'s}\,\mathtt{m}$, and so $\mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}\text{'s}\,\mathtt{m}$ holds. However, this argument depends critically on the assumption that $K = \{\mathtt{k}\}$. The formula $\mathtt{n} \longmapsto \mathtt{k} \Rightarrow \mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}\text{'s}\,\mathtt{m}$ is not valid if there are at least two keys: If $\mathtt{k}'$ is a key distinct from $\mathtt{k}$, then it is possible that $\mathtt{k}'$ is bound to $\mathtt{k}\text{'s}\,\mathtt{m}$ and not bound to $\mathtt{k}\text{'s}\,\mathtt{n}$. In light of this discussion, we omit N and Act from the subscript from here on in, but include K when it plays an important role.

We can now make precise the intuition that in the minimal interpretation only name bindings and permissions and delegations forced by r hold. Define the *formula associated with the naming certificate* $\mathtt{c} = (\mathtt{cert\ k\ n\ p\ V}\ \langle \mathtt{k}_r \rangle)$ to be

$$\mathtt{now} \in V \Rightarrow (\mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{p}).$$

Similarly, the *formula associated with the authorization certificate* $\mathtt{c} = (\mathtt{cert\ k\ p\ D\ A\ V}\ \langle \mathtt{k}_r \rangle)$ is

$$\mathtt{now} \in V \Rightarrow [Perm(\mathtt{k}, \mathtt{p}, \mathtt{A}) \land (\mathtt{D} \Rightarrow Del(\mathtt{k}, \mathtt{p}, \mathtt{A}))].$$

Let $\phi_{\mathtt{c}}$ be the formula associated with certificate $\mathtt{c} \in \mathcal{C}$.

Proposition 4.4: *The interpretation π is consistent with a run r iff for all times $\mathtt{t} \in I\!N$, keys $\mathtt{k} \in K$, and $\mathtt{c} \in \mathcal{C}$, we have $r, \pi, \mathtt{k}, \mathtt{t} \models \mathtt{c} \land applic(\mathtt{c}) \Rightarrow \phi_{\mathtt{c}}$.*

Proof: Immediate from the definition of consistency. ∎

Note that it follows from Proposition 4.4 that if π is an interpretation consistent with run r and a certificate $\mathtt{c} \in \mathcal{C}$ was issued in r at or before time $\mathtt{t}$ and remains applicable at $\mathtt{t}$, then $r, \pi, \mathtt{k}, \mathtt{t} \models \phi_c$. Moreover, the minimal interpretation associated with r is the minimal intepretation π satisfying $r, \pi, \mathtt{k}, \mathtt{t} \models \phi_c$ for all certificates $\mathtt{c} \in \mathcal{C}$ that have been issued in r at or before time $\mathtt{t}$ and remain applicable at $\mathtt{t}$. In this sense, the formulas associated with certificates precisely capture their meaning. It is also worth noting that $\models_{cl} \mathtt{c} \land applic(\mathtt{c}) \Rightarrow \phi_{\mathtt{c}}$ for all $\mathtt{c} \in \mathcal{C}$.

5 Soundness and completeness of tuple reduction

We are now in a position to compare the SPKI tuple reduction rules to our semantics. Intuitively, we would like to understand tuple reduction as drawing inferences about the run based on evidence provided in the certificates presented to the verifier. Thus, we want to show that all conclusions based on tuple reduction are true under our semantics and, conversely, all valid conclusions about bindings and authorizations that follow from the issuing of certain certificates are derivable from those certificates using tuple reductions. More precisely, we want to show that, given a finite set $C \cup C_R$ of certificates, where C consists of naming and authorization certificates and C_R consists of CRLs, we have $\models_{cl} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$ iff $\tau_{\mathtt{c}}$ can be derived from the tuples in $\mathtt{Tuples}(C, C_R)$, using the tuple reduction rules. This can be broken up into two questions:

- *soundness*: if $\tau_{\mathtt{c}}$ can be derived from the tuples in $\mathtt{Tuples}(C, C_R)$ using the tuple reduction rules, then $\models_{cl} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$;
- *completeness*: if $\models_{cl} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$, then $\tau_{\mathtt{c}}$ can be derived from the tuples in $\mathtt{Tuples}(C, C_R)$ using the tuple reduction rules.

Of course, whether soundness and completeness hold depend in large part on *which* reduction rules are used. We are particularly interested in questions such as whether R2′ and R3′ are needed to derive all conclusions of interest about certificates and, if not, what conclusions they can be used to derive.

Showing that the tuple reduction rules discussed earlier are sound with respect to the $\mathcal{L}_{SPKI}$ semantics is straightforward. To make this precise, if T is a set of tuples and τ is a tuple, we write $T \longrightarrow_0^* \tau$ if there exists a sequence of tuples $\tau_1, \ldots, \tau_k$ such that $\tau_k = \tau$ and for each $i \leq k$ either $\tau_i \in T$ or there exist $j, j' < i$ such that $\tau_j + \tau_{j'} \longrightarrow \tau_i$ is an instance of one of R1, R2, or R3.

Theorem 5.1: *Suppose that C is a finite subset of $\mathcal{C}$, C_R is a finite set of CRLs, and $\mathtt{c} \in \mathcal{C}$. If $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \tau_{\mathtt{c}}$, then $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$.*

Since Theorem 5.1 is a special case of a more general soundness result, Theorem 5.3, we defer the proof until after the statement of the latter theorem.

Completeness is not at all straightforward; indeed, it does not hold for R1–R3. What does hold is a weak version of completeness. To understand this in more detail, we need a few definitions.

A *concrete certificate* is one whose corresponding tuple has the form $\langle \mathtt{k}, \mathtt{n}, \mathtt{k}', [\mathtt{t}, \mathtt{t}] \rangle$ (in the case of naming certificates) or $\langle \mathtt{k}, \mathtt{k}', \mathtt{D}, \{\mathtt{a}\}, [\mathtt{t}, \mathtt{t}] \rangle$ (in the case of authorization certificates). That is, concrete certificates talk about the keys that are bound to names and the keys that are authorized to perform certain actions, and are concerned only with a single point in time and single actions.

The following result shows that R1, R2, and R3 suffice in a certain sense to deal with concrete certificates. We say that a naming certificate with 4-tuple $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V} \rangle$ *subsumes* a naming certificate with 4-tuple $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}' \rangle$ if $\mathtt{V} \supseteq \mathtt{V}'$. Similarly an authorization certificate with 5-tuple $\langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V} \rangle$ subsumes an authorization certificate with 5-tuple $\langle \mathtt{k}, \mathtt{p}, \mathtt{D}', \mathtt{A}', \mathtt{V}' \rangle$

if $\mathtt{V} \supseteq \mathtt{V}'$ and $\alpha_{\mathcal{A}}(\mathtt{A}) \supseteq \alpha_{\mathcal{A}}(\mathtt{A}')$ and $\mathtt{D} \Rightarrow \mathtt{D}'$ is valid. (Recall that we have assumed that it is possible to tell if one action expression is a superset of another.) Clearly if $\mathtt{c}$ subsumes $\mathtt{c}'$, then $\models_{cl} \phi_{\mathtt{c}} \Rightarrow \phi_{\mathtt{c}'}$.

Theorem 5.2: *If $\mathtt{c}$ is a concrete certificate, C is a finite subset of $\mathcal{C}$, C_R is a finite set of CRLs, and $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$, then there exists a certificate $\mathtt{c}'$ that subsumes $\mathtt{c}$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \tau_{\mathtt{c}'}$.*

Proof: See the appendix. ∎

Theorem 5.2 tells us that if all we want to do is to check if a given key is currently bound to a given name or if a given key is currently authorized to perform a given action, then R1–R3 essentially suffice. (Clarke et al. (2001) establish a closely related result. They define the semantics of names operationally using just R2' (which they call *rule composition*), and then establish that all conclusions about whether a given key is bound to a given name can be derived using just R2.)

However, R1–R3 do not suffice if we want to do full-fledged reasoning about the consequences of a set of certificates. For example, in general, R1–R3 may not suffice to draw a conclusion of the form $\mathtt{k}$'s $\mathtt{n} \longmapsto \mathtt{p}$, where $\mathtt{p}$ is an arbitrary principal expression, even though it may be a logical consequence of the certificates issued. Such conclusions are of interest in the context of "certificate result certificates" ((Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a), Section 6.7), which are new certificates stating facts that are consequences of a set of certificates previously issued. Certificate result certificates help to reduce the amount of work a relying party needs to do when processing a request.

There are three impediments to getting a full completeness theorem that is a converse to Theorem 5.1. The first two are easily dealt with by adding rules.

First, we want to get rid of the restriction that allows conclusions only about keys. R2 and R3 do not suffice for this. For example, let $\mathtt{c}_1$, $\mathtt{c}_2$, and $\mathtt{c}_3$ be irrevocable naming certificates whose corresponding 4-tuples are $\langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}_2\text{'s}\,\mathtt{m}\text{'s}\,\mathtt{p}, [\mathtt{t}, \mathtt{t}] \rangle$, $\langle \mathtt{k}_2, \mathtt{m}, \mathtt{q}, [\mathtt{t}, \mathtt{t}] \rangle$, and $\langle \mathtt{k}_1, \mathtt{n}, \mathtt{q}\text{'s}\,\mathtt{p}, [\mathtt{t}, \mathtt{t}] \rangle$, respectively. Clearly $\models_{cl} \mathtt{c}_1 \wedge \mathtt{c}_2 \Rightarrow \phi_{\mathtt{c}_3}$. However, we cannot get $\tau_{\mathtt{c}_3}$ from $\tau_{\mathtt{c}_1}$ and $\tau_{\mathtt{c}_2}$, since the only rule that could possibly be of help, R2, applies only to 4-tuples whose third argument is a key. To deal with this problem, we need R2$'$. Similarly, R3$'$ is needed to deal with the analogous problem for R3. We also need the following trivial axiom to deal with a special case:

R0. $\longrightarrow \langle \mathtt{k}, \mathtt{n}, \mathtt{k}\text{'s}\,\mathtt{n}, [0, \infty] \rangle$.

(The fact that there is nothing to the left of the $\longrightarrow$ is meant to indicate that the conclusion on the right-hand side can be reached in all circumstances.)

A naming certificate is *point-valued* if its time interval has the form $[t, t]$. A concrete certificate is point-valued but a point-valued naming certificate can have as its third component an arbitrary fully-qualified expression. As we shall see, R0, R1, R2$'$, and R3$'$ essentially suffice to get us conclusions about point-valued naming certificates, but do not suffice to get us conclusions about arbitrary time intervals and sets of actions. To see that they do not suffice, consider three irrevocable naming certificates $\mathtt{c}_1$, $\mathtt{c}_2$, and $\mathtt{c}_3$ whose corresponding 4-tuples are $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, [1,2]\rangle$, $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, [3,4]\rangle$, and $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, [1,4]\rangle$. Clearly $\models_{cl} \mathtt{c}_1 \wedge \mathtt{c}_2 \Rightarrow \phi_{\mathtt{c}_3}$. However, we cannot get $\tau_{\mathtt{c}_3}$ from $\tau_{\mathtt{c}_1}$ and $\tau_{\mathtt{c}_2}$, since none of the reduction rules increase the size of intervals. Similar issues arise with 5-tuples.

There are a number of ways to deal with this. Perhaps the simplest is just to add the following reduction rule:

R4(a). $\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_1\rangle + \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_3\rangle$ if $\mathtt{V}_1 \cup \mathtt{V}_2 \supseteq \mathtt{V}_3$.

R4(b). $\langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1\rangle + \langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_2\rangle \longrightarrow \langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_3\rangle$ if $\mathtt{V}_1 \cup \mathtt{V}_2 \supseteq \mathtt{V}_3$.

R4(c). $\langle \mathtt{k}, \mathtt{p}, \mathtt{D}_1, \mathtt{A}_1, \mathtt{V}\rangle + \langle \mathtt{k}, \mathtt{p}, \mathtt{D}_2, \mathtt{A}_2, \mathtt{V}\rangle \longrightarrow \langle \mathtt{k}, \mathtt{p}, \mathtt{D}_3, \mathtt{A}_3, \mathtt{V}\rangle$ if $\mathtt{D}_3 \Rightarrow \mathtt{D}_1 \wedge \mathtt{D}_2$ is a tautology and $\alpha_{\mathcal{A}}(\mathtt{A}_1) \cup \alpha_{\mathcal{A}}(\mathtt{A}_2) \supseteq \alpha_{\mathcal{A}}(\mathtt{A}_3)$.

Note that the rule

$$\langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_1\rangle \longrightarrow \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_3\rangle \quad \text{if} \quad \mathtt{V}_1 \supseteq \mathtt{V}_3$$

is a special case of R4(a) (taking $\mathtt{V}_1 = \mathtt{V}_2$). Similar comments apply to R4(b) and R4(c). How easy it is to apply R4(c) depends on how easy it is to check that $\alpha_{\mathcal{A}}(\mathtt{A}_1) \cup \alpha_{\mathcal{A}}(\mathtt{A}_2) \supseteq \alpha_{\mathcal{A}}(\mathtt{A}_3)$. If action expressions can represent infinite sets of actions, this may be nontrivial. We do not address this issue here. Of course, it is trivial to determine if $\mathtt{V}_1 \cup \mathtt{V}_2 \supseteq \mathtt{V}_3$.

We write $T \longrightarrow_1^* \tau$ (resp., if $T \longrightarrow_2^* \tau$) if there is a derivation of τ from T using rules R0, R1, R2$'$, and R3$'$ (resp., R0, R1, R2$'$, R3$'$, and R4). There is no difficulty showing that $\longrightarrow_2^*$ is sound. Note that the soundness of $\longrightarrow_0^*$ and $\longrightarrow_1^*$ follow as special cases.

Theorem 5.3: *Suppose that C is a finite subset of $\mathcal{C}$, C_R is a finite set of CRLs, and $\mathtt{c} \in \mathcal{C}$. If $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$, then $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$. In fact, if π is an interpretation consistent with a run r and $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$, then $r, \pi \models \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$.*

Theorem 5.3 follows easily from the following two propositions, whose straightforward proof we leave to the reader. The first shows that the process of transforming a set of certificates into a set of tuples corresponds to a valid inference in the logic.

Proposition 5.4: *Suppose that $\mathtt{c}_1$ is a revocable certificate in $\mathcal{C}$, $\mathtt{c}_2$ is a CRL such that $\mathtt{c}_1$ is live with respect to $\mathtt{c}_2$, and $\mathtt{c}_3 = \tau(\mathtt{c}_1, \mathtt{c}_2)$. Then if π is an interpretation consistent with a run r and $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$, then $r, \pi \models \mathtt{c}_1 \wedge \mathtt{c}_2 \Rightarrow \phi_{\mathtt{c}_3}$. Similarly, if $\mathtt{c}$ is an irrevocable certificate in $\mathcal{C}$, then $r, \pi \models \mathtt{c} \Rightarrow \phi_{\mathtt{c}}$.*

The second proposition shows that single reductions are sound with respect to the closed semantics.

Proposition 5.5: *Suppose that* $\mathtt{c}_1$, $\mathtt{c}_2$, *and* $\mathtt{c}_3$ *are certificates and* $\tau_{\mathtt{c}_1} + \tau_{\mathtt{c}_2} \rightarrow \tau_{\mathtt{c}_3}$ *is an instance of R1, R2′, R3′, or R4. Then*

$$\models_c \phi_{\mathtt{c}_1} \wedge \phi_{\mathtt{c}_2} \Rightarrow \phi_{\mathtt{c}_3}.$$

Moreover, if $\rightarrow \tau_{\mathtt{c}}$ *is an instance of R0, then* $\models_c \phi_{\mathtt{c}}$.

As we show shortly, R4 (together with R0, R1, R2′, R3′) suffices for completeness provided that the set of keys is infinite. It does not suffice if the set of keys is finite. As we have seen, the set of valid formulas depends on the cardinality of the set K of keys. Consider the very simplest case where K consists of only one key $\mathtt{k}$. Suppose that $\mathtt{m}$ and $\mathtt{n}$ are names in N. Let $\mathtt{c}$ and $\mathtt{c}'$ be irrevocable naming certificate whose 4-tuples are $\langle \mathtt{k}, \mathtt{n}, \mathtt{k}, \mathtt{V}\rangle$ and $\langle \mathtt{k}, \mathtt{n}, \mathtt{k}\text{'s}\,\mathtt{m}, \mathtt{V}\rangle$, respectively. It is easy to see that $\models_{cl,K} \mathtt{c} \Rightarrow \phi_{\mathtt{c}'}$. This is true for essentially the same reasons that $\models_{cl,K} \mathtt{n} \longmapsto \mathtt{k} \Rightarrow \mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}\text{'s}\,\mathtt{m}$. On the other hand, $\models_{cl,K} \mathtt{c} \Rightarrow \phi_{\mathtt{c}'}$ does not hold if $|K| \geq 2$ (and, in particular, if K is infinite). It easily follows that $\tau_{\mathtt{c}'}$ is not derivable from $\{\mathtt{c}\}$. Additional derivation rules would be necessary to allow this derivation.

With a little more effort, examples like this can be given as long as K is finite. That is, if K is finite, then there exist finite sets $C \subseteq \mathcal{C}(K, N, \mathcal{A})$ and $C_R \subseteq \mathcal{C}_R(K, N, \mathcal{A})$ and a naming certificate $\mathtt{c}$ such that $\models_{cl,K} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$, but $\tau_{\mathtt{c}}$ cannot be derived from $\mathtt{Tuples}(C, C_R)$ using R0, R1, R2′, R3′, R4. Nevertheless, these reduction rules are "almost" complete. We show that, for any fixed set $C \subseteq \mathcal{C}^+(K, N, \mathcal{A})$, as long as $|K| > |C| + |\mathtt{c}|$, then for all sets C_R, we have $\models_{cl,K} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$ iff $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \mathtt{c}$. (Here $|\mathtt{c}|$ denotes the length of the certificate $\mathtt{c}$ as a string of symbols, and $|C|$ denotes the sum of the lengths of the certificates in the finite set C.)

Theorem 5.6: *If* $\mathtt{c}$ *is a certificate,* C *is a finite subset of* $\mathcal{C}$, C_R *is a finite set of CRLs,* $|K| > |C| + |\mathtt{c}|$, *and* $\models_{c,K} \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$, *then* $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$; *moreover, if* $\mathtt{c}$ *is a point-valued certificate, then* $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \tau_{\mathtt{c}'}$ *for some certificate* $\mathtt{c}'$ *subsuming* $\mathtt{c}$.

Proof: See the appendix. ∎

Basically, Theorem 5.6 says that, by using R2′, R3′, and R4 in the tuple reduction rules, we can derive all conclusions about certificates, provided that K is not "small", in the sense that $|K| \leq |C| + |\mathtt{c}|$. This proviso is not at all unreasonable. The set K is, after all, intended to model the collection of potential public keys being used by the principals. In order for such a set of keys to be secure for signatures and encryption, it needs to be a very large set, so as to render brute force attacks on encrypted messages impractical (i.e., the key length needs to be large enough). By contrast, C models a set of certificates generated by the principals and presented in a particular authorization request, and $\mathtt{c}$ is a particular certificate. We do not expect the size of these to be of the same order of magnitude as K, so we would expect that $|K| \leq |C| + |\mathtt{c}|$ will hold in practice. Another way to think about things is to understand K as modelling the set of all keys that could ever be used, if we allow the key length to be made arbitrarily large. In this case K is an infinite set, so we immediately have that $|K| \leq |C| + |\mathsf{c}|$. Thus, in a precise sense, the rules R0, R1, R2′, R3′, and R4 together give us all interesting conclusions about certificates.

6 Related Work

As mentioned in the introduction, there have been a number of other recent approaches to giving semantics to SPKI. In this section, we compare these approaches to ours.

Howell and Kotz (2000) give a semantics to names that closely resembles the semantics of Abadi (1998). In particular, a name is associated with a relation on possible worlds, rather than a set of keys as is the case in our semantics. We criticized Abadi's semantics in our earlier paper (Halpern and van der Meyden 2001); many of our criticisms apply to the Howell-Kotz semantics as well. Perhaps most significantly, the semantics for name binding (as Howell and Kotz themselves say) is rather opaque; it is hard to explain exactly what the meaning of the relation on worlds is. Their logic also uses a "speaks-for" relation $\mathtt{p} \Rightarrow \mathtt{q}$, where $\mathtt{p}, \mathtt{q}$ are principal expressions, to capture both the binding relationship between names (similar to our use of $\longmapsto$) and the notion that $\mathtt{q}$ has delegated certain rights to $\mathtt{p}$. As we argued in (Halpern and van der Meyden 2001), we believe that these are distinct notions that should be modeled using different constructs. Indeed, Howell and Kotz themselves note that their axiom

$$\mathtt{p} \Rightarrow \mathtt{q} \supset \mathtt{p}\text{'s}\,\mathtt{n} \Rightarrow \mathtt{q}\text{'s}\,\mathtt{n}$$

is "suprisingly powerful" and needs to be "tempered", particularly when one considers the related notion "speaks for on topic T". In our logic, the formula

$$\mathtt{p} \longmapsto \mathtt{q} \Rightarrow \mathtt{p}\text{'s}\,\mathtt{n} \longmapsto \mathtt{q}\text{'s}\,\mathtt{n}$$

is valid, but this is not a problem for us, since we do not interpret "$\longmapsto$" as "speaks-for".

Just as we do, Howell and Kotz translate SPKI certificates to formulas in their logic. There seem to be some problems with their translation, although there are not enough formal details in the paper for us to be really sure that these are in fact problematic.

- To capture the difference between just being given permission to perform an action and being given permission to delegate authority to perform the action, they split a real principal $\mathtt{k}$ into two principals $\mathtt{k}_u$ and $\mathtt{k}_b$; $\mathtt{k}_b$ is supposed to deal with the situation where $\mathtt{k}$ can delegate authority, while $\mathtt{k}_u$ is supposed to deal with the situation where $\mathtt{k}$ cannot delegate authority. There is no formal semantics given to $\mathtt{k}_u$ and $\mathtt{k}_b$, so it is not clear (to us) the extent to which this approach works.

- Their approach to dealing with time seems to involve talking about time explicitly in the formula. But there is no type corresponding to time in the semantics, so there is nothing to connect the statements about time to the actual time. Again, since the discussion of how time is handled is quite informal (with no examples), it is hard to tell how well it will work.

Howell and Kotz do claim that their semantics is sound with respect to tuple reduction; there is no discussion of completeness.

Aura (1998) defines the notion of a *delegation network*, which is essentially a graphical description of a collection of certificates. Although he does not have a formal logic to reason about authorization, given a delegation network DN, he does define a relation $\mathit{authorizes}_{DN}(\mathtt{k}_1, \mathtt{k}_2, \mathtt{a})$, which can be read as "$\mathtt{k}_1$ authorizes $\mathtt{k}_2$ to perform $\mathtt{a}$". Given a collection of SPKI certificates, Aura defines a corresponding authorization network DN and obtains a certain type of soundness and completeness result for reduction. Very roughly speaking, given a collection of certificates with corresponding delegation network DN, $\mathit{authorizes}_{DN}(\mathtt{k}_1, \mathtt{k}_2, \mathtt{a})$ holds iff there is a sequence of delegation networks $DN_1, \ldots, DN_m$ such that DN_{i+1} is obtained from DN_i by certificate reduction in a precise sense and in DN_m there is an explicit certificate issued by $\mathtt{k}_1$ authorizing $\mathtt{k}_2$ to perform $\mathtt{a}$. This is not a soundness and completeness result in the logical sense that we have here, although it provides what can be viewed as an operational semantics for the SPKI 5-tuple reduction rule R1. One of the main differences from our work is that, where we deal with principal expressions, Aura's framework deals with an explicitly given set of keys; there is nothing in his paper that corresponds to our discussion of name reduction. Aura also does not seem to have time or revocation in his framework, nor does he consider the delegation bit. On the other hand, he does deal with threshold principals (as do Howell and Kotz), while we do not.

Li (2000) has also considered the semantics of SPKI/SDSI. He presents a logic program, and provides results showing that this program derives the same set of concrete conclusions about SDSI names as 4-tuple reduction and SDSI's name resolution procedure REF (1996). He also provides an extension of the logic program intended to capture concrete conclusions about authorization certificates. However, he also does not explicitly treat timing and revocation, as we have done, and does not consider general reasoning such as that captured by our rules. On the other hand, he does discuss threshold subjects, which we have not treated. Another approach to formally capturing SPKI's semantics is presented by Weeks (2001) as an example of a more general framework for trust management in a functional programming style. This work does not attempt to prove any correspondence with the tuple reduction rules.

Also related to our results in this paper is the work of Clarke et al. (Clarke, Elien, Ellison, Fredette, Marcos, and Rivest 2001), who consider the problem of discovering "certificate chains", i.e. proofs that a given set of certificates entails a given *concrete* certificate. Our notion of derivation "$\longrightarrow_0^*$" is similar to their notion of "name-reduction closure", and Theorem 5.2 is closely related to a result they state (Theorem 1) concerning the completeness of the name reduction closure. They do not explicitly take time and revocation into account, as we have done, and they also do not consider the more general sorts of consequences we have discussed. On the other hand, they do characterize the computational complexity of the certificate chain discovery problem for concrete certificates, whereas we have not discussed the complexity of the inference problems we treat. We are currently examining the complexity of certificate chain discovery in our more general setting.

7 Discussion

In this paper, we have given a semantics for SPKI certificates independent of the semantics given in terms of tuple reduction in the SPKI documentation (Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999a; Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen 1999b). This allowed us to examine the extent to which the SPKI tuple reduction rules are complete. The SPKI documents are ambiguous as to the purpose of the tuple reduction rules, and conflates their use for purposes of semantics, making concrete authorization decisions, and general reasoning (e.g., generating "certificate result certificates"). We have carefully separated the three concerns here, and have shown that the relations between them are somewhat subtle. In particular, we have shown that extra reduction rules are needed in order to do general reasoning about certificates. Our main technical results show that, in a precise sense, the reduction rules given in the SPKI document are complete with respect to concrete certificates; adding a few more rules gives us an "almost" complete system with respect to general reasoning about certificates. This "almost completeness" result seems to be the best we can do without having rules that take into account the cardinality of the set of keys.

We need to be careful about the interpretation of the conclusions for which we have shown the tuple reduction rules to be complete. The form of conclusion associated with the tuple reduction rules is $\models_{cl} (\wedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}') \Rightarrow \phi_{\mathtt{c}}$, where $\phi_{\mathtt{c}}$ has the form $\mathtt{now} \in \mathtt{V} \Rightarrow \phi'_{\mathtt{c}}$ (with the exact form of $\phi'_{\mathtt{c}}$ depending on the type of certifcate that $\mathtt{c}$ is). This states that $\phi'_{\mathtt{c}}$ holds for all times in $\mathtt{V}$ at which additionally, all the certificates in $C \cup C_R$ have been issued. Consider the certificate $\mathtt{c} = (\mathtt{cert\ k\ n\ k'}\ [0, 10])$. According to our semantics, if $\mathtt{c} \in r(5)$, (i.e., this certificate is issued at time 5), and no other certificates are issued in r, then $\mathtt{k}$'s $\mathtt{n} \longmapsto \mathtt{k}'$ holds in r during the interval $[5, 10]$, but not during the interval $[0, 4]$. On the other hand, the tuple reduction rules (trivially) allow the derivation of the tuple $\langle \mathtt{k}, \mathtt{n}, \mathtt{k}', [0, 10] \rangle$, which suggests that $\mathtt{k}$'s $\mathtt{n} \longmapsto \mathtt{k}'$ also holds during the interval $[0, 4]$.[14]

Whether this difference matters depends on the use that is made of derived tuples. For

[14]We have followed the SPKI definitions closely, but we remark that we could modify the definition of the tuple generated by a certificate and a CRL: if $\mathtt{c}$ has validity interval $[\mathtt{t}_1, \mathtt{t}_2]$, and the computation is done at $\mathtt{t}_0$, then we redefine the interval in $\tau(\mathtt{c}, C_R)$ to be interval $[max(\mathtt{t}_0, \mathtt{t}_1), \mathtt{t}_2]$. All our completeness results would go through with this change, and it would result in a better match between our semantics and the tuple reduction rules.

conclusions about the current time, the difference does not matter, since agents will only be making authorization decisions based on c after the time at which c is issued, i.e., after time 5. However, for conclusions requiring reasoning about the past, the difference may be an issue needing careful consideration. (Such reasoning occurs in proposals by Rivest (1998), Stubblebine (1995), and Stubblebine and Wright (1996) that involve authorizing an action when the latest time for which the existence of a right to perform that action can be proved is "sufficiently recent". Another example where reasoning about the past may be important is where an auditor is verifying that an authorization decision made in the past was justified by certificates issued at the time of the decision.) As it does not appear that reasoning about the past was a significant concern in the design of SPKI, we do not pursue this further here.

Another contribution of this paper is to show that nonmonotonic logic is not required in a logical modeling of revocation if one takes the SPKI perspective that revocation is not a change of mind but a revalidation. (This viewpoint is supported by the text in (SPKI Working Group 1998, Section 5.2): "The CRL is ... a completion of the certificate, rather than a change of mind.") The logic of this paper, like our earlier logic LLNC for SDSI, is monotonic. This does not prevent some aspects of its semantics from behaving nonmonotonically. In particular, $L(\mathtt{k}, \mathtt{n}, \mathtt{t})$ may decrease as $\mathtt{t}$ increases if, for example, a certificate is revoked at time $\mathtt{t}' > \mathtt{t}$ that was not revoked at time $\mathtt{t}$ or if the validity interval of a certificate passes. Similarly, the set of actions a principal is permitted to perform may decrease over time. Note also that the semantics does not require that if a certificate appears on a CRL then it will also appear on all later CRLs issued during the certificate's validity interval.[15] All of these types of "nonmonotonic" behaviour are entirely consistent with the monotonic logic we have developed.

We have focused on the SPKI reduction rules. However, we feel that the logic $\mathcal{L}_{SPKI}$ will be useful for more general reasoning about names and authorization in SPKI than just whether a particular principal is authorized to perform some actions. For example, we may want to know which principals are authorized to perform a certain action, which actions a principal is allowed to perform, which principals are bound to a particular name, and which names have a particular principal bound to them. In (Halpern and van der Meyden 2001), we showed how we could translate queries about names (like the last two) into Logic Programming queries, allowing us to take advantage of the well-developed Logic Programming technology for answering such queries. We believe that it should be relatively straightforward to extend the translation so that it can handle more general SPKI queries. It also seems useful to try to obtain a sound and complete axiomatization for the full logic $\mathcal{L}_{SPKI}$.

We believe that it should not be difficult to get such a complete axiomatization, using ideas from our earlier paper on SDSI, but we have not pursued this question. We have also not considered a number of features of SPKI, like threshold subjects and the precise

[15] Although it does not appear to have been noted by the authors of SPKI, this allows CRLs to be used to obtain temporary suspensions.

syntax of SPKI's tags (authorization expressions). It seems that our approach should be extendible to handle these features, but we have not checked any details.

A Appendix:Proofs

Proposition 4.3: *For every run r, there exists a unique (K, N, Act) interpretation minimal in the set of (K, N, Act) interpretations consistent with r.*

Proof: Let $\mathcal{I}_r$ consist of all interpretations consistent with r. $\mathcal{I}_r$ is nonempty, since the maximal interpretation is clearly consistent with r. Given an interpretation π, let L_π denote the L component of π and let P_π denote the P component of π. Define an interpretation π_0 by taking $L_{\pi_0}(\mathtt{k},\mathtt{n},\mathtt{t}) = \cap_{\pi\in\mathcal{I}_r} L_\pi(\mathtt{k},\mathtt{n},\mathtt{t})$ and taking $P_{\pi_0}(\mathtt{k}_1,\mathtt{t},\mathtt{k}_2,\mathtt{a}) = \min_{\pi\in\mathcal{I}_r} P_\pi(\mathtt{k}_1,\mathtt{t},\mathtt{k}_2,\mathtt{a})$. We now show that π_0 is the minimal element of $\mathcal{I}_r$.

Clearly $\pi_0 \leq \pi$ for all $\pi \in \mathcal{I}_r$. Thus, we must only show that $\pi_0 \in \mathcal{I}_r$. First we must show that for all naming certificates $\mathtt{c} = (\mathtt{cert\ k\ n\ p\ V}\ \langle \mathtt{k}_r\rangle)$ in $\cup_{\mathtt{t}'\leq\mathtt{t}}\ r(\mathtt{t}')$, if $\mathtt{t} \in V$ and $\mathtt{c}$ is applicable at $\mathtt{t}$ in r, then $[\![\mathtt{n}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}}$. By definition $[\![\mathtt{n}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}} = \cap_{\pi\in\mathcal{I}_r}[\![\mathtt{n}]\!]_{L_\pi,\mathtt{k},\mathtt{t}}$. Moreover, since each $\pi \in \mathcal{I}_r$ is consistent with r, we have that $[\![\mathtt{n}]\!]_{L_\pi,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L_\pi,\mathtt{k},\mathtt{t}}$ for $\pi \in \mathcal{I}_r$. It follows immediately that $[\![\mathtt{n}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}} = \cap_{\pi\in\mathcal{I}_r}[\![\mathtt{n}]\!]_{L_\pi,\mathtt{k},\mathtt{t}} \supseteq \cap_{\pi\in\mathcal{I}_r}[\![\mathtt{p}]\!]_{L_\pi,\mathtt{k},\mathtt{t}}$. Thus, it suffices to show that $\cap_{\pi\in\mathcal{I}_r}[\![\mathtt{p}]\!]_{L_\pi,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}}$. This follows by an easy induction on the structure of $\mathtt{p}$. For the base case, as we have already observed, we have equality; for the general case the argument is almost immediate from the definition.[16]

Next we must show that P_{π_0} is indeed a permission/delegation assignment, that is, that if $P_{\pi_0}(\mathtt{k}_1,\mathtt{t},\mathtt{k}_2,\mathtt{a}) = 2$ and $P_{\pi_0}(\mathtt{k}_2,\mathtt{t},\mathtt{k}_3,\mathtt{a}) = i$, then $P_{\pi_0}(\mathtt{k}_1,\mathtt{t},\mathtt{k}_3,\mathtt{a}) \geq i$. But if $P_{\pi_0}(\mathtt{k}_1,\mathtt{t},\mathtt{k}_2,\mathtt{a}) = 2$ and $P_{\pi_0}(\mathtt{k}_2,\mathtt{t},\mathtt{k}_3,\mathtt{a}) = i$, then $P_\pi(\mathtt{k}_1,\mathtt{t},\mathtt{k}_2,\mathtt{a}) = 2$ and $P_\pi(\mathtt{k}_2,\mathtt{t},\mathtt{k}_3,\mathtt{a}) \geq i$ for all $\pi \in \mathcal{I}_r$. Thus, $P_\pi(\mathtt{k}_1,\mathtt{t},\mathtt{k}_3,\mathtt{a}) \geq i$ for all $\pi \in \mathcal{I}_r$, so $P_{\pi_0}(\mathtt{k}_1,\mathtt{t},\mathtt{k}_3,\mathtt{a}) \geq i$.

Finally, we must show that P_{π_0} satisfies the second requirement of consistency. So suppose that $\mathtt{c} = (\mathtt{cert\ k\ p\ D\ A\ V}\ \langle\mathtt{k}_r\rangle)$ is an authorization certificate in $\cup_{\mathtt{t}'\leq\mathtt{t}}\ r(\mathtt{t}')$, $\mathtt{t} \in V$, and $\mathtt{c}$ is applicable at $\mathtt{t}$ in r. Then for all $\pi \in \mathcal{I}_r$, for all $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$, and all keys $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L_\pi,\mathtt{k},\mathtt{t}}$, we have

1. $P_\pi(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) \geq 1$,
2. if $\mathtt{D} = \mathtt{true}$ then $P_\pi(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) = 2$.

As we observed earlier, if $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L_{\pi_0},\mathtt{k},\mathtt{t}}$, then $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L_\pi,\mathtt{k},\mathtt{t}}$ for all $\pi \in \mathcal{I}_r$. Thus, $P_\pi(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) \geq 1$ for $\pi \in \mathcal{I}_r$, so $P_{\pi_0}(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) \geq 1$. Moreover, if $\mathtt{D} = \mathtt{true}$, then $P_\pi(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) = 2$ for $\pi \in \mathcal{I}_r$, so $P_{\pi_0}(\mathtt{k},\mathtt{t},\mathtt{k}',\mathtt{a}) = 2$. This completes the proof. ∎

[16] We remark that this is a significantly simpler proof than that given for the analogous minimality result for SDSI in our earlier paper (Halpern and van der Meyden 2001, Theorem 3.1).

We next prove the completeness results (Theorems 5.2 and 5.6). To do this, we need some preliminary definitions and results.

Rules R2 and R3 are very similar in the way that they transform principal expressions. For the completeness proofs, it is convenient to capture the commonalities by introducing a new type of tuple that we call a *3-tuple*. A 3-tuples has the form $\langle \mathtt{p}, \mathtt{q}, \mathtt{V}\rangle$ where $\mathtt{p}, \mathtt{q}$ are fully qualified principal expressions and $\mathtt{V}$ is an interval. Intuitively, this tuple says that p is bound to q during the time interval $\mathtt{V}$.

We have the following rules for reasoning about 3-tuples:

R5. $\longrightarrow \langle \mathtt{p}, \mathtt{p}, [0,\infty]\rangle$ for all fully qualified principal expressions $\mathtt{p}$.

R6. if $\langle \mathtt{p}, \mathtt{k}_1\text{'s}\,\mathtt{n}\text{'s}\,\mathtt{q}, \mathtt{V}_1\rangle + \langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}_2, \mathtt{V}_2\rangle \rightarrow \langle \mathtt{p}, \mathtt{k}_2\text{'s}\,\mathtt{q}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

R5, like R0, is essentially an axiom. R6 is somewhat in the spirit of R2, in that the third component of the 4-tuple is a key, rather than an arbitrary full qualified name. R6′ extends R6 in much the same way that R2′ extends R2.

R6′. if $\langle \mathtt{p}, \mathtt{k}_1\text{'s}\,\mathtt{n}\text{'s}\,\mathtt{q}, \mathtt{V}_1\rangle + \langle \mathtt{k}_1, \mathtt{n}, \mathtt{r}, \mathtt{V}_2\rangle \rightarrow \langle \mathtt{p}, \mathtt{r}\text{'s}\,\mathtt{q}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

We abuse notation somewhat and continue to write $T \longrightarrow_0^* \tau$ (resp., $T \longrightarrow_1^* \tau$) if τ can be derived from T using the rules R1–R3, R5, and R6 (resp., R0, R1, R2′, R3′, R5, R6′). The following two propositions collect the key properties of 3-tuples.

Proposition A.1: *Suppose that $i \in \{0,1\}$. If T is a set of 4- and 5-tuples such that $T \longrightarrow_i^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}_1\rangle$, then*

(a) if $T \longrightarrow_i^* \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_2\rangle$ then $T \longrightarrow_i^* \langle \mathtt{k}, \mathtt{n}, \mathtt{q}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$;

(b) if $T \longrightarrow_i^* \langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_2\rangle$ then $T \longrightarrow_i^* \langle \mathtt{k}, \mathtt{q}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$;

(c) if $T \longrightarrow_i^* \langle \mathtt{p}', \mathtt{p}, \mathtt{V}_2\rangle$ then $T \longrightarrow_i^* \langle \mathtt{p}', \mathtt{q}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$.

Proof: By a straightforward induction on the length of the derivation of $T \longrightarrow_i^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}_1\rangle$. ∎

Proposition A.2: *Suppose that $i \in \{0,1\}$. For all fully qualified names $\mathtt{p}, \mathtt{q}$ and local names $\mathtt{n}$, we have $T \longrightarrow_i^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}\rangle$ iff $T \longrightarrow_i^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{V}\rangle$.*

Proof: Again, by a straightforward induction on the length of the derivation of $T \longrightarrow_i^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}_1\rangle$. ∎

For convenience, we treat 3-tuples as certificates. For the 3-tuple $\mathtt{c} = \langle \mathtt{p}, \mathtt{q}, \mathtt{V}\rangle$, we take $\tau_{\mathtt{c}} = \mathtt{c}$ and define $\phi_{\mathtt{c}}$ to be the formula $\mathtt{now} \in \mathtt{V} \Rightarrow \mathtt{p} \longmapsto \mathtt{q}$. This formula captures the intuition that the 3-tuple is essentially saying that $\mathtt{p}$ is bound to $\mathtt{q}$.

The following theorem, whose proof is just like that of Theorem 5.3, says that $\longrightarrow_1^*$ (and hence $\longrightarrow_0^*$) continues to be sound in the presence of the rules for 3-tuples.

Theorem A.3: *Suppose that C is a finite subset of $\mathcal{C}$, C_R is a finite set of CRLs, and* $\mathtt{c}$ *is either a certificate in* $\mathcal{C}$ *or a 3-tuple. If* $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \tau_{\mathtt{c}}$*, then* $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$*. In fact, if π is an interpretation consistent with a run r and* $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \tau_{\mathtt{c}}$*, then* $r, \pi \models \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$.

With this background, we are ready to prove Theorem 5.2.

Theorem 5.2: *If* $\mathtt{c}$ *is a concrete certificate, C is a finite subset of $\mathcal{C}$, C_R is a finite set of CRLs, and* $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$*, then there exists a certificate* $\mathtt{c}'$ *that subsumes* $\mathtt{c}$ *such that* $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \tau_{\mathtt{c}'}$.

Proof: We prove the contrapositive. Suppose that there does not exist a certificate $\mathtt{c}'$ that subsumes $\mathtt{c}$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \tau_{\mathtt{c}'}$. We show that it is not the case that $\models_c \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$, by showing that there is a run r such that $r, \mathtt{k}, \mathtt{t} \models_c \bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'$ for all times $\mathtt{t}$ but $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$, where $\mathtt{t}_0$ is the time in the concrete certificate $\mathtt{c}$.

Construct r as follows: define $r(0) = C \cup C_R$ and $r(n) = \emptyset$ for all $n > 0$. Clearly $r, \mathtt{k}, \mathtt{t} \models_c \bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'$ for all times $\mathtt{t}$. To show that $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$, we need to identify the minimal interpretation consistent with r. Consider the interpretation $\pi = \langle L, P \rangle$ defined as follows:

1. $L(\mathtt{k}, \mathtt{n}, \mathtt{t})$ is the set of keys $\mathtt{k}'$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{k}', \mathtt{V} \rangle$ where $\mathtt{t} \in \mathtt{V}$.
2. $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) = 2$ if $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{k}', \mathtt{A}, \mathit{true}, \mathtt{V} \rangle$ for some action expression $\mathtt{A}$ with $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ and $\mathtt{t} \in \mathtt{V}$; $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) = 1$ if $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{k}', \mathtt{A}, \mathit{false}, \mathtt{V} \rangle$ for some $\mathtt{A}$ with $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ and $\mathtt{t} \in \mathtt{V}$ and it is not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{k}', \mathtt{A}', \mathit{true}, \mathtt{V}' \rangle$ for some $\mathtt{A}'$ with $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}')$ and $\mathtt{t} \in \mathtt{V}'$; and $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) = 0$ otherwise.

We must check that P is a legimitate permission/delegation assignment. In particular, suppose that $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_2, \mathtt{a}) = 2$ and $P(\mathtt{k}_2, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) = i \geq 1$. We must show that $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) = i$. Since $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_2, \mathtt{a}) = 2$, $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_1, \mathtt{k}_2, \mathtt{A}_1, \mathit{true}, \mathtt{V}_1 \rangle$ for some $\mathtt{A}_1$ with $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}_1)$ and some $\mathtt{V}_1$ containing $\mathtt{t}$. Similarly, we must have that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_2, \mathtt{k}_3, \mathtt{D}_2, \mathtt{A}_2, \mathtt{V}_2 \rangle$ where $\mathtt{t} \in \mathtt{V}_2$, for some $\mathtt{A}_2$ with $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}_2)$, some $\mathtt{D}_2$ such that if $i = 2$ then $\mathtt{D} = \mathit{true}$, and some $\mathtt{V}_2$ containing $\mathtt{t}$. Using R1, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_1, \mathtt{k}_3, \mathtt{D}_2, \mathtt{A}_1 \cap \mathtt{A}_2, \mathtt{V}_1 \cap \mathtt{V}_2 \rangle$. Since $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}_1 \cap \mathtt{A}_2)$ and $\mathtt{t} \in \mathtt{V}_1 \cap \mathtt{V}_2$, it follows that $P(\mathtt{k}_1, \mathtt{t}, \mathtt{k}_3, \mathtt{a}) \geq i$, as desired.

Next we must show that this interpretation is the minimal one consistent with r. We first establish that it is consistent. Thus, we must show that the formulas associated with naming and authorization certificates are satisfied in r. To do this, we use the following lemma. (This lemma was our main motivation for introducing 3-tuples.)

Lemma A.4: *If* $\mathtt{p}$ *is a fully qualified principal expression, then* $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$ *iff* $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', V \rangle$ *for some interval* $\mathtt{V}$ *containing* $\mathtt{t}$.

Proof: We proceed by induction on the structure of $\mathtt{p}$. If $\mathtt{p}$ is a key $\mathtt{k}'$ then, by definition, $[\![\mathtt{k}']\!]_{L,\mathtt{k},\mathtt{t}} = \{\mathtt{k}'\}$. By R5, $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}', \mathtt{k}', [0,\infty]\rangle$. For the converse, note that a straightforward induction on the length of the derivation shows that if $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}', \mathtt{q}, \mathtt{V}\rangle$, then $\mathtt{q} = \mathtt{k}'$ and $\mathtt{V} = [0,\infty]$.

For the inductive step, suppose that $\mathtt{p} = \mathtt{q}\text{'s}\,\mathtt{n}$. We first suppose that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{\mathtt{k},L,\mathtt{t}}$, and show that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', \mathtt{V}\rangle$ for some interval $\mathtt{V}$ containing $\mathtt{t}$. By definition of $[\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$, there exists $\mathtt{k}_1 \in [\![\mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}}$ such that $\mathtt{k}' \in L(\mathtt{k}_1, \mathtt{n}, \mathtt{t})$. By the inductive hypothesis, we have $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{q}, \mathtt{k}_1, \mathtt{V}_1\rangle$ for some interval $\mathtt{V}_1$ containing $\mathtt{t}$. By Proposition A.2, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{V}_1\rangle$. By definition of L and the fact that $\mathtt{k}' \in L(\mathtt{k}_1, \mathtt{n}, \mathtt{t})$, we have $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}', \mathtt{V}_2\rangle$ for some $\mathtt{V}_2$ containing $\mathtt{t}$. By R6, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{k}', \mathtt{V}_1 \cap \mathtt{V}_2\rangle$. This is what we need, since $\mathtt{t} \in \mathtt{V}_1 \cap \mathtt{V}_2$.

For the converse, suppose that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', \mathtt{V}\rangle$ for some interval $\mathtt{V}$ containing $\mathtt{t}$. We must show that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. Since $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', \mathtt{V}\rangle$ and $\mathtt{p} = \mathtt{q}\text{'s}\,\mathtt{n}$, it follows that there must be some key $\mathtt{k}_1$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{V}_1\rangle$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_1, \mathtt{n}, \mathtt{k}', \mathtt{V}_2\rangle$, where $\mathtt{V} = \mathtt{V}_1 \cap \mathtt{V}_2$. By the definition of L, we have that $\mathtt{k}' \in L(\mathtt{k}_1, \mathtt{n}, \mathtt{t})$. Since $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{V}_{m-1}\rangle$, by Proposition A.2, we also have $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{q}, \mathtt{k}_1, \mathtt{V}_{m-1}\rangle$. By the induction hypothesis, it follows that $\mathtt{k}_1 \in [\![\mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}}$. It is now immediate that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. ∎

Continuing with the proof of the theorem, recall that we must show that the formulas associated with certificates issued in r are satisfied in r. Suppose that $\mathtt{c} = (\mathtt{cert\ k\ n\ p\ V}\ \langle \mathtt{k}_r\rangle)$ is a naming certificate in $\cup_{\mathtt{t}' \le \mathtt{t}}\, r(\mathtt{t}')$ that is applicable in r at time $\mathtt{t} \in \mathtt{V}$. We need to show that $[\![\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. For this, note that either $\mathtt{k}_r$ is not present in $\mathtt{c}$ and and $\tau(\mathtt{c}) = \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}\rangle$, or there exists a CRL $\mathtt{c}_R \in C_R$ with respect to which $\mathtt{c}$ is live and $\tau(\mathtt{c}, \mathtt{c}_R) = \langle \mathtt{k}, \mathtt{n}, \mathtt{q}, \mathtt{V}'\rangle$, where again $\mathtt{t} \in \mathtt{V}'$. In either case, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_1\rangle$ for some $\mathtt{V}_1$ containing $\mathtt{t}$.

We now want to show that if $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_1\rangle$, then $[\![\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}} \supseteq [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. Suppose that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. By Lemma A.4, we have that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', \mathtt{V}_2\rangle$ for some $\mathtt{V}_2$ containing $\mathtt{t}$. Since $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V}_1\rangle$, by Proposition A.1, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{k}', \mathtt{V}_1 \cap \mathtt{V}_2\rangle$. Using the definition of L, it follows that $\mathtt{k}' \in [\![\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}}$, as desired.

Next, suppose that $\mathtt{c} = (\mathtt{cert\ k\ p\ D\ A\ V}\ \langle \mathtt{k}_r\rangle)$ is an authorization certificate in $\cup_{\mathtt{t}' \le \mathtt{t}} r(\mathtt{t}')$ that is valid in r at time $\mathtt{t} \in \mathtt{V}$. As in the case of naming certificates, it follows that there is some interval $\mathtt{V}_1$ such that $\mathtt{t} \in \mathtt{V}_1$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1\rangle$. Suppose that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. By Lemma A.4, $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{p}, \mathtt{k}', \mathtt{V}_2\rangle$ for some $\mathtt{V}_2$ containing $\mathtt{t}$. By Proposition A.1, $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{k}', \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \cap \mathtt{V}_2\rangle$. From the definition of P, it follows that

$$r, \pi, \mathtt{k}, \mathtt{t} \models \mathit{Perm}(\mathtt{k}, \mathtt{p}, \mathtt{A}) \wedge (\mathtt{D} \Rightarrow \mathit{Del}(\mathtt{k}, \mathtt{p}, \mathtt{A})),$$

as required.

To see that $\langle L, P\rangle$ is in fact the minimal interpretation consistent with r, suppose that $\pi' = \langle L', P'\rangle$ is another interpretation consistent with r. We need to show that $\pi \leq \pi'$. We first show that $L \leq L'$. Suppose that $\mathtt{k}' \in L(\mathtt{k}, \mathtt{n}, \mathtt{t})$. By definition of L, $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{n}, \mathtt{k}', \mathtt{V}\rangle$ for some interval $\mathtt{V}$ containing $\mathtt{t}$. Since π' is consistent with r, by Theorem A.3,

$$r, \pi' \models \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow (\mathtt{now} \in \mathtt{V} \Rightarrow \mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}').$$

Moreover, we have that $r, \pi', \mathtt{k}, \mathtt{t} \models \bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'$, by defintion of r. Since $\mathtt{t} \in \mathtt{V}$, it follows that $r, \pi', \mathtt{k}, \mathtt{t} \models \mathtt{k}\text{'s}\,\mathtt{n} \longmapsto \mathtt{k}'$, so $\mathtt{k}' \in L'(\mathtt{k}, \mathtt{n}, \mathtt{t})$. Thus, $L \leq L'$. The argument that $P \leq P'$ is very similar, and is left to the reader.

It remains to show that $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$. To see this, suppose first that $\mathtt{c}$ is a naming certificate such that $\tau_{\mathtt{c}} = \langle \mathtt{k}, \mathtt{n}, \mathtt{k}', [\mathtt{t}_0, \mathtt{t}_0]\rangle$. Since $\mathtt{c}$ is not subsumed by any 4-tuple derivable from $\mathtt{Tuples}(C, C_R)$, it is immediate from the definition of L that $\mathtt{k}' \notin L(\mathtt{k}, \mathtt{n}, \mathtt{t}_0)$. From this it follows that $r, \pi, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$. The argument for authorization certificates is similar. ∎

We now prove Theorem 5.6.

Theorem 5.6: *If* $\mathtt{c}$ *is a certificate,* C *is a finite subset of* $\mathcal{C}$*,* C_R *is a finite set of CRLs, and* $|K| > |C| + |\mathtt{c}|$*, and* $\models_{c,K} \left(\bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'\right) \Rightarrow \phi_{\mathtt{c}}$*, then* $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$*; moreover, if* $\mathtt{c}$ *is a point-valued certificate, then* $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \tau_{\mathtt{c}'}$ *for some certificate* $\mathtt{c}'$ *subsuming* $\mathtt{c}$*.*

Proof: The proof proceeds much in the same spirit as the proof of Theorem 5.2, using 3-tuples. Suppose that it is not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$ (or it is not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \tau_{\mathtt{c}}$, in the case that $\mathtt{c}$ is a point-valued naming certificate). Again the idea is to construct a run r such that $r, \mathtt{k}, \mathtt{t} \models_c \bigwedge_{\mathtt{c}' \in C \cup C_R} \mathtt{c}'$ for all times $\mathtt{t}$ and that $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$ for some time $\mathtt{t}_0$. The construction of r is somewhat more complicated than in the case of Theorem 5.2.

Given a principal $\mathtt{p}$, let $Cl(\mathtt{p})$ be the smallest set S' of principal expressions containing $\mathtt{p}$, such that if $\mathtt{p}\text{'s}\,\mathtt{n} \in S'$ then $\mathtt{p} \in S'$. It is easy to see that $|Cl(\mathtt{p})| \leq |\mathtt{p}|$, where $|\mathtt{p}|$ is the length of $\mathtt{p}$ viewed as a string of symbols. If C' is a set of certificates, let $Cl(C')$ be the union of $Cl(\mathtt{p})$, for all the principal expressions $\mathtt{p}$ that appear in a certificate in C' as well as the principal expressions $\mathtt{k}\text{'s}\,\mathtt{n}$ for 4-tuples $\langle \mathtt{k}, \mathtt{n}, \mathtt{q}, V\rangle$ in C'. Again, it should be clear that $|Cl(C')| \leq |C'|$. We will be interested in the set $S = Cl(C \cup \{\mathtt{c}\})$ of pricipal expressions. Note that since we have assumed that $|K| > |C| + |c|$, it follows that $|K| > |S|$.

Let T be the set of time points containing 0, ∞, and the left and right components of each interval in $C \cup C_R$. Note first that only a finite number of intervals $\mathtt{V}$ can appear in a tuple $\langle \mathtt{p}_0, \mathtt{q}_0, \mathtt{V}\rangle$ generated from $\mathtt{Tuples}(C, C_R)$, since every interval in a tuple generated has both left and right components in T. Hence, T is finite. Let $\mathcal{V}$ be the set of all point intervals $[\mathtt{t},\mathtt{t}]$ where $\mathtt{t} \in T - \{\infty\}$, together with all intervals $[\mathtt{t}_1+1, \mathtt{t}_2-1]$ such that $\mathtt{t}_1, \mathtt{t}_2 \in T$, $\mathtt{t}_1 < \mathtt{t}_2$ and for no $\mathtt{t} \in T$ do we have $\mathtt{t}_1 < \mathtt{t} < \mathtt{t}_2$. (We take $\infty - 1 = \infty$). Note that the intervals in $\mathcal{V}$ are pairwise disjoint and span $I\!N$. Moreover, if $\mathtt{V}$ is the validity interval of a tuple derivable from $\mathtt{Tuples}(C, C_R)$ and $\mathtt{W} \in \mathcal{V}$, then either $\mathtt{V} \cap \mathtt{W} = \emptyset$ or $\mathtt{V} \supseteq \mathtt{W}$. Finally, note that $\mathcal{V}$ is finite, since T is.

For each interval $\mathtt{W} \in \mathcal{V}$, define the equivalence relation $\approx_{\mathtt{W}}$ on S by $\mathtt{p}_1 \approx_{\mathtt{W}} \mathtt{p}_2$ if both $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}_1, \mathtt{p}_2, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}_2, \mathtt{p}_1, \mathtt{V}'' \rangle$ for some $\mathtt{V}'' \supseteq \mathtt{W}$. We write $[\mathtt{p}]_{\mathtt{W}}$ for the equivalence class of $\approx_{\mathtt{W}}$ containing $\mathtt{p}$.

Let X be a set of keys in K of cardinality $|S|$. For each interval $\mathtt{W} \in \mathcal{V}$ and equivalence class x of $\approx_{\mathtt{W}}$, choose a key $\mathtt{k}_{x,\mathtt{W}}$ in X in such a way that if $x \neq y$, then $\mathtt{k}_{x,\mathtt{W}} \neq \mathtt{k}_{y,\mathtt{W}}$. Without loss of generality, we may assume that $\mathtt{k}_{[\mathtt{k}]_{\mathtt{W}},\mathtt{W}} = \mathtt{k}$ for each key $\mathtt{k} \in S$. (Note that $[\mathtt{k}]_{\mathtt{W}} = \{\mathtt{k}\}$ since, as we observed earlier, if $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{q}, \mathtt{V} \rangle$, then we must have $\mathtt{p} = \mathtt{k}$.) For ease of exposition, we write $\mathtt{k}_{\mathtt{p},\mathtt{W}}$ rather than $\mathtt{k}_{[\mathtt{p}]_{\mathtt{W}},\mathtt{W}}$.

Consider the run r where no certificates are issued after time 0 and $r(0)$ consists of the following certificates:

1. all certificates in $C \cup C_R$,
2. for each $\mathtt{W} \in \mathcal{V}$, principal expression of the form $\mathtt{p}$'s $\mathtt{n}$ in S, principal expression $\mathtt{q} \in S$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}$, the naming certificate ($\mathtt{cert}\ \mathtt{k}_{\mathtt{p},\mathtt{W}}\ \mathtt{n}\ \mathtt{k}_{\mathtt{q},\mathtt{W}}\ \mathtt{W}$),
3. for each $\mathtt{W} \in \mathcal{V}$, principal expression $\mathtt{q} \in S$, and key $\mathtt{k} \in S$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{q}, \mathtt{D}, \mathtt{A}, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}$, the authorization certificate ($\mathtt{cert}\ \mathtt{k}\ \mathtt{k}_{\mathtt{q},\mathtt{W}}\ \mathtt{D}\ \mathtt{A}\ \mathtt{W}$).

Note that this is a finite set of certificates since $\mathcal{W}$ and S are a finite sets.

We now construct an interpretation $\pi = \langle L, P \rangle$ consistent with r. For $\mathtt{t} \in I\!N$, let $\mathtt{W}(\mathtt{t})$ be the unique interval in $\mathcal{V}$ containing $\mathtt{t}$. Define $L(\mathtt{k}, \mathtt{n}, \mathtt{t}) = \emptyset$ unless $\mathtt{k}$ is $\mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t})}$ for some principal expression $\mathtt{p}$; $L(\mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t})}, \mathtt{n}, \mathtt{t}) = \{\mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})} : \mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle, \mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})\}$. $L(\mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t})}, \mathtt{n}, \mathtt{t})$ is well defined since, if $\mathtt{p}' \approx_{\mathtt{W}(\mathtt{t})} \mathtt{p}$, then by Proposition A.2, $(\mathtt{p}')\text{'s}\,\mathtt{n} \approx_{\mathtt{W}(\mathtt{t})} \mathtt{p}\text{'s}\,\mathtt{n}$, hence $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle$ iff $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle (\mathtt{p}')\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle$. Similarly, $P(\mathtt{k}, \mathtt{t}, \mathtt{k}', \mathtt{a}) = 0$ unless $\mathtt{k}'$ is $\mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}$, for some principal expression $\mathtt{q}$. If $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{q}, \mathit{true}, \mathtt{A}, \mathtt{V} \rangle$, where $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ and $\mathtt{V} \supseteq \mathtt{W}(\mathtt{t})$, then $P(\mathtt{k}, \mathtt{t}, \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}, \mathtt{a}) = 2$; if $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{q}, \mathit{false}, \mathtt{A}, \mathtt{V} \rangle$, where $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ and $\mathtt{V} \supseteq \mathtt{W}(\mathtt{t})$ and it is not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}, \mathtt{q}, \mathit{true}, \mathtt{A}', \mathtt{V}' \rangle$ for some $\mathtt{A}'$ such that $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}')$ and $\mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})$, then $P(\mathtt{k}, \mathtt{t}, \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}, \mathtt{a}) = 1$; otherwise, $P(\mathtt{k}, \mathtt{t}, \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}, \mathtt{a}) = 0$.

We want to show that π is the minimal interpretation consistent with r. We first need an analogue of Lemma A.4.

Lemma A.5: *For all* $\mathtt{p} \in S$, *keys* $\mathtt{k}$, *and times* $\mathtt{t}$, $[\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}} = \{\mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})} : \mathtt{q} \in S, \mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}' \rangle, \mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})\}$.

Proof: We proceed by induction on the construction of $\mathtt{p}$. If $\mathtt{p}$ is a key or an expression of the form $\mathtt{k}$'s $\mathtt{n}$, the claim follows trivially from the definitions. Note we cannot have $\mathtt{p} = \mathtt{n}$ for a local name $\mathtt{n}$, by the construction of S. So suppose that $\mathtt{p}$ has the form $\mathtt{q}$'s $\mathtt{n}$. We first show that if $\mathtt{k}' \in [\![\mathtt{q}\text{'s}\,\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}}$, then there exist $\mathtt{r} \in S$ and $\mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})$ such that $\mathtt{k}' = \mathtt{k}_{\mathtt{r},\mathtt{W}(\mathtt{t})}$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{r}, \mathtt{V}' \rangle$. By definition of $[\![\mathtt{q}\text{'s}\,\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}}$, we have that $\mathtt{k}' \in L(\mathtt{k}'', \mathtt{n}, \mathtt{t})$ for some key $\mathtt{k}'' \in [\![\mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}}$. By the induction hypothesis, there exists $\mathtt{q}_1 \in S$ such that $\mathtt{k}'' = \mathtt{k}_{\mathtt{q}_1,\mathtt{W}(\mathtt{t})}$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}, \mathtt{q}_1, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})$. It follows from Proposition A.2 that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{q}_1\text{'s}\,\mathtt{n}, \mathtt{V}' \rangle$. By definition of L, since $\mathtt{k}' \in L(\mathtt{k}_{\mathtt{q}_1,\mathtt{W}(\mathtt{t})}, \mathtt{n}, \mathtt{t})$, there exists a principal expression $\mathtt{q}_2$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}_1\text{'s}\,\mathtt{n}, \mathtt{q}_2, \mathtt{V}'' \rangle$, where $\mathtt{V}'' \supseteq \mathtt{W}(\mathtt{t})$ and $\mathtt{k}' = \mathtt{k}_{\mathtt{q}_2,\mathtt{W}(\mathtt{t})}$. By Proposition A.1, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{q}_2, \mathtt{V}' \cap \mathtt{V}'' \rangle$. Since $\mathtt{V}' \cap \mathtt{V}'' \supseteq \mathtt{W}(\mathtt{t})$, we are done.

For the converse, suppose that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}\text{'s}\,\mathtt{n}, \mathtt{r}, \mathtt{V}' \rangle$, where $\mathtt{r} \in S$ and $\mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})$. Since $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{q}, \mathtt{q}, [0,\infty] \rangle$, we have that $\mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})} \in [\![\mathtt{q}]\!]_{L,\mathtt{k},\mathtt{t}}$, by the induction hypothesis. Moreover, by definition of L, we have $\mathtt{k}_{\mathtt{r},\mathtt{W}(\mathtt{t})} \in L(\mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}, \mathtt{n})$. It follows that $\mathtt{k}_{\mathtt{r},\mathtt{W}(\mathtt{t})} \in [\![\mathtt{q}\text{'s}\,\mathtt{n}]\!]_{L,\mathtt{k},\mathtt{t}}$. ∎

We use Lemma A.5 to show that π is consistent with r. Consider a naming certificate $\mathtt{c} = (\mathtt{cert\ k_1\ n\ p\ V_1}\ \langle \mathtt{k_r} \rangle) \in C$ that is applicable at time $\mathtt{t} \in \mathtt{V}_1$. Either $\mathtt{c}$ is irrevoccable, or there exists a CRL $\mathtt{c}'$ in $r(0)$ with validity interval $\mathtt{V}_2$ such that $\mathtt{t} \in \mathtt{V}_2$ and $\mathtt{c}$ is live with respect to $\mathtt{c}'$. As in the proof of Theorem 5.2, in either case, there is an interval $\mathtt{V}' \subseteq \mathtt{V}_1$ containing $\mathtt{t}$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_0^* \langle \mathtt{k}_1, \mathtt{n}, \mathtt{p}, \mathtt{V}' \rangle$. Since $\mathtt{t} \in \mathtt{V}_1$, we must have $\mathtt{V}' \supseteq \mathtt{W}(\mathtt{t})$. By R5, $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{k}_1\text{'s}\,\mathtt{n}, [0,\infty] \rangle$ so, using R6′, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{p}, \mathtt{V}' \rangle$. Now if $\mathtt{k}_2 \in [\![\mathtt{p}]\!]_{L,\mathtt{k}_1,\mathtt{t}}$, then by Lemma A.5, $\mathtt{k}_2 = \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}$ for some $\mathtt{q} \in S$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}'' \rangle$, where $\mathtt{V}'' \supseteq \mathtt{W}(\mathtt{t})$. It follows from Proposition A.1(c) that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}_1\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \cap \mathtt{V}'' \rangle$. Since $\mathtt{V}' \cap \mathtt{V}'' \supseteq \mathtt{W}(\mathtt{t})$, it follows from the defintion of L that $\mathtt{k}_2 = \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})} \in [\![\mathtt{n}]\!]_{L,\mathtt{k}_1,\mathtt{t}}$, as required.

Next, consider certificates of the form $(\mathtt{cert}\ \mathtt{k}_{\mathtt{p},\mathtt{W}}\ \mathtt{n}\ \mathtt{k}_{\mathtt{q},\mathtt{W}}\ \mathtt{W})$, where $\mathtt{W} \in \mathcal{V}$, $\mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q} \in S$, and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}$. Since such certificates are irrevocable, they are always applicable. If $\mathtt{t} \in W$, we have $[\![\mathtt{n}]\!]_{L,\mathtt{k}_{\mathtt{p},\mathtt{W}},\mathtt{t}} = L(\mathtt{k}_{\mathtt{p},\mathtt{W}}, \mathtt{n}, \mathtt{t}) \supseteq \{\mathtt{k}_{\mathtt{q},\mathtt{W}}\} = [\![\mathtt{k}_{\mathtt{q},\mathtt{W}}]\!]_{L,\mathtt{k}_{\mathtt{p},\mathtt{W}},\mathtt{t}}$ by construction.

Next, suppose that $\mathtt{c} = (\mathtt{cert\ k\ p\ D\ A\ V}\ \langle \mathtt{k}_r \rangle)$ is an authorization certificate in C that is applicable in r at time $\mathtt{t} \in \mathtt{V}$. As in the case of naming certificates, it follow that there is some interval $\mathtt{V}_1$ such that $\mathtt{t} \in \mathtt{V}_1$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{p}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \rangle$. Suppose that $\mathtt{k}' \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}}$. By Lemma A.5, $\mathtt{k}' = \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}$ for some $\mathtt{q} \in S$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}, \mathtt{q}, \mathtt{V}_2 \rangle$ for some $\mathtt{V}_2 \supseteq \mathtt{W}(\mathtt{t})$. By Proposition A.1, it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{q}, \mathtt{D}, \mathtt{A}, \mathtt{V}_1 \cap \mathtt{V}_2 \rangle$. From the definition of P and the fact that $\mathtt{k}' = \mathtt{k}_{\mathtt{q},\mathtt{W}(\mathtt{t})}$, it follows that

$$r, \pi, \mathtt{k}, \mathtt{t} \models \mathit{Perm}(\mathtt{k}, \mathtt{k}', \mathtt{A}) \wedge (\mathtt{D} \Rightarrow \mathit{Del}(\mathtt{k}, \mathtt{k}', \mathtt{A})),$$

as required.

Finally, consider an authorization certificate in $r(0)$ of the form $(\mathtt{cert}\ \mathtt{k}\ \mathtt{k}_{\mathtt{q},\mathtt{W}}\ \mathtt{D\ A\ W})$. By definition, $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{q}, \mathtt{D}, \mathtt{A}, \mathtt{V}' \rangle$ for some $\mathtt{V}' \supseteq \mathtt{W}$. Again, it is immediate from the definition of P that

$$r, \pi, \mathtt{k}, \mathtt{t} \models Perm(\mathtt{k}, \mathtt{k}_{\mathtt{q},\mathtt{W}}, \mathtt{A}) \wedge (\mathtt{D} \Rightarrow Del(\mathtt{k}, \mathtt{k}_{\mathtt{q},\mathtt{W}}, \mathtt{A})).$$

This completes the proof that π is consistent with r.

To show that π is the minimal interpretation consistent with r, suppose that $\pi' = (L', P')$. Suppose that $\mathtt{k}' \in L(\mathtt{k}, \mathtt{n}, \mathtt{t})$ and let $\mathtt{W} = \mathtt{W}(\mathtt{t})$. Then there exist principal expressions $\mathtt{p}$ and $\mathtt{q}$ such that $\mathtt{k} = \mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t})}$, $\mathtt{k}' = \mathtt{k}_{\mathtt{q},\mathtt{W}}$ and $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{p}\text{'s}\,\mathtt{n}, \mathtt{q}, \mathtt{V}' \rangle$ for some interval $\mathtt{V}' \supseteq \mathtt{W}$. Thus, $r(0)$ contains the certificate ($\mathtt{cert}\ \mathtt{k}_{\mathtt{p},\mathtt{W}}\ \mathtt{n}\ \mathtt{k}_{\mathtt{q},\mathtt{W}}\ \mathtt{W}$). This implies that if $\pi' = (L', P')$ is another interpretation consistent with r then we must also have $\mathtt{k}' \in L'(\mathtt{k}, \mathtt{n}, \mathtt{t})$. A similar argument works in the case of P. Thus, π is the minimal interpretation consistent with r.

It remains to show that $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$ for some choice of $\mathtt{t}_0$. If $\mathtt{c}$ is a point-valued certificate such that $\tau_{\mathtt{c}} = \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, [\mathtt{t}_0, \mathtt{t}_0] \rangle$, then since it is not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$, it is also not the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}\text{'s}\,\mathtt{n}, \mathtt{p}, V \rangle$ for any interval V containing $\mathtt{t}_0$. (Otherwise, from Proposition A.1, we would be able to use R0 and R4(a) to derive $\tau_{\mathtt{c}}$.) Hence, by Lemma A.5, it follows that $\mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t}_0)} \notin L(\mathtt{k}, \mathtt{n}, \mathtt{t}_0)$. Of course, we do have (using R0) that $\mathtt{k}_{\mathtt{p},\mathtt{W}(\mathtt{t}_0)} \in [\![\mathtt{p}]\!]_{L,\mathtt{k},\mathtt{t}_0}$. Thus, $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$.

Next, suppose that $\mathtt{c}$ is an arbitrary naming certificate, with $\tau_{\mathtt{c}} = \langle \mathtt{k}, \mathtt{n}, \mathtt{p}, \mathtt{V} \rangle$. Let $\mathcal{V}'$ consist of all intervals $\mathtt{V}' \in \mathcal{V}$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}\text{'s}\,\mathtt{n}, \mathtt{p}, \mathtt{V}' \rangle$. $\mathcal{V}'$ is finite, since $\mathcal{V}$ is. Moreover, it cannot be the case that $\mathtt{V} \subseteq \cup\mathcal{V}'$, for otherwise, using R4(a) and R0, it would follow that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \tau_{\mathtt{c}}$. Choose $\mathtt{t}_0 \in \mathtt{V} - \cup\mathcal{V}'$. It follows just as above that $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$.

Finally, suppose that $\mathtt{c}$ is an authorization certificate, with $\tau_{\mathtt{c}} = \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}, \mathtt{V} \rangle$. (The argument is similar if *true* is replaced by *false*.) We claim that there must be some time $\mathtt{t}_0$ and action $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$ such that for no interval $\mathtt{V}'$ containing $\mathtt{t}_0$ and $\mathtt{A}'$ such that $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}')$ is it the case that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}', \mathtt{V}' \rangle$. It follows from the claim that $P(\mathtt{k}, \mathtt{t}_0, \mathtt{k}_{\mathtt{p},\mathtt{W}_{\mathtt{t}_0}}, \mathtt{a}) \neq 2$, so $r, \mathtt{k}, \mathtt{t}_0 \not\models_c \phi_{\mathtt{c}}$. This completes the proof of the theorem. For the proof of the claim, suppose, by way of contradiction, that for all $\mathtt{t} \in \mathtt{V}$ and all actions $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$, there exists an interval $\mathtt{V}_{\mathtt{t},\mathtt{a}}$ containing $\mathtt{t}$ and an action expression $\mathtt{A}_{\mathtt{t},\mathtt{a}}$ such that $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A}_{\mathtt{t},\mathtt{a}})$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}_{\mathtt{t},\mathtt{a}}, \mathtt{V}_{\mathtt{t},\mathtt{a}} \rangle$. Note that there are only finitely many time intervals $\mathtt{V}'$ and action expressions $\mathtt{A}'$ such that $\mathtt{Tuples}(C, C_R) \longrightarrow_1^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}', \mathtt{V}' \rangle$. For each $\mathtt{a} \in \alpha_{\mathcal{A}}(\mathtt{A})$, let $\mathtt{A}_{\mathtt{a}}$ be the intersection of all the action expressions $\mathtt{A}_{\mathtt{a},\mathtt{t}}$ for $\mathtt{t} \in \mathtt{V}$. Since this is a finite intersection, $\mathtt{A}_{\mathtt{a}} \in \mathcal{A}$. Moreover, even if $\alpha_{\mathcal{A}}(\mathtt{A})$ is infinite, the number of distinct sets $\mathtt{A}_{\mathtt{a}}$ is finite. By R4(c), for each $\mathtt{t} \in \mathtt{V}$, we have that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}_{\mathtt{a}}, \mathtt{V}_{\mathtt{t},\mathtt{a}} \rangle$. Since the union of the sets $\mathtt{V}_{\mathtt{t},\mathtt{a}}$ contains $\mathtt{V}$, and there are only finitely many such sets, by R4(b) it follows that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}_{\mathtt{a}}, \mathtt{V} \rangle$. Finally, since $\alpha_{\mathcal{A}}(\mathtt{A})$ is contained in the union of the sets $\alpha_{\mathcal{A}}(\mathtt{A}_{\mathtt{a}})$, it follows from R4(c) that $\mathtt{Tuples}(C, C_R) \longrightarrow_2^* \langle \mathtt{k}, \mathtt{p}, \mathit{true}, \mathtt{A}, \mathtt{V} \rangle$, which is a contradiction. ∎

Acknowledgments: We thank Jon Howell and the reviewers of this paper for their useful comments.

References

Abadi, M. (1998). On SDSI's linked local name spaces. *Journal of Computer Security 6*(1-2), 3–21.

Aura, T. (1998). On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, pp. 14–26.

Cadoli, M. and M. Lenzerini (1994). The complexity of closed world reasoning and circumscription. *Journal of Information and System Sciences 48*, 255–301.

Clarke, D., J.-E. Elien, C. Ellison, M. Fredette, A. Marcos, and R. L. Rivest (2001). Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security 9*(4), 285–322.

Ellison, C., B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen (1999b). Simple public key certificate. At `http://world.std.com/~cme/spki.txt`. Internet RFC 2693.

Ellison, C., B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen (1999a). SPKI certificate theory. At `http://www.ietf.org/html.charters/spki-charter.html`. Internet RFC 2693.

Halpern, J. Y. and R. van der Meyden (2001). A logic for SDSI's linked local name spaces. *Journal of Computer Security 9*(1,2), 47–74.

Howell, J. and D. Kotz (2000). A formal semantics for SPKI. Technical Report TR 2000-363, Department of Computer Science, Dartmouth College, Hanover, NH. (an extended version of a paper in *Proc. 6th European Symposium on Research in Computer Security*).

Jajodia, S., P. Samarati, and V. Subramanian (1997). A logical language for expressing authorizations. In *Proceedings 1997 IEEE Symposium on Security and Privacy*, pp. 31–42.

Li, N. (2000). Local names in SPKI/SDSI. In *Proc. 13th IEEE Computer Security Foundations Workshop*, pp. 2–15. IEEE Press.

Li, N. and J. Feigenbaum (2001). Nonmonotonicity, user interfaces, and risk assessment in certificate revocation (Position paper). In *Proceedings of Financial Cryptography 2001*.

Li, N., B. N. Grosof, and J. Feigenbaum (1999). A logic-based knowledge representation for authorization with delegation. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pp. 162–174.

Li, N., B. N. Grosof, and J. Feigenbaum (2000). A nonmonotonic delegation logic with prioritized conflict handling. Unpublished manuscript.

Rivest, R. (1998). Can we eliminate certificate revocation lists? In *Proceedings of Financial Cryptography 1998*. Also available at `http://theory.lcs.mit.edu/~rivest/publications.html`.

Rivest, R. and B. Lampson (1996). SDSI — a simple distributed security infrastructure. Available at `http://theory.lcs.mit.edu/~cis/sdsi.html`.

SPKI Working Group (1998). Simple public key infrastructure, internet draft. `http://www.ietf.org/html.charters/spki-charter.html`.

Stubblebine, S. (1995). Recent-secure authentication: enforcing revocation in distributed systems. In *Proc. 19th IEEE Symposium on Security and Privacy*, pp. 224–235.

Stubblebine, S. and R. Wright (1996). An authentication logic supporting synchronization, revocation, and recency. In *Proc. Third ACM Conference on Computer and Communications Security*, pp. 95–105.

Weeks, S. (2001). Understanding trust management systems. In *IEEE Symposium on Security and Privacy*, pp. 94–105.

Woo, T. and S. Lam (1993). Authorization in distributed systems: a new approach. *Journal of Computer Security 2*, 107–136.