Avoiding Delegation Subterfuge using Linked Local Permission Names

Simon N. Foley and Samane Abdi

Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork, Ireland
s.foley@cs.ucc.ie; s.abdi@4c.ucc.ie

Abstract. Trust Management systems are typically explicit in their assumption that principals are uniquely identifiable. However, the literature has not been as prescriptive concerning the uniqueness of the permissions delegated by principals. Delegation subterfuge may arise when there is ambiguity concerning the uniqueness and interpretation of a permission. As a consequence, delegation chains that are used by principals to prove authorization may not actually reflect the original intention of all of the participants in the chain. This paper describes an extension to SPKI/SDSI that uses the notion of linked local permissions to eliminate ambiguity concerning the interpretation of a permission and thereby avoid subterfuge attacks.

1 Introduction

Trust Management systems such as [3,4,6,12] are intended to provide a decentralized approach to constructing and interpreting trust/authorization relationships between principals. Unlike a centralized authorization server-based approach, authorization rules are defined and signed locally by issuing principals and these cryptographic credentials can be distributed in any manner to suit the design of the (Trust Management-based) access control mechanism. While credential-based policy rules are inherently decentralized, many implicitly assume unique and unambiguous global permissions [10], effectively originating from some *central* authority that provides a permission namespace that everyone agrees to consistently use. For example, relying on IANA as a central source of unique identifiers for Internet resources [3], or relying on CCITT's X500 names [5] as a means of identifying principals.

However, principals may prefer not to have to trust some global authority, whatever about the practicalities of such an authority providing permission names for *everything.* Under reasonable assumptions, public keys can be considered to be globally unique and, by signing a permission, a principal can be sure that the resulting value is globally unique. The position that underlies the design of SPKI/SDSI [6] is that referencing public-key values as principal identifiers is difficult and error prone and proposes the use of *local names* as a means of providing a more effective way to identify principals. This paper starts from a similar position. That is, that the use of public key values as permission identifiers is equally problematic and proposes the use of *local permission names*, whereby principals can use local names to reference globally unique permissions in an unambiguous manner.

The paper is organized as follows. Section 2 provides a simple logic model for reasoning over SPKI/SDSI credentials. SPKI/SDSI uses s-expressions to define permissions and, in the absence of global agreement regarding their interpretation, Section 3 describes how it can lead to *subterfuge* [10, 15] whereby it is possible for an attacker to interfere with the intent of a delegation chain. Section 4 extends the SPKI/SDSI logic by incorporating local permission names and argues how the use of these permissions can avoid subterfuge. The logic proposed in Section 4 supports truly decentralized Trust Management whereby a principal may define, without reference to any central authority, its own local permission namespace, define a permission ordering over that namespace and also orderings relative to permissions in other namespaces. Section 5 describes how this logic can be used to support subterfuge-safe Trust Management.

2 Trust Management

2.1 SDSI Names

SDSI [6] uses *local names* to refer to unique principals whereby local name $(K\ N)$ identifies a principal named as N in the namespace of the principal that owns the public key K. A name certificate $\{\!|N, P, V|\!\}_{sK}$ is a statement signed by the owner of public key K that principal P is a definition for the name N in K's local namespace, during validity period V. For example, certificate $\{\!|\textsf{Bob}, K_B, V|\!\}_{sK_A}$ specifies that Bob is the name that principal K_A uses to refer to (the owner of) K_B. Local names may be linked whereby an (extended) local name $(P\ N)$ identifies a principal named as N in the namespace of a principal identified by local name P. For example, $(K_A\ \textsf{Bob Clare})$ is the principal named Clare in the namespace of the principal $(K_A\ \textsf{Bob})$. Hereafter, we use local names and/or public keys to reference principals.

Local name relationships are represented using the *speaks-for* relation whereby statement $P \rightarrow Q$ denotes that the principal Q speaks-for the principal P. For example, $(K_A\ \textsf{Bob}) \rightarrow K_B$ means that a message signed by the owner of K_B can be considered to originate from $(K_A\ \textsf{Bob})$. The following rewrite rule provides a speaks-for interpretation for name certificates. Note that for ease of exposition we ignore validity period V.

$$\frac{\{\!|N, P, V|\!\}_{sK}}{(K\ N) \rightarrow P}[\text{N1}]$$

Principal Name Reduction Speaks-for relationships may be reasoned over using SDSI name reduction. Given local names (or public keys) P, Q, R and a name N then:

$$\frac{P \rightarrow (Q\ N); Q \rightarrow R}{P \rightarrow (R\ N)}[\text{N2}]$$

For example, given (K_A Clare) → (K_A $\mathsf{Bob\ Clare}$) and (K_A Bob) → K_B then deduce (K_A Clare) → (K_B Clare). Given a collection of name certificates then we limit reasoning in the logic to determining whether $P \rightarrow Q$ can be deduced for given principals P and Q. In this case, it is safe for us to assume that the speaks-for relation is reflexive, that is $P \rightarrow P$. This simplifies our definition of the logic, and following [2] we assume that principal (P null) can be re-written as P and thus given principals P, Q and R then rule N2 can be used to infer that

$$(P \rightarrow Q \wedge Q \rightarrow R) \Rightarrow P \rightarrow R \tag{1}$$

holds.

2.2 SPKI Delegation

A delegation statement $P \overset{X}{\Longrightarrow} Q$ indicates that principal P delegates authority for permission X to principal Q. Delegation is implemented as a SPKI certificate $\{\!|P, X, D, V|\!\}_{sK}$, whereby the owner of public key K signs a statement that it trusts principal P for permission X. For ease of exposition we ignore the delegation bit D and validity period V. The following rule provides an interpretation for delegation.

$$\frac{\{\!|P, X, D, V|\!\}_{sK}}{K \overset{X}{\Longrightarrow} P}[\text{D1}]$$

Delegation Reduction Delegation statements may be reasoned over using SPKI reduction. Given principals P, Q, R, S and permissions X and Y then

$$\frac{P \overset{X}{\Longrightarrow} Q; Q \rightarrow R}{P \overset{X}{\Longrightarrow} R}[\text{D2}] \qquad \frac{P \overset{X}{\Longrightarrow} Q; Q \overset{Y}{\Longrightarrow} R;}{P \overset{X \sqcap Y}{\Longrightarrow} R}[\text{D3}]$$

where $X \sqcap Y$ denotes permission intersection. The set of all permissions $PERM$ may be considered to form a preorder $(PERM, \sqsubseteq)$ with intersection $\sqcap$ providing a lower bound operator. For example, the set of all s-expression permission tags used by SPKI/SDSI form a preorder with tag intersection providing a greatest lower bound operation.

3 Subterfuge in SPKI/SDSI

While Trust Management systems are typically explicit in their assumption that principals are uniquely identified, the literature has generally not been as prescriptive regarding the uniqueness of permissions. Delegation subterfuge [10] arises when there is ambiguity concerning the uniqueness and interpretation of a permission. This issue is considered in the following example.

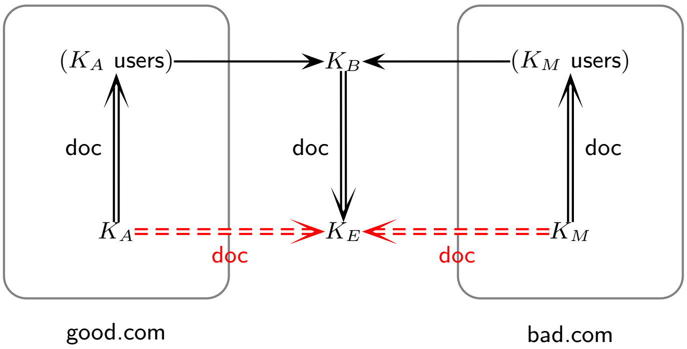


Fig. 1. Subterfuge in the delegation of permission doc

Suppose that the web-servers at Internet domains good.com and bad.com use Trust Management for controlling access to web-pages. The statement $K_A \overset{\mathsf{doc}}{\Longrightarrow}$ (K_A users) by the owner K_A of website good.com delegates authority to access web-page doc (representing a local document path) to a (SDSI) group of registered principals (K_A users) where (K_A users) $\rightarrow K_B$ and (K_A users) $\rightarrow K_C$. Suppose that principal K_B is also registered on website bad.com that is administrated by the (malicious) owner of K_M who in turn applies similar access controls on its group of users identified as (K_M users).

A delegation statement $K_B \overset{\mathsf{doc}}{\Longrightarrow} K_E$ results in subterfuge since it is not clear whether the (non-unique) permission doc refers to the authority to access the document on the good or bad websites. This uncertainty might arise in practice if K_B is unaware of this ambiguity. For example, K_M, intercepts the delegation certificate $K_A \overset{\mathsf{doc}}{\Longrightarrow}$ (K_A users) and replaces it by the certificate $K_M \overset{\mathsf{doc}}{\Longrightarrow}$ (K_M users), leading K_B to believe that permission doc is related to K_M's access to website bad.com. K_B is willing to grant K_E access the bad website and writes $K_B \overset{\mathsf{doc}}{\Longrightarrow} K_E$. However, K_E, colluding with K_M, can use the intercepted certificate $K_A \overset{\mathsf{doc}}{\Longrightarrow}$ (K_A users) to obtain access to the good website ($K_A \overset{\mathsf{doc}}{\Longrightarrow} K_E$).

It might be argued that this inadequacy in the permission design is 'obvious' and that additional information should be included in the name of the permission. For example, the permission good.com/doc is clearly related to its website. However, on receipt of a certificate $K_M \overset{\mathsf{good.com/doc}}{\Longrightarrow}$ (K_M users), K_B may unwittingly delegate $K_B \overset{\mathsf{good.com/doc}}{\Longrightarrow} K_E$, not understanding that K_M has no authority over good.com and the intercepted certificate $K_A \overset{\mathsf{good.com/doc}}{\Longrightarrow}$ (K_A users) can be used by K_E to obtain access to good.com. Furthermore, design of the permission good.com/doc assumes that there is a non-transient association between the domain good.com and some principal. However, domain name owners change in practice, intentionally or otherwise [14], and therefore, permission good.com/doc should not be considered to necessarily specify an unambiguous authorization.

Arguing that prior to issuing a delegation statement K_B has a responsibility to confirm that K_M owns the good.com website is inappropriate since it places part of the reasoning outside of, and is contrary to the intent of, the Trust Management system [4].

A variety of ad-hoc techniques could be used to ensure that a permission is globally unique and its interpretation unambiguous. For example, on the basis that public keys can be considered as unique then K_A:doc provides a unique permission doc for K_A's website. However, in order for this scheme to avoid subterfuge, the recognition of a permission string such as

$$\begin{pmatrix} \text{Modulus (1024 bits): c0 fd 51 7b 70 29 51 d7 d8 8d 59 c4 a1 bb da c9 fc c6 51 fc 90 b3 46 83 bd 45} \\ \text{22 98 47 1c e8 2c 56 2f fe 2c e4 d4 fd 4b 3d b4 8a 82 e0 e5 c8 08 4d fe 80 a7 cf d4 5f 4f 31 08 4d e5} \\ \text{e5 f0 14 e3 40 f1 12 4c b0 7f 97 b9 fa 29 c0 88 bf 23 8f bc b2 df 49 1c f6 72 a3 1f fa fe 83 11 c8 45} \\ \text{89 fb e4 1f fa 02 57 59 68 a5 d0 d8 a6 f0 29 9f eb d9 43 86 ea f9 1f 70 48 2d f1 4c e4 e7 70 43 b4 7f} \\ \text{Exponent (24 bits): 01 00 01} \end{pmatrix} : \mathsf{doc}$$

is required, which is, in itself, subject to confusion. Notwithstanding this issue, it is argued [15] that subterfuge can be avoided by including the originator of the permission in a delegation statement of the form $K_A \overset{K_o:p}{\Longrightarrow} K_B$, whereby principal K_A delegates the permission p, originating from the principal K_o, to the principal K_B. There is also an argument that a permission built using X500 Distinguished Names is, by definition, globally unique. If it were referenced in an extended validation certificate [1] then it is, in some legal sense, unambiguous, and is therefore not subject to subterfuge. However, X509-style approaches suffer from a variety of practical problems [7] when used to identify principals.

Delegation subterfuge is a consequence of non-unique permissions that have ambiguous interpretations in the sense of what they entitle the holder to do. Rather than relying on ad-hoc permission-naming strategies we are interested in characterizing what is meant by subterfuge and developing a Trust Management system that is subterfuge-free. A number of subterfuge scenarios and their defense are discussed in [10, 15]. It is argued in [15] that the problem of delegation subterfuge is analogous to the problem of a message freshness-attack in a security protocol and a BAN-like logic is developed that can be used to analyze a delegation scheme for subterfuge. In this paper we build on this and develeop an extension to SPKI/SDSI that ensures subterfuge-freedom.

4 Local Permission Names

Subterfuge-freedom can be achieved using delegation statements of the form $K_A \overset{K_o:p}{\Longrightarrow} K_B$ [15]. However, as observed above, simply referencing a public key within a permission identifier is impractical. SDSI's rationale of using local names as reliable references to principals is extended in this section to include local names for permissions that are linked to principal namespaces.

A signed permission $\{\!|N|\!\}_{sK}$ represents an authorization named N that originates from a principal owning public key K. On the basis of the assumption that a public key is considered to be globally unique then a permission signed by the key can be considered to be a globally unique permission identifier and is assumed unambiguous in the sense that, by signing $\{\!|N|\!\}_{sK}$, its originator K has just one interpretation for N.

A *local permission name* $\langle P\ N\rangle$ identifies a permission named locally as N in the namespace of the principal P. In general, a permission name certificate $\{\!|N, X|\!\}_{sK}$ is a statement signed by the owner of public key K that the permission with name N in K's local namespace is defined as X with the interpretation that a principal that holds permission $\langle K\ N\rangle$ may be considered to hold the permission X . For example, $\{\!|\mathsf{doc}, \{\!|\mathsf{doc}|\!\}_{sK_A}|\!\}_{sK_B}$ is a statement by K_B that when it refers to permission name doc in its namespace then it refers to signed permission $\{\!|\mathsf{doc}|\!\}_{sK_A}$.

4.1 Permission Holding

When defining (originating) a new permission, we assume that the principal signs a self-signed name certificate $\{\!|N, \{\!|N|\!\}_{sK}|\!\}_{sK}$ that binds the name N to the globally unique value $\{\!|N|\!\}_{sK}$. In doing this, the principal is considered to *hold* the permission, denoted $K \ni \langle K\ N\rangle$. Thus, given a key K and name string N we define the following inference rule.

$$\frac{\{\!|N, \{\!|N|\!\}_{sK}|\!\}_{sK}}{K \ni \langle K\ N\rangle}[\text{H1}]$$

For example, in defining permission doc by signing $\{\!|\mathsf{doc}, \{\!|\mathsf{doc}|\!\}_{sK_A}|\!\}_{sK_A}$, principal K_A is considered to hold the permission, that is, $K_A \ni \langle K_A\ \mathsf{doc}\rangle$.

Delegation of a permission does not necessarily imply that the recipient holds the permission: it depends on whether the delegator has (holds) the permission to give away in the first place. We have,

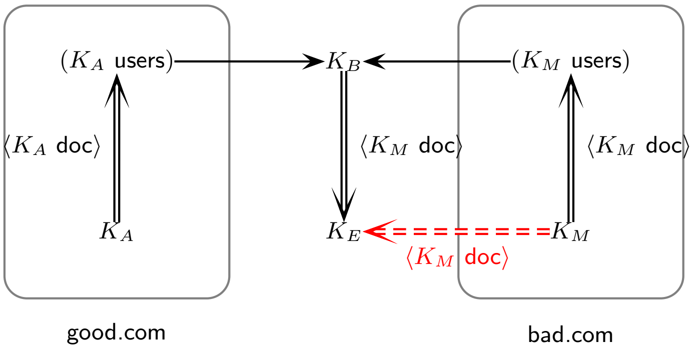
$$\frac{P \ni X, P \overset{X}{\Longrightarrow} Q}{Q \ni X}[\text{H2}]$$

In our delegation logic we make a distinction between a principal being delegated a permission and actually holding the permission. In determining whether a request (permission X) from Q is authorized, rather than just checking $P \overset{X}{\Longrightarrow} Q$ (SPKI/SDSI), the principal P should confirm $Q \ni X$ can be deduced. Figure 2 depicts the web-server example using local permissions with K_A delegating its local permission doc to its users group, whereupon by Rules H1 and H2, $(K_A\ \mathsf{users}) \ni \langle K_A\ \mathsf{doc}\rangle$ can be deduced.

If a principal can speak for another principal then the former is considered to implicitly hold the permissions of the latter, that is, given principals P and Q and local permission X then

$$\frac{P \ni X, P \rightarrow Q}{Q \ni X}[\text{H3}]$$

Returning to Figure 2, since K_B can speak for the group (K_A users), then K_B holds the permission $\langle K_A\ \mathsf{doc}\rangle$. Note that if K_B is unaware of the delegation



$(K_A$ users$)$
K_B
$(K_M$ users$)$
$\langle K_A$ doc$\rangle$
$\langle K_M$ doc$\rangle$
$\langle K_M$ doc$\rangle$
K_A
K_E
K_M
$\langle K_M$ doc$\rangle$
good.com
bad.com

Fig. 2. Avoiding ambiguity in delegation using local permissions

$K_A \overset{\langle K_A \text{ doc}\rangle}{\Longrightarrow} K_B$ (or it did not occur) then in the presence of a malicious delegation statement $K_M \overset{\langle K_A \text{ doc}\rangle}{\Longrightarrow} K_B$, K_B cannot deduce $K_B \ni \langle K_A$ doc$\rangle$, that is, K_B cannot mistakenly think that it holds the permission.

4.2 Permission Ordering

Conventional Trust Management systems assume that the set of all permissions are globally understood and implicitly have a globally defined preorder $(PERM, \sqsubseteq)$. For example, a preorder exists over conventional SPKI/SDSI s-expression permission tags and thus, for instance,

$$\text{(tag (http good.com/doc))} \sqsubseteq \text{(tag (http (* prefix good.com/)))}$$

However, local permissions are created locally and a principal must explicitly define how the permissions that it originates, relate to other permissions.

An ordering relation is defined between local permissions whereby $X \rightsquigarrow Y$ defines that permission Y dominates permission X, in the sense that Y is no less authoritative than X. In this case a principal that is authorized for permission Y is considered to be authorized for permission X. For example, $\langle K_B$ read$\rangle \rightsquigarrow \langle K_A$ readWrite$\rangle$ means that the holder of local permission $\langle K_A$ readWrite$\rangle$ also has authority for permission $\langle K_B$ read$\rangle$. Principals use permission certificates to define permission orderings: $\{\!|N, X|\!\}_{sK}$ is a statement by K that the permission X is no less authoritative than the permission $\langle K\ N\rangle$ in its namespace.

The following inference rule provides an interpretation, under this ordering, for permission certificates. Given public key K, permission X and name N then:

$$\frac{\{\!|N, X|\!\}_{sK}, K \ni X}{X \rightsquigarrow \langle K\ N\rangle}[\text{P1}]$$

Note that the principal K must hold the permission X over which it asserts authority ordering $X \rightsquigarrow \langle K\ N\rangle$ relative to the permission $\langle K\ N\rangle$ in its namespace.

If this were not the case then a malicious principal K_M could, for example, effectively obtain (hold) permission $\langle K_A\ \mathsf{doc}\rangle$ by simply signing $\{\!|\mathsf{doc}, \langle K_A\ \mathsf{doc}\rangle|\!\}_{sK_M}$, resulting in $\langle K_A\ \mathsf{doc}\rangle \leadsto \langle K_M\ \mathsf{doc}\rangle$.

The ordering relationship $\leadsto$ between permissions is, by definition, reflexive.

$$\frac{P \ni X}{X \leadsto X}[\text{P2}]$$

The relationship is defined only between valid permissions, that is permissions that originate from/are held by some principal. This avoids introduction of relationships between arbitrary values (that do not define permissions).

A principal holding permission Y, holds all permissions X dominated by Y.

$$\frac{P \ni Y; X \leadsto Y}{P \ni X}[\text{P3}]$$

Note that we assume that there is sufficient redundancy in the implementation of a permission name certificate that will enable a principal to distinguish a permission name from a principal name certificate, thereby providing distinction between the principal $(P\ N)$ and the permission $\langle P\ N\rangle$.

4.3 Permission Delegation

In delegating a permission Y to principal Q, principal P implicitly delegates authority for any permission X dominated by Y.

$$\frac{P \overset{Y}{\Longrightarrow} Q; X \leadsto Y}{P \overset{X}{\Longrightarrow} Q}[\text{P4}]$$

Continuing the web-server example, if K_B delegates its local name file for the $\mathsf{good.com}$ document permission as $K_B \overset{\langle K_B\ \mathsf{file}\rangle}{\Longrightarrow} K_D$ where $\langle (K_B\ \mathsf{good.com})\ \mathsf{doc}\rangle \leadsto \langle K_B\ \mathsf{file}\rangle$, then by L1 (defined in Section 4.5) we deduce $\langle K_A\ \mathsf{doc}\rangle \leadsto \langle K_B\ \mathsf{file}\rangle$ and by P4 we deduce K_B delegates the original permission, that is $K_B \overset{\langle K_A\ \mathsf{doc}\rangle}{\Longrightarrow} K_D$.

It follows from Rule P4 and Holding Rules H2 and H3 that if the delegating principal holds the permission then the recipient also holds any dominated permission:

$$(P \ni X \wedge P \overset{Y}{\Longrightarrow} Q \wedge X \leadsto Y) \Rightarrow Q \ni X \tag{2}$$

or more generally, any recipient holds the permissions:

$$(P \ni X \wedge P \overset{Y}{\Longrightarrow} Q \wedge X \leadsto Y \wedge Q \rightarrow R) \Rightarrow R \ni X \tag{3}$$

The conventional SPKI delegation reduction rule defines that the permission delegated via a delegation chain is computed as the intersection of the permissions along the chain. The definition of intersection as a greatest lower bound operation requires that all permissions and their ordering are globally known; in

SPKI/SDSI this is effectively defined as intersection over s-expression permission tags. However, in the proposed model it is assumed that a principal is aware only of the local permissions (and orderings) for which it holds permission name certificates. Therefore, the principal cannot compute a reliable greatest lower bound of given permissions since there may exist, unknown to the principal, a permission that is a lower bound and dominates the claimed greatest lower bound.

Based on the permission orderings that a principal is aware of, it is safe for a principal to infer that any permission that is a lower bound of permissions X and Y is dominated by the greatest lower bound of X and Y. This is defined by the permission intersection reduction rule

$$\frac{Z \rightsquigarrow X; Z \rightsquigarrow Y}{Z \rightsquigarrow (X \sqcap Y)}[\text{P5}]$$

Combining this rule with the SPKI-delegation rule in Section 2.2 allows the following inference to be made. Given principals P, Q, R and permissions X, Y, Z then we have

$$(P \overset{X}{\Longrightarrow} Q \wedge Q \overset{Y}{\Longrightarrow} R \wedge Z \rightsquigarrow X \wedge Z \rightsquigarrow Y) \Rightarrow P \overset{Z}{\Longrightarrow} R \qquad (4)$$

4.4 Delegation Accountability

A principal is considered to be *accountable* for a permission if it accepts responsibility for how the permission is used by other principals. For example, in hosting documents on good.com, the principal K_A is considered to accept responsibility for the use of the (copyright) documents. In delegating access to documents to K_B, principal K_A asserts that she accepts responsibility for how the documents are subsequently handled by K_B (that is, K_A trusts K_B). This notion of accountability is at the heart of subterfuge. Subterfuge is considered to occur when there is ambiguity regarding the accountability of the permission. In the original web-server example (Figure 1) subterfuge occurs because accountability for the actions authorized by permission doc is unclear; in delegating doc to K_E, K_B wants to be sure about who can be held accountable.

A principal with public key K that originates a permission $\langle K\ N\rangle$ is considered, by definition, to be accountable, denoted $K \rhd \langle K\ N\rangle$ for any actions enabled by that permission. We have,

$$\frac{K \ni \langle K\ N\rangle}{K \rhd \langle K\ N\rangle}[\text{A1}]$$

For example, by signing a permission granting access to the good.com web-site documents then K_A is implicitly accepting accountability for the use of those documents.

A principal K may elect to accept accountability for an arbitrary permission X that it holds by signing a statement to that effect, and we have:

$$\frac{\{\!|\mathsf{accept\_accountability}(X)|\!\}_{sK}, K \ni X}{K \rhd X}[\text{A2}]$$

For the web-server example, perhaps K_A choses not to delegate her doc authority on good.com to K_B unless K_B signs a statement accepting accountability, in which case $K_B \rhd \langle K_A\ \mathsf{doc}\rangle$. Note that the principal asserting accountability must hold the permission; this ensures that a malicious principal cannot claim accountability for a permission for which they are not trusted. For the web-server example, regardless of an assertion by K_M that it is willing to be accountable for $\langle K_A\ \mathsf{doc}\rangle$, K_B cannot deduce $K_M \rhd \langle K_A\ \mathsf{doc}\rangle$. Therefore, as we shall see in the next section, K_M maliciously concealing $K_A \overset{\langle K_A\ \mathsf{doc}\rangle}{\Longrightarrow} (K_A\ \mathsf{users})$ from K_B and replacing it by $K_M \overset{\langle K_A\ \mathsf{doc}\rangle}{\Longrightarrow} (K_M\ \mathsf{users})$, cannot result in K_B unwittingly delegating this unaccountable permission $\langle K_A\ \mathsf{doc}\rangle$ to K_E.

A local name may be used to refer to the originator of the permission:

$$\frac{Q \rhd X, P \rightarrow Q}{P \rhd X}[\text{A3}]$$

Thus, in general $P \rhd X$ reflects that there exists *some* principal who can speak for P and that can be held accountable for permission X. Note that the converse is not the case, for example, membership of a group P that is accountable for some permission does not necessarily imply accountability of an arbitrary member. Similarly, if a member holds a permission and asserts accountability then while the group implicitly holds the accountability, it does not necessarily hold that the group holds the permission, since by Rule H3, it would imply that all its members also hold the permission.

Accountability follows on reduction of a principal name referenced within a local permission, that is,

$$\frac{R \rhd \langle P\ N\rangle, P \rightarrow Q}{R \rhd \langle Q\ N\rangle}[\text{A4}]$$

Thus, for example, a principal accepting accountability for a group permission accepts accountability for any any group-member reference to that permission. Note, however, that we do not define a similar relationship between permission ordering and accountability. One should not infer $Q \rhd X$ given $Q \rhd Y$ and $X \rightsquigarrow Y$; if this were permitted then should K_B name his own permission $\langle K_B\ \mathsf{file}\rangle$ and assert $\langle K_A\ \mathsf{doc}\rangle \rightsquigarrow \langle \mathsf{K_B}\ \mathsf{file}\rangle$ then since K_B is by default accountable for all the permissions he names then he would hold accountability for the $\langle K_A\ \mathsf{doc}\rangle$ permission.

A valid permission must originate from some principal and, therefore, the principal is accountable for that permission, that is we can prove:

$$Q \ni \langle Q\ N\rangle \Rightarrow Q \rhd \langle Q\ N\rangle \tag{5}$$

This effectively generalizes Rule A1—that a key is accountable for any permission it originates—to any principal name for which the key speaks for.

4.5 Local Permission Name Reduction

Local principal names can be used to refer to principals in a local permission ordering. We define

$$\frac{\langle P\ N\rangle \rightsquigarrow X; P \rightarrow Q}{\langle Q\ N\rangle \rightsquigarrow X}[\mathrm{L1}]$$

For the web-server example, suppose that K_B uses the local (principal) name (K_B $\mathsf{good.com}$) to refer to K_A, that is, (K_B $\mathsf{good.com}$) → K_A and suppose that K_B refers to the permission locally as $\langle K_B\ \mathsf{file}\rangle$, that is, $\langle(K_B\ \mathsf{good.com})\ \mathsf{doc}\rangle \rightsquigarrow \langle K_B\ \mathsf{file}\rangle$. In this case, by Rule L1, we can deduce that $\langle K_A\ \mathsf{doc}\rangle \rightsquigarrow \langle K_B\ \mathsf{file}\rangle$.

If $P \ni \langle Q\ N\rangle$ then it follows that $\langle Q\ N\rangle$ is a valid permission and thus (reflexivity) $\langle Q\ N\rangle \rightsquigarrow \langle Q\ N\rangle$ and if $Q \rightarrow R$ then by L1 and P3 the following holds.

$$P \ni \langle P\ N\rangle \wedge P \rightarrow Q \Rightarrow \langle Q\ N\rangle \rightsquigarrow \langle P\ N\rangle \tag{6}$$

The permission ordering $\rightsquigarrow$ is intended to be reflexive and transitive. Transitivity can be influenced by the reduction of a local principal name referenced within a local permission name. Permission reduction is defined by the following rule, whereby given principals P and Q, permissions X and Y and a name N then

$$\frac{X \rightsquigarrow \langle P\ N\rangle; P \rightarrow Q; \langle Q\ N\rangle \rightsquigarrow Y; Q \rhd \langle P\ N\rangle}{X \rightsquigarrow Y}[\mathrm{L2}]$$

If we consider well-defined (held by principals) permissions X, Y and Z then by reflexivity of → and $\rightsquigarrow$ it follows that permission ordering is transitive in the sense that:

$$X \rightsquigarrow Y \wedge Y \rightsquigarrow Z \wedge Q \rhd Y \Rightarrow X \rightsquigarrow Z \tag{7}$$

One can consider proposition (7) in the context of a conventional Trust Management system whereby some ‘super security authority’ effectively asserts a global preorder over permissions $(PERM, \sqsubseteq)$. For example, the ordering over s-expression tags as implicitly defined by [13]. This ‘super user’, corresponding to Q in Proposition (7), can, in a sense, be regarded as accepting accountability for the ordering and thus the set $(PERM, \rightsquigarrow)$ forms a preorder. In the truly open/decentralized scenario there is not one but a number of separate super security authorities, each asserting accountability and defining the ordering over the permissions that originate within their domain.

Rule L2 supports inferences based on principal name reduction, that is,

$$\langle Q\ N\rangle \rightsquigarrow X \wedge P \rightarrow Q \wedge Q \rhd \langle P\ N\rangle \Rightarrow \langle P\ N\rangle \rightsquigarrow X \tag{8}$$

and

$$X \rightsquigarrow \langle P\ N\rangle \wedge P \rightarrow Q \wedge Q \rhd \langle P\ N\rangle \Rightarrow X \rightsquigarrow \langle Q\ N\rangle \tag{9}$$

Thus, for the web-server example, suppose again that K_B uses local (principal) name (K_B $\mathsf{good.com}$) to refer to K_A, that is, (K_B $\mathsf{good.com}$) $\rightarrow K_A$, and uses local permission file to refer to K_A's document permission, that is, $\langle K_A\ \mathsf{doc}\rangle \rightsquigarrow \langle \mathsf{K_B}\ \mathsf{file}\rangle$. As originator of the permission we have $K_A \rhd \langle K_A\ \mathsf{doc}\rangle$, and on the basis that (K_B $\mathsf{good.com}$) is a local name for K_A, then by Rule A3 we have (K_B $\mathsf{good.com}$) $\rhd \langle K_A\ \mathsf{doc}\rangle$ and in turn, by Rules L1 and A4 we can deduce (K_B $\mathsf{good.com}$) $\rhd \langle (K_B\ \mathsf{good.com})\ \mathsf{doc}\rangle$.

By reflexivity $\langle Q\ N\rangle \rightsquigarrow \langle Q\ N\rangle$ on well-defined permissions, it follows from Proposition (8) that

$$P \rightarrow Q \wedge Q \rhd \langle P\ N\rangle \Rightarrow \langle P\ N\rangle \rightsquigarrow \langle Q\ N\rangle \qquad (10)$$

The reader should compare this proposition (10) with

$$P \rightarrow Q \wedge Q \ni \langle P\ N\rangle \Rightarrow \langle Q\ N\rangle \rightsquigarrow \langle P\ N\rangle \qquad (11)$$

which can be derived from Rule L1 (by reflexivity $\langle P\ N\rangle \rightsquigarrow \langle P\ N\rangle$). Intuitively, as far as permission ordering is concerned, there is no distinction between using a group name or its member name in a permission when the member is willing to be accountable. However, if the member is not willing to be held accountable ((10) does not apply), then it is not necessarily the case that a permission in the members name space is as authoritative as that permission for the group. This also illustrates why, within the logic, it is not considered safe to arbitrarily apply principal name reduction to principal names within permissions.

5 Subterfuge-Safe Trust Management

A conventional Trust Management compliance check, given a collection of delegation statements/credentials, corresponds to a query: *(by principal P) is it safe to carry out the action authorized by permission X, as requested by principal Q?* This is evaluated by determining whether $P \overset{X}{\Longrightarrow} Q$ can be deduced. If the principal P originates the permission as $X = \langle P\ N\rangle$ then it follows that the principal holds ($P \ni X$) and is accountable ($P \rhd X$) for the permission and that there is no ambiguity as to the meaning of the permission.

However, P may wish to carry out a compliance check on permissions that it did not originate. For the web-server example, perhaps K_B provides a mashup that includes documents from $\mathsf{good.com}$, and checks whether the requester holds a permission $\langle (K_B\ \mathsf{good.com})\ \mathsf{doc}\rangle$ that originated from K_A. In this case, K_B will want to be sure that the permission can be tied to a principal willing to be held accountable, that is that (K_B $\mathsf{good.com}$) $\rhd \langle (\mathsf{K_B}\ \mathsf{good.com})\ \mathsf{doc}\rangle$. A check for authorization is therefore defined as follows.

5.1 Checking for Subterfuge-Safe Authorization

A subterfuge-safe compliance check corresponds to the query: *(by principal P) is it safe to carry out the action authorized by permission X, as requested by* *principal Q with principal R held accountable?* This is evaluated by determining whether the requester holds the permission ($Q \ni X$) and that R is accountable ($R \rhd X$) can be deduced. In this paper we assume that the principal claimed to be accountable is provided in the query. Future research will consider how an accountable principal might be searched for as part of the query.

Note that P may wish to check, in addition, whether it delegated the authority to Q ($P \overset{X}{\Longrightarrow} Q$) or whether Q received the permission X from another source (that is presumably trusted by P, by virtue of P willingness to query $Q \ni X$).

5.2 Subterfuge-Safe Delegation

Before signing a delegation statement a principal should determine whether it might lead to subterfuge, in particular the delegating principal should check that some principal can be held accountable for actions associated with the permission.

A subterfuge-safe delegation check corresponds to the query: *(by principal P) is it safe to delegate permission X to Q whereby R can be held accountable?* This is evaluated by determining whether R is accountable ($R \rhd X$) and that P trusts the principal providing the accountability ($P \overset{X}{\Longrightarrow} R$). If the check succeeds then P asserts $P \overset{X}{\Longrightarrow} Q$.

Reconsider the subterfuge attack from Section 3, but where local permissions are used. Suppose that K_M intercepts the delegation $K_A \overset{\langle K_A\ \mathsf{doc}\rangle}{\Longrightarrow} K_B$ so that K_B is unaware of its existence. Principal K_M asserts $K_M \overset{\langle (K_M\ \mathsf{bad})\ \mathsf{doc}\rangle}{\Longrightarrow} K_B$, and informs K_B that $(K_M\ \mathsf{bad}) \rightarrow K_M$. However, K_M conceals from K_B that $(K_M\ \mathsf{bad}) \rightarrow K_A$ and therefore, its not unreasonable for K_B to mistakenly think that $\langle (K_M\ \mathsf{bad})\ \mathsf{doc}\rangle$ is a permission related to the bad domain. In thinking this, K_B wishes to delegate the permission (with $(K_M\ \mathsf{bad})$ accountable) to K_E who is considered to be associated with the bad domain. However, this delegation is not subterfuge-safe since it is not possible to derive $(K_M\ \mathsf{bad}) \rhd \langle (K_M\ \mathsf{bad})\ \mathsf{doc}\rangle$. While K_M does originate $\langle K_M\ \mathsf{doc}\rangle$ and by rule A3 we can infer $(K_M\ \mathsf{bad}) \rhd \langle K_M\ \mathsf{doc}\rangle$, however, we cannot infer that the principal is also accountable for $\langle (K_M\ \mathsf{bad})\ \mathsf{doc}\rangle$.

Continuing the web-server example, the malicious principal K_M could elect to assert accountability for this permission and thus $(K_M\ \mathsf{bad}) \rhd \langle (K_M\ \mathsf{bad})\ \mathsf{doc}\rangle$ can be derived. This highlights an underlying assumption in the logic that a declaration of accountability is taken as formal evidence of the principal's willingness to be held to account, regardless of their actual intent or reputation outside of the logic. Therefore, subterfuge-safe delegation also requires that the accountable principal is trusted by the delegator. In this case K_B does not trust permissions issued by K_M, that is, $K_B \overset{\langle (K_M\ \mathsf{bad})\ \mathsf{doc}}{\Longrightarrow} K_M$ cannot be derived. Note that the current version of the logic assumes that if we trust a principal for some permission then we are willing to accept any assertion the principal may make over their willingness to accept accountability for that permission. While it is possible to avoid this by using separate permissions to reflect authorization versus accountability, distinguishing between trust for authorization versus trust for accountability within the logic is a topic for future research.

6 Discussion and Conclusion

Signed permissions are an effective approach to avoiding ambiguity in permission names. This paper follows SDSI's rationale for local principal names and proposes an extension to SPKI/SDSI that uses local permission names in order to provide support signed permissions and thereby provide an authorization language that is delegation subterfuge-safe. The logic supports truly decentralized Trust Management whereby a principal may define, without reference to any central authority, its own local permission namespace, define a permission ordering over that namespace and also orderings relative to permissions in other namespaces.

Typical trust management/distributed authorization systems make the implicit assumption that there exists a 'super security authority' that defines the permission namespace and ordering. In [16] a role-based distributed authorization language is described that provides subterfuge-freedom by constraining delegation to permissions that have an associated 'originating' public key. While effective, this approach suffers the challenge of reliably referencing public keys and relies on a globally-defined function to define permission relationships (corresponding to ordering). The FRM distributed policy management framework [8,9] permits principals to locally define their permissions and orderings, and while it does permit a principal to define permission relationships with local policies of other principals, it is limited to permission orderings that form tree-hierarchies. FRM also uses signed permissions to avoid subterfuge, but effectively relies on using public key values/X509 certificates as principal identifiers.

The proposed logic is comprised of 13 axioms in addition to the 5 axioms that describe SPKI/SDSI. The focus of this paper has been to propose and develop an understanding for linked local permission names. While the 10 propositions derived from these axioms provide some degree of confidence in the logic, future work will develop a semantics in order to demonstrate soundness and completeness. Like Subterfuge Logic [15], this paper does not characterize subterfuge as a behavioral property, rather it is implicit in its axioms and the interpretation of accountability. Given the analogy between subterfuge-attacks on certificate chains and freshness-attacks on authentication protocols [15], we are currently investigating an attacker-model approach based on [11] to verify the proposed model.

Acknowledgments This research is supported by Science Foundation Ireland grant 08/SRC/11403. The authors would like to thanks the anonymous reviewers for their helpful feedback.

References

1. Guidelines for the issuance and management of extended validation certificates. Tech. rep., CA/Browser Forum (2009), http://cabforum.org/Guidelines_v1_2.pdf
2. Abadi, M.: On sdsi's linked local name spaces. In: Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97). p. 98. IEEE Computer Society, Washington, DC, USA (1997)
3. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The keynote trust-management system, version 2 (September 1999)
4. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance checking in the policymaker trust management system. In: FC '98: Proceedings of the Second International Conference on Financial Cryptography. pp. 254–274. Springer-Verlag, London, UK (1998)
5. CCITT Draft Recomendation: The Directory Authentication Framework, Version 7 (Nov 1987)
6. Clarke, D., Elien, J., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L.: Certificate chain discovery in spki/sdsi. Journal of Computer Security 9(4), 285–322 (2001)
7. Ellison, C.: The nature of a usable PKI. Computer Networks 31, 823–830 (1999)
8. Feeney, K., Lewis, D., D.O'Sullivan: Service oriented policy management for web-application frameworks. IEEE Internet Computing Magazine (13):6, 39–47 (Nov/Dec 2009)
9. Feeney, K., Brennan, R., Foley, S.N.: A trust model for capability delegation in federated policy systems. In: International Conference on Network and Service Management. pp. 226–229. IEEE (2010)
10. Foley, S.N., Zhou, H.: Authorisation subterfuge by delegation in decentralised networks. In: International Security Protocols Workshop. Cambridge, UK (April 2005)
11. Foley, S.: Noninterference analysis of delegation subterfuge, IEEE Computer Security Foundations Workshop, short-presentations, 2006
12. Li, J., Li, N., Winsborough, W., J.C. Mitchell, Distributed Credential Chain Discovery in Trust Management, Journal of Computer Security, 11(1), 2003.
13. Rivest, R.: S-expressions. Internet Draft draft-rivest-sexp-00.txt, IEFT Network Working Group (1997)
14. Zeller, T.: Purloined domain name is an unsolved mystery. In: New York Times (Jan 18 2005)
15. Zhou, H., Foley, S.N.: A logic for analysing subterfuge in delegation chains. In: Workshop on Formal Aspects in Security and Trust (FAST2005). Newcastle upon Tyne, UK (July 2005)
16. Zhou, H., Foley, S.N.: A framework for establishing decentralized secure coalitions. In: Proceedings of IEEE Computer Security Foundations Workshop. IEEE CS Press (2006)