Noninterference Analysis of Delegation Subterfuge in Distributed Authorization Systems

Simon Foley

Noninterference Analysis of Delegation Subterfuge in Distributed Authorization Systems

Simon. N. Foley

Department of Computer Science,
University College Cork, Ireland
s.foley@cs.ucc.ie

Abstract. A principal carrying out a delegation may not be certain about the state of its delegation graph as it may have been perturbed by an attacker. This perturbation may come about from the attacker concealing the existence of selected delegation certificates and/or injecting new delegation certificates. As a consequence of this *delegation subterfuge* the principal may violate its own policy that guides delegation actions. This paper considers the verification of the absence of subterfuge in systems that accept and issue delegation certificates. It is argued that this absence of subterfuge is not a safety property and a non-interference style security-property based interpretation is proposed.

1 Introduction

Trust Management systems [3,5,17,18] provide a decentralized approach for managing delegation of trust between principals. These systems are typically explicit in their assumption that principals can be tied to an unambiguous identification, for example, Alice with her unique public key. However, the literature has generally not been as prescriptive in terms of how permission identifiers should be tied to the actions that they authorize. While central authorities such as the Internet Corporation for Assigned Names and Numbers (ICANN) provide identifiers that could be used for this purpose, a malicious principal can still choose to ignore or misrepresent the interpretation. *Delegation subterfuge* [10] arises when there is ambiguity in interpreting a delegated permission. This can come about from an attacker perturbing a victim's delegation graph by concealing and/or injecting delegation certificates. As a consequence, the victim may violate the requirements that guide its own delegation actions.

A number of subterfuge scenarios and their defense have been previously considered [10, 12]. Rather than presuming correct operation of ad-hoc permission-naming strategies we are interested in characterizing what is meant by subterfuge and in designing delegation schemes that can be proven to be subterfuge-free. It is argued [23] 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 consider the verification of the absence of subterfuge in applications and mechanisms that accept and issue delegation certificates. Using

a running example, we argue that subterfuge-freedom should not be treated as an Alpern-Schneider [1] safety-style property. It is insufficient for an application to decide whether it is safe to delegate, based on its view of a delegation graph (current state), as this view may have been perturbed by an attacker. In deciding whether to delegate, the application must also consider that other delegation graph configurations exist that may be as equally valid as the current state, based on the available information. Therefore, we conjecture that subterfuge-freedom is not a property on a single state but a property on sets of states. A non-interference [11, 13] style property is proposed for delegation subterfuge. Demonstrating and encoding subterfuge-freedom as a security (confidentiality) property is the primary contribution of this paper.

The paper is organized as follows. In Section 2 we describe a simple SPKI-based delegation model that is sufficient to present the results of this paper. Section 3 presents an example of a subterfuge attack on a service reseller application that uses delegation to manage trust relationships. Section 4 argues that treating the problem as a safety property is insufficient as it does not prevent the subterfuge-attack in the application. Section 5 proposes a non-interference style property to characterize subterfuge freedom and demonstrates its interpretation in the service reseller example. Related work is discussed in Section 6 and Section 7 concludes the paper.

2 Authorization Delegation

A delegation statement $P \stackrel{X}{\Longrightarrow} Q$ defines that principal P delegates authority for permission X to principal Q. Delegation can be implemented, for example, as a SPKI certificate [7] $\{\!|\ Q, X, D, V\ |\!\}_{sK}$, whereby the owner of public key K signs a statement that it trusts principal Q for permission X. For the sake of simplicity in this paper we ignore the delegation bit D and validity period V. The following inference rule provides an interpretation for delegation certificates.

$$\frac{\{\!|\ Q, X, D, V\ |\!\}_{sK}}{K \stackrel{X}{\Longrightarrow} Q} \quad [\ D1\ ]$$

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

$$\frac{P \stackrel{X}{\Longrightarrow} Q;\ Q \stackrel{Y}{\Longrightarrow} R;}{P \stackrel{X \sqcap Y}{\Longrightarrow} R} \quad [\ D2\ ] \qquad \frac{P \stackrel{Y}{\Longrightarrow} Q;\ X \sqsubseteq Y}{P \stackrel{X}{\Longrightarrow} Q} \quad [\ D3\ ]$$

where $X \sqsubseteq Y$ denotes permission ordering and $X \sqcap Y$ permission intersection. The set of all permissions $PERM$ may be considered to form a preorder $(PERM, \sqsubseteq)$ with intersection $\sqcap$ providing a greatest 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 Delegation Subterfuge

3.1 Example: Trust Management for service reselling

Principal Reese agrees to act as a reseller of hotel rooms offered by Harry and Mike. When reselling a room, Reese decides a room resell rate based on her business contract with the hotel and issues customers with an unforgable room resell rate agreement to be presented on arrival to the hotel. Customers pay the hotel directly for their room according to the amount specified in the resell rate. The arrangement is that, in reselling a room, Reese provides the hotel with a guaranteed room rate. Any surplus between the room resell rate and the guaranteed rate is passed on to Reese, while Reese is liable for a deficit.

The trust relationships in this scenario are encoded as SPKI style delegation statements. Note that in this paper, we interchangeably refer to a principal by its name $R(eese)$ or by its public key K_R, when no ambiguity can arise.

Reese and hotel Harry enter contracts by issuing $Reese \overset{\mathsf{contract}(r,v)}{\Longrightarrow} Harry$ and $Harry \overset{\mathsf{contract}(r,v)}{\Longrightarrow} Reese$ for guaranteed rate v for room r. Harry subsequently issues $Harry \overset{\mathsf{resell}.r.*}{\Longrightarrow} Reese$, delegating reselling authority for room r to Reese. The second attribute of permission resell specifies the rate at which the room is sold and the wildcard reflects that Harry places no constraint on the resell rate. We can treat the wildcard as an upper bound on resell permissions whereby $\mathsf{resell}.r.u \sqsubseteq \mathsf{resell}.r.*$, that is, a holder of permission $\mathsf{resell}.r.*$ is authorized (by inference rule $D3$) to resell the room for any rate u. In general, a principal authorized to resell a room at rate v can also sell it at any rate u higher than v, that is, $\mathsf{resell}.r.u \sqsubseteq \mathsf{resell}.r.v \Leftrightarrow v \leq u$.

For example, suppose that Reese guarantees a \$50 room rate for Harry. On reselling this room to Clare for \$60, Reese issues $Reese \overset{\mathsf{resell}.r.60}{\Longrightarrow} Clare$. On check-in, Clare presents delegation chain

$$Harry \overset{\mathsf{resell}.r.*}{\Longrightarrow} Reese;\ Reese \overset{\mathsf{resell}.r.60}{\Longrightarrow} Clare$$

which Harry reduces (by Rule $D2$) to $Harry \overset{\mathsf{resell}.r.60}{\Longrightarrow} Clare$, as proof that Clare is authorized for the \$60 room rate. This chain, along with the contract certificates are used by Harry and Reese in claiming reimbursement of any deficit/surplus (in this case, a surplus for Reese).

We are not concerned with the claim process in this paper, however, we are interested in Reese ensuring that she never resells a room below some minimum rate *minRate* that she decides. Table 1 gives sample guaranteed (contracted) and minimum rates for any rooms in hotels Harry and Mike. We assume that Reese may be willing to sell a room at a loss, for example, when it is bundled as part of a package that is profitable overall. These minimum rates are decided by Reese, and are represented as a delegation statement $Reese \overset{\mathsf{rate},v}{\Longrightarrow} Harry$ indicating that Reese is willing to resell any room in hotel Harry at rate v or higher. For simplicity we assume that all rooms in the hotel are the same. If Reese is willing to resell a room at rate v then it follows she is willing to resell the

Hotel	*Guarantee*	*minRate*
Harry	50	40
Mike	20	10

Table 1. Contracted guaranteed and minimum-sell room rates for Reese.

room at any higher rate u where $u \geq v$; thus, we define the permission ordering $\mathsf{rate}.u \sqsubseteq \mathsf{rate}.v \Leftrightarrow u \geq v$. Note that this rate delegation statement is used by Reese internally to implement the *minRate* relationship and it is not necessary for her to share the corresponding certificates with any other principal. For example, *Reese* $\overset{\mathsf{rate}.40}{\Longrightarrow}$ *Harry*; *Reese* $\overset{\mathsf{rate}.10}{\Longrightarrow}$ *Mike* implement the minimum rates in Table 1.

3.2 Subterfuge

Consider the following reselling scenario. Suppose that (malicious) Mike interferes with communication between hotel Harry and reseller Reese, intercepts the delegation certificate *Harry* $\overset{\mathsf{resell.r}.*}{\Longrightarrow}$ *Reese*, and replaces it by *Mike* $\overset{\mathsf{resell.r}.*}{\Longrightarrow}$ *Reese*, leading Reese to believe that permission resell.r.* is related to a room at Mike's hotel. Eve, who is colluding with Mike, then uses Reese's website to book this room r for a cost of $20, in compliance with Reese's minimum-rate policy in Table 1. Reese issues a certificate for *Reese* $\overset{\mathsf{resell.r}.20}{\Longrightarrow}$ *Eve*. However, Eve obtains the intercepted certificate *Harry* $\overset{\mathsf{resell.r}.*}{\Longrightarrow}$ *Reese* from Mike and offers this, along with *Reese* $\overset{\mathsf{resell.r}.20}{\Longrightarrow}$ *Eve*, as proof to Harry that she is authorized for this rate at his hotel.

It could 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, one could argue that permission mike.com/resell.r.20 is clearly related to Mike's website/hotel. However, on receipt of a certificate

$$\mathit{Mike} \overset{\mathsf{harry.com/resell.r}.*}{\Longrightarrow} \mathit{Reese}$$

Reese may unwittingly delegate *Reese* $\overset{\mathsf{harry.com/resell.r.20}}{\Longrightarrow}$ *Eve*, not understanding that Mike has no authority over harry.com and that the intercepted certificate *Harry* $\overset{\mathsf{harry.com/resell.r.20}}{\Longrightarrow}$ *Reese* can be used by Eve to obtain a room at Harry's hotel for $20. Furthermore, design of the permission harry.com/resell.r.* assumes that there is a non-transient association between the domain harry.com and a (hotel) principal. However, domain name owners change in practice, intentionally or otherwise [22], and therefore, permission harry.com/resell.r.* should not be considered to necessarily specify an unambiguous authorization. Arguing that prior to issuing a delegation statement that Reese has a responsibility to confirm that Mike owns the Harry.com domain is inappropriate. This presumes authentication of Mike's identity and places part of the reasoning outside of, and is contrary to the intent of, the Trust Management system [5]. Moreover, with the

declining numbers of system administrators relative to the number of Internet domains [16], it becomes more difficult for organizations to maintain a consistent view of domains/permissions.

3.3 Eliminating Subterfuge

Various ad-hoc techniques can be used to ensure unambiguous interpretation of a permission. For example, on the basis that public keys are considered unique and if Harry owns public key K_H then signed permission $\{\!|\ \mathsf{resell.r.*}\ |\!\}_{sK_H}$ provides a unique and unambiguous permission identifier that can be tied to Harry. It is argued [23] that subterfuge can be avoided by including the originating principal K_O of the permission p in a delegation statement of the form $K_A \overset{\{|p|\}_{sK_O}}{\Longrightarrow} K_B$, whereby principal K_A delegates the permission p, originating from the principal K_O, to the principal K_B. In this case we revise the reduction rule $D2$ to the following. Given principals (public keys) P, Q, R and permissions X, Y then

$$\frac{P \overset{\{|X|\}_{sP}}{\Longrightarrow} Q;\ Q \overset{\{|Y|\}_{sP}}{\Longrightarrow} R}{P \overset{\{|X \sqcap Y|\}_{sP}}{\Longrightarrow} R} \quad [\ D2'\ ]$$

reflecting that principal P is originator of the permission.

For the purposes of this paper we assume that permission signing is implemented in such a way that given $\{\!|\ X\ |\!\}_{sP}$ then another principal can also refer to any signed permission $\{\!|\ X'\ |\!\}_{sP}$ where $X' \sqsubseteq X$, for instance, in a subsequent delegation. For example, one might implement the signing of permission $\mathsf{resell.r.*}$ as expression $\{\!|\ \mathsf{resell.r.}(v \geq 0)\ |\!\}_{sHarry}$ which constrains the values of its free variable v. On receipt of this permission from Harry, Reese can use expression $\{\!|\ \mathsf{resell.r.}(v \geq 0)\ |\!\}_{sHarry} \backslash [v \leftarrow 50]$, binding the value 50 to the free variable v, in order to represent the permission $\mathsf{resell.r.50}$ signed by Harry. In general, we argue that the extent to which a principal can refer to signed permissions depends on the design of the delegation logic. For example, a principal may only refer to signed permissions that it has witnessed [12].

Returning to the reselling example, customer *Clare* presents the chain

$$\mathit{Harry} \overset{\{|\mathsf{resell.r.*}|\}_{sHarry}}{\Longrightarrow} \mathit{Reese};\qquad \mathit{Reese} \overset{\{|\mathsf{resell.r.50}|\}_{sHarry}}{\Longrightarrow} \mathit{Clare}$$

to Harry, who, using rule $D2'$, can verify that $\mathit{Harry} \overset{\{|\mathsf{resell.r.50}|\}_{sHarry}}{\Longrightarrow} \mathit{Clare}$. Reconsidering the subterfuge attack, even if Mike delegates a copy of the signed permission $\{\!|\ \mathsf{resell.r.*}\ |\!\}_{sHarry}$ originating from Harry and tricks Reese into thinking he (Mike) is Harry, then when Eve presents the chain

$$\mathit{Mike} \overset{\{|\mathsf{resell.r.*}|\}_{sHarry}}{\Longrightarrow} \mathit{Reese};\qquad \mathit{Reese} \overset{\{|\mathsf{resell.r.20}|\}_{sHarry}}{\Longrightarrow} \mathit{Eve}$$

to Harry, then, as originator, Harry can not infer $\mathit{Harry} \overset{\{|\mathsf{resell.r.20}|\}_{sHarry}}{\Longrightarrow} \mathit{Eve}$.

Given the simplicity of the delegation scheme described above it is not unreasonable to rely on an intuitive argument that subterfuge is eliminated when one uses the revised rule $D2'$ for certificate reduction. However, the intuitive argument is less convincing for more complex/expressive delegation languages, such as [12], which claim to be subterfuge-free. Therefore, we are interested in characterizing subterfuge-freedom as a property so that the operation of a delegation scheme, such as that used in reseller example, can be verified to be subterfuge-free.

4 Delegation as a safety property

A delegation system is defined to be a system that carries out operations on behalf of a principal and based on a collection of delegation certificates that it maintains. The implementation of Reese's reselling service, along with its interpretation of delegation, is an example of a delegation system.

Delegation State. Let *STATE* represent the set of all possible states of a delegation system. For the purposes of this paper we do not consider behavioral properties of the system and define a state g to be simply the current delegation graph that is accessible by the system. This graph may be stored locally by the system, hosted by an authorization server, distributed among peers, or some combination. Given state g then $P \stackrel{X}{\Longrightarrow}_g Q$ denotes delegation of permission X by principal P to principal Q in state g.

Delegation System. The implementation of a delegation system is characterized in terms of a predicate $System(g)$, whereby $System(g)$ is true iff delegation network g is a reachable state of the implementation.

Delegation Policy. Let a *delegation policy* be a set of delegation states that are considered to be valid. A policy is defined as a predicate $Policy(g)$, whereby $Policy(g)$ is true iff the delegation graph g is considered valid.

For example, the policy for the hotel reseller in Section 3.1 is that any room resold by Reese is at least at the minimum-sell rate for the hotel. In particular, if state g indicates that Reese sold a room r in hotel H for rate v then Reese is willing to sell that hotel room at that rate, that is,

$$Policy0(g) \equiv \forall H, C : Principals;\ r : Room;\ v : \mathbb{N} \bullet \\ (H \stackrel{\mathsf{resell}.r.*}{\Longrightarrow}_g Reese \wedge Reese \stackrel{\mathsf{resell}.r.v}{\Longrightarrow}_g C) \Rightarrow Reese \stackrel{\mathsf{rate}.v}{\Longrightarrow}_g H$$

The reader should recall the encoding of the *minRate* relationship as a delegation $Reese \stackrel{\mathsf{rate}.u}{\Longrightarrow} H$, where $\mathsf{rate}.u \sqsubseteq \mathsf{rate}.v \Leftrightarrow u \geq v$. Given that Reese signed $Reese \stackrel{\mathsf{rate}.40}{\Longrightarrow}_g Harry$ and that $\mathsf{rate}.60 \sqsubseteq \mathsf{rate}.40$ we infer by Rule $D3$ that $Reese \stackrel{\mathsf{rate}.60}{\Longrightarrow}_g Harry$. Thus, the sale $Reese \stackrel{\mathsf{resell}.r.50}{\Longrightarrow}_g Clare$ is valid.

Safe Delegation. A delegation system $System(g)$ safely upholds a delegation policy $Policy(g)$ every state reachable by the system upholds the delegation policy.

$$\forall\, g : STATE \bullet System(g) \Rightarrow Policy(g) \tag{1}$$

In constructing $System(g)$ we assume the use of inference rule $D2$ when carrying out certificate reduction in g.

4.1 A poor implementation of the hotel reseller

Suppose that Reese only enters into new contracts from hotels with which she does not already have a contract (identified as having no minimum rate information). As noted previously, for the sake of simplicity, we do not consider management of the contract permission delegations and the creation of a new contract in state g with hotel H corresponds to the setting of a *minRate* rate v using delegation state transition operation:

```
newRate0(g, H, v){
    if (∄ i : ℕ • Reese ⟹_g^rate.i H) then
       add [Reese ⟹^rate.v H] to g;
    return(g);
}
```

Having decided a minimum resell rate for a hotel, Reese engages state transition operation $\texttt{newRoom0}(g, H, r)$ whenever she receives a resell delegation statement $H \overset{\mathsf{resell}.r.*}{\Longrightarrow} Reese$ for room r in Hotel H.

```
newRoom0(g, H, r){
    if (∃ i : ℕ • Reese ⟹_g^rate.i H) then
       add [H ⟹^resell.r.* Reese] to g;
    return(g);
}
```

Having decided a suitable price v at which to resell hotel H's room r to customer C, *Reese* engages state transition operation $\texttt{bookRoom0}(g, H, C, r, v)$ in state g to issue the booking.

```
bookRoom0(g, H, C, r, v){
    if (Reese ⟹_g^rate.v H ∧ H ⟹_g^resell.r.* Reese) then
       add [Reese ⟹_g^resell.r.v C] to g and issue;
    return(g);
}
```

Reese's rationale in this implementation is that, regardless of however she may decide the selling price, then so long as she only uses transitions `newRate0`, `newRoom0` and `bookRoom0` to update her delegation graph and issue certificates then she will never violate her minimum selling policy.

Proposition 1. If we define $System0(g)$ to be the set of all states reachable from an empty graph by operations `newRate0`, `newRoom0` and `bookRoom0` then Reese can prove that every reachable state in her implementation upholds her delegation policy, that is,

$$\forall\, g : STATE \bullet System0(g) \Rightarrow Policy0(g) \tag{2}$$

That is, $System0$ provides safe delegation under $Policy0$. In constructing $System0(g)$ we assume that Reese uses inference rule $D2$ when carrying out certificate reduction in g.

The proof of Proposition 1 characterizes delegation correctness as a refinement that can be considered to be a safety-style property in the Alpern-Schneider sense alpern:87. However, the subterfuge example in Section 3.2 demonstrates that such a characterization, as a property on a state, is inadequate[1]. This is not surprising: a frequent argument [11, 15, 21] that is that security properties are not safety properties.

5 Delegation as a security property

A principal operating a delegation system may not be certain about the entirety of its delegation state g as a portion of it may have been perturbed by an attacker. The perturbation in the delegation state may come about from an attacker concealing the existence of selected delegation statements and/or injecting new delegation statements. Like a Dolev-Yao attacker [6], we assume that the attacker can only intercept, copy and paste signed statements, and cannot forge cryptographic signatures.

Delegation state equivalence. Let $\approx$ be an invariant relation over delegation states whereby $g \approx_R h$ is interpreted to mean that principal R is as certain of being in state g as it is of being in state h.

An example of a definition of this equivalence relation is:

$$g \approx_R h \equiv \forall\, Q : Principal;\ X : PERM \bullet R \overset{X}{\Longrightarrow}_g Q \Leftrightarrow R \overset{X}{\Longrightarrow}_h Q \tag{3}$$

This reflects an assumption that R cannot rely on fully knowing the delegations of others and, therefore, the only thing that principal R can be sure about is the delegation statements that it has directly made itself.

Alternatively, suppose that the principle R had a reliable connection with a set of principles $\mathcal{S}$. This is interpreted to mean that R *knows* the delegation statements that the principals in $\mathcal{S}$ have or have not made. For example, $\mathcal{S}$ might represent the principals over which a trusted authorization server has jurisdiction and to which R has a reliable connection. In this case, and assuming $R \in \mathcal{S}$, then state equivalence can be generalized to:

$$g \approx_R h \equiv \forall\, P : \mathcal{S};\ Q : Principal;\ X : PERM \bullet P \overset{X}{\Longrightarrow}_g Q \Leftrightarrow P \overset{X}{\Longrightarrow}_h Q$$

[1] At least to the extent that our characterization of delegation correctness can be considered to represent a safety property

Returning to the hotel reseller example, Reese does not have a reliable connection to any of the hotels and, therefore, cannot be sure about the absence or otherwise of statements she has not directly signed herself. For example, if she does not hold statement $Harry \overset{\mathsf{resell.r.*}}{\Longrightarrow}_h Reese$ she cannot be sure that it has not been said by *Harry* and thus we have the state equivalence:

$$\begin{aligned} &[Mike \overset{\mathsf{resell.r.*}}{\Longrightarrow}_g Reese;\ Reese \overset{\mathsf{resell.r.20}}{\Longrightarrow}_g Eve;\\ &\quad Reese \overset{\mathsf{rate.40}}{\Longrightarrow} Harry;\ Reese \overset{\mathsf{rate.20}}{\Longrightarrow} Mike]\\ \approx_{Reese}\ &[Harry \overset{\mathsf{resell.r.*}}{\Longrightarrow}_h Reese;\ Reese \overset{\mathsf{resell.r.20}}{\Longrightarrow}_h Eve;\\ &\quad Reese \overset{\mathsf{rate.40}}{\Longrightarrow} Harry;\ Reese \overset{\mathsf{rate.20}}{\Longrightarrow} Mike] \end{aligned}$$

Note that Reese can always be sure about the minimum selling rate since $Reese \overset{\mathsf{rate}.v}{\Longrightarrow} H$ is a delegation statement that she makes directly and stores locally.

A delegation *Policy* defines a valid delegation state under an assumption that there is certainty about its delegation statements. A principal implementing a delegation *System* cannot make this assumption and the uncertainty must be considered when deciding whether it is safe to issue a delegation certificate. Therefore, an implementation system in some state g should uphold not just $Policy(g)$ but should also uphold the policy for any other uncertain state that is potentially equivalent.

Secure Delegation (Subterfuge Freedom). A delegation *System* used by principal R is resilient to subterfuge when upholding a delegation *Policy* if the delegation policy is upheld by the system for every delegation state h that R can be as certain it is in as each reachable state g.

$$\begin{aligned} &\forall\, g : STATE \bullet System(g) \Rightarrow\\ &\quad (\forall\, h : STATE \bullet g \approx_R h \Rightarrow Policy(h)) \end{aligned}$$

5.1 Subterfuge in the original hotel reseller

Consider again the attack on the hotel reseller in Section 3.2. Suppose that Reese has had a series of legitimate interactions with hotels leading to delegation state f, containing a number of sales/bookings and minimum sell rates. Harry then issues a new resell certificate for room rx, which Mike intercepts and conceals and issues a resell certificate of his own for room rx. Reese accepts this resell certificate from Mike and sells the room to Eve:

$$\begin{aligned} &f' \mathrel{\widehat{=}} \mathtt{newRoom0}(f, Mike, \mathtt{rx})\\ &g \mathrel{\widehat{=}} \mathtt{bookRoom0}(f', Mike, Eve, \mathtt{rx}, 20) \end{aligned}$$

The resulting delegation state of Reese is:

$$g = [Mike \overset{\mathsf{resell.rx.*}}{\Longrightarrow} Reese;\quad Reese \overset{\mathsf{resell.rx.20}}{\Longrightarrow} Eve] \cup f$$

and $Policy0(g)$ holds. However, Reese is not certain that g represents the complete state, and there is an alternative state is h, where $g \approx_{Reese} h$ (based on Equation (3) above), to which the policy should also apply:

$$h \mathrel{\widehat{=}} [Harry \overset{\mathsf{resell.rx.*}}{\Longrightarrow} Reese;\quad Reese \overset{\mathsf{resell.rx.20}}{\Longrightarrow} Eve] \cup f$$

This state h violates the *minRate* policy. In particular, $Reese \overset{\mathsf{rate.20}}{\Longrightarrow}_h Harry$ does not hold and therefore $Policy0(h)$ does not hold. Thus, we have a state g of the system such that

$$System0(g) \land g \approx_{Reese} h \land \neg\, Policy0(h)$$

and, therefore, the system is not subterfuge free.

5.2 Subterfuge-free hotel reseller

Consider the revised reseller delegation mechanism outlined in Section 3.3. The principals use signed permissions along with the revised reduction rule $D2'$. We modify the minimum-sell $Policy0$ defined in Section 4 in order to consider the new permission syntax:

$$\begin{aligned} Policy1(g) \equiv \forall\, H, C, K : Principals;\ r : ROOM;\ v : \mathbb{N} \bullet \\ (H \overset{\{|\mathsf{resell}.r.*|\}_{sK}}{\Longrightarrow_g} Reese \land Reese \overset{\{|\mathsf{resell}.r.v|\}_{sK}}{\Longrightarrow_g} C) \\ \Rightarrow (Reese \overset{\{|\mathsf{rate}.v|\}_{Reese}}{\Longrightarrow_g} H) \end{aligned}$$

As before, the policy is concerned only with enforcing the minimum selling rule, regardless of who may have signed the original permission or how it is implemented. In this way $Policy1$ matches the requirements of the original specification of $Policy0$ in Section 3.3. Indeed, a simple translation of $System0$ to support signed permissions would provide safe delegation under $Policy1$ (safety property), while failing to provide secure delegation under the same policy (security property).

The state transition operations are revised to incorporate a new *implementation* decision that resell permissions must be signed by the delegating hotel. On receipt of a new room resell certificate $H \overset{\{|\mathsf{resell.r.*}|\}_{sK}}{\Longrightarrow} Reese$ from hotel H for room `r` (signed by some K) she engages the operation:

```
newRoom1(g, H, K, r){
    if (H = K ∧ ∃ i : ℕ • Reese ⟹{|rate.i|}_sReese_g H) then
        add [H ⟹{|resell.r.*|}_sK Reese] to g;
    return(g);
}
```

Operation `bookRoom1` is similarly revised:

```
bookRoom1(g, H, C, r, v){
    if (Reese ⟹_g^{|rate.v|}_{sReese} H ∧ H ⟹^{|resell.r.∗|}_{sH} Reese) then
        add [Reese ⟹_g^{|resell.r.v|}_{sH} C] to g and issue;
    return(g);
}
```

and operation `newRate1` is defined in terms of signed rates:

```
newRate1(g, H, v){
    if (∄ i : ℕ • Reese ⟹_g^{|rate.i|}_{sReese} H) then
        add [Reese ⟹^{|rate.v|}_{sReese} H] to g;
    return(g);
}
```

These operations along with revised reduction rule $D2'$ are used to describe the corrected delegation system implementation $System1$.

The definition of the delegation state equivalence invariant $\approx$ is unchanged from Equation (3), since whoever may have signed the permission has no impact over what part of the delegation state might be concealed by an attacker.

The sample subterfuge attack no longer works. If Reese is in a state with $[Mike \overset{\{|\mathsf{resell.r.}*|\}_{sHarry}}{\Longrightarrow_g} Reese]$ then the new `bookRoom` operation will not delegate (on behalf of Reese) permission $\{|\ \mathsf{resell.r.20}\ |\}_{sHarry}$ to *Eve* since the delegator *Mike* of the permission held by Reese is not the signer *Harry* of the permission.

Proposition 2. $System1$ provides secure delegation under $Policy1$:

$$\forall\, g, h : STATE \bullet (System1(g) \land g \approx_{Reese} h) \Rightarrow Policy1(h)$$

6 Related Work

Trust Management systems such as [3, 5, 7, 18] are intended to provide a decentralized approach to constructing and interpreting authorization relationships between principals/domains. Unlike a centralized authorization server-based approach, authorization rules are defined and signed locally by issuing principals. These cryptographic delegation credentials can be distributed in any manner to suit the design of the (Trust Management-based) access control mechanism that mediates according to the policy. In this way, trust management provides a basis for secure decentralized policy based management.

While credential-based policy rules are inherently decentralized, many implicitly assume unique and unambiguous global permissions, effectively originating from some central authority that provides a permission namespace that everyone agrees to consistently use. For example, Keynote [4] relies on the Internet Assigned Number Authority (IANA), RT [18] relies on Application Domain Specification Documents (ADSDs), and X509 relies on the X500 name service, to

ensure that different parties use the right name for resources, conditions, other participants, and so forth. 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.

Delegation subterfuge [10] arises when there is ambiguity in interpreting a permission in its namespace. This can come about from an attacker concealing and/or injecting delegation credentials whereby, as a consequence, a victim may violate the policy that guides its own delegation actions. 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. Subterfuge-freedom can be provided in a role-based distributed authorization language by constraining delegation to permissions that have an associated originating public key [24]. While effective, this approach suffers the challenge of reliably referencing public keys. A SDSI-like naming system can alternatively be used to provide subterfuge-free local permission namespaces [12]. The FRM distributed policy management framework [9] also relies on signed permissions to avoid subterfuge and uses Distinguished Names/X509 certificates to uniquely tie a permission to its namespace.

While subterfuge is concerned with how an attacker can interfere with a target's policy certificates, *probing-free authorization* [2, 3, 14] is concerned with determining what an attacker can infer about hidden policy certificates when making queries. Both subterfuge and probing are effectively concerned with determining whether information flows [15, 19, 21] from one principal to another as a result of using the delegation system. Investigating the relationships between subterfuge and probing, and their relationship to information flow properties in general is an ongoing topic of research [2]. One avenue of interest is whether intransitive information flow [8, 13, 20] might provide an interpretation for *conditional* subterfuge-freedom. In this case, a principal may declare that it is willing to accept accountability for some third-party's permission with the consequence that any perturbation to its delegation state prior to the declaration can be safely ignored.

7 Discussion and Conclusion

Security refinement can be defined to be a system robustly upholding functional requirements (policy) in the presence of threats that may perturb a state that has been arrived at via some trace [11]. If one considers delegation states to be analogous to system traces then the definition of delegation security/subterfuge freedom proposed in this paper is similar, at least in intent, to this definition.

We argue that subterfuge-freedom is not a safety-style property in the conventional sense [1], but that it is a security property [11, 15, 21] that is similar to non-interference [13, 19, 21]. This paper presents an example of an implementation of a policy requirement that appears to be preserved under a safety refinement (Proposition 1), but which is subject to a subterfuge attack. Section 5 demonstrates that the implementation of the policy requirement is not

preserved under the proposed security-style refinement. By using a restricted form of (subterfuge-free) delegation, a revised implementation can be shown to preserve the policy under security refinement.

The primary contribution of this paper is a characterization of subterfuge-freedom as a security-style property. While relatively simple, the SPKI-based delegation model provided a sufficient scheme in which to present the result. While we use the example in Sections 4 and 5 to illustrate the result, we have not provided a formal proof. We are currently exploring a Kripke-based semantics in which to more formally investigate subterfuge properties and their relationship to safety and security properties in general. Future research will also consider analysis of secure delegation in more complex delegation schemes and how that analysis might be automated.

Acknowledgement

This research has been supported in part by Science Foundation Ireland grant 08/SRC/11403. The author would like to thank the anonymous reviewers for their useful feedback.

References

1. Alpern, B., Schneider, F.: Recognizing safety and liveness. Distributed Computing 2, 181–126 (1987)
2. Becker, M.Y.: Information flow in trust management systems. Journal of Computer Security 20(6), 677–708 (2012)
3. Becker, M.Y., Fournet, C., Gordon, A.D.: Secpal: Design and semantics of a decentralized authorization language. Journal of Computer Security 18(4), 619–665 (2010)
4. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: The KeyNote Trust-Management System Version 2. RFC 2704 (Informational) (Sep 1999), http://www.ietf.org/rfc/rfc2704.txt
5. 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 (1998)
6. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29(2), 198–208 (1983)
7. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory. RFC 2693 (Experimental) (Sep 1999), http://www.ietf.org/rfc/rfc2693.txt
8. Engelhardt, K., van der Meyden, R., Zhang, C.: Intransitive noninterference in nondeterministic systems. In: ACM Conference on Computer and Communications Security. pp. 869–880 (2012)
9. 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)
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.: A non-functional approach to system integrity. IEEE Journal on Selected Areas in Communications 21(1) (Jan 2003)
12. Foley, S., Abdi, S.: Avoiding delegation subterfuge using linked local permission names. In: Proceedings of 8th International Workshop on Formal Aspects of Security and Trust (FAST2011). LNCS, Springer (2011)
13. Goguen, J., Meseguer, J.: Unwinding and inference control. In: IEEE Symposium on Security and Privacy. pp. 75–87 (1984)
14. Gurevich, Y., Neeman, I.: DKAL: Distributed-knowledge authorization language. In: CSF (2008)
15. Jacob, J.: Basic theorems about security. Journal of Computer Security 1, 385–411 (1992)
16. Jr., D.G.: Power. law. IEEE Security & Privacy 10(1) (Jan 2012)
17. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in distributed systems: Theory and pratice. ACM Trans. Computer Systems 10(4), 265–310 (1992)
18. Li, N., Mitchell, J.C.: RT: A role-based trust-management framework. In: The Third DARPA Information Survivability Conference and Exposition (DISCEX III). pp. 201–212. IEEE Computer Society Press, Los Alamitos, California, Washington, D.C. (April 2003)
19. Mantel, H.: Information flow and noninterference. In: Encyclopedia of Cryptography and Security (2nd Ed.), pp. 605–607. Springer (2011)
20. Roscoe, A.W., Goldsmith, M.H.: What is intransitive noninterference? In: CSFW. pp. 228–238 (1999)
21. Ryan, P.: Mathematical models of computer security. In: Focardi, R., Gorrieri, R. (eds.) Foundations of Security Analysis and Design, Lecture Notes in Computer Science, vol. 2171, pp. 1–62. Springer Berlin / Heidelberg (2001)
22. Zeller, T.: Purloined domain name is an unsolved mystery. In: New York Times (Jan 18 2005)
23. 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)
24. Zhou, H., Foley, S.: A framework for establishing decentralized secure coalitions. In: Proceedings of IEEE Computer Security Foundations Workshop. IEEE CS Press (2006)

A Proof of Propositon 2

*System*1 provides secure delegation under *Policy*1:

$$\forall\, g : STATE \bullet System1(g) \Rightarrow (\forall\, h : STATE \bullet g \approx_R h \Rightarrow Policy1(h))$$

Let $speakers(g)$ be the set of principals who have signed/made delegation statements in g.

Base case. Given an initial delegation state $init = []$ then it follows that $System1(init)$. Consider any equivalent state h where $init \approx_R h$, then by definition we have $R \notin speakers(h)$. Thus, there is no delegation of the form $R \overset{\{|\mathsf{resell}.r.v|\}_{sK}}{\Longrightarrow_h} C$ in state h and, therefore, $policy1(h)$ holds. Therefore, the base case holds.

Inductive step. Assume that *System*1 is in a state g where $(\forall\, h : STATE \bullet g \approx_R h \Rightarrow Policy1(h))$. The inductive goal is to prove that for all $h' : STATE$ then $op(g) \approx_R h' \Rightarrow Policy1(h'))$, for each system transition operation op.

Consider each system transition $g' = op(g)$. If the operation precondition is satisfied then $g = g'$ and the inductive goal trivially holds. Consider g' when the operation precondition holds.

- $g' = \texttt{newRate1}(g, H, v)$: given precondition $(\nexists\, i : \mathbb{N} \bullet R \overset{\{\!|\mathsf{rate}.i|\!\}_R}{\Longrightarrow_g} H)$ then $g' = g \cup [R \overset{\{\!|\mathsf{rate}.v|\!\}_R}{\Longrightarrow} H]$. Consider a state h' where $g' \approx_R h'$. It follows from the definition of equivalence that $[R \overset{\{\!|\mathsf{rate}.v|\!\}_R}{\Longrightarrow} H] \in h'$, and thus if $h = h'/[R \overset{\{\!|\mathsf{rate}.v|\!\}_R}{\Longrightarrow} H]$ then $g \approx_R h$. Therefore, given the inductive hypothesis, $Policy1(h)$ holds. Suppose $Policy1(h \cup [R \overset{\{\!|\mathsf{rate}.v|\!\}_R}{\Longrightarrow} H])$ is false; for this to be the case, the $Policy1(h')$ antecedent $(H \overset{\{\!|\mathsf{resell}.r.*|\!\}_{sK}}{\Longrightarrow_{h'}} R \wedge R \overset{\{\!|\mathsf{resell}.r.u|\!\}_{sK}}{\Longrightarrow_{h'}} C)$ has a negative conclusion $[\neg\ R \overset{\{\!|\mathsf{rate}.u|\!\}_R}{\Longrightarrow_{h'}} H]$. However, if $R \overset{\{\!|\mathsf{resell}.r.u|\!\}_{sK}}{\Longrightarrow_{h}} C$ holds in state h then, by $g \approx_R h$, $R \overset{\{\!|\mathsf{resell}.r.u|\!\}_{sK}}{\Longrightarrow_{g}} C$ must also hold in state g. However, the system cannot be in a state g where a resell certificate is issued without a corresponding rate statement. Therefore, this antecedent is false and thus, given $g' \approx_R h'$ then $Policy1(h')$ holds.
- $g' = \texttt{newRoom1}(g, H, K, r)$. For precondition $(\exists\, i : \mathbb{N} \bullet R \overset{\{\!|\mathsf{rate}.i|\!\}_{sR}}{\Longrightarrow_g} H)$ then $g' = g \cup [H \overset{\{\!|\mathsf{resell}.\mathsf{r}.*|\!\}_{sH}}{\Longrightarrow} R]$. By the definition of equivalence, we have $g \approx_R g'$, and therefore, by transitivity of equivalence, for any state h' such that $g' \approx_R h'$ holds, then the inductive hypothesis gives $g' \approx_R h' \Rightarrow Policy1(h')$.
- $g' = \texttt{bookRoom1}(g, H, C, r, v)$. Given precondition $(R \overset{\{\!|\mathsf{rate}.v|\!\}_{sR}}{\Longrightarrow_g} H \wedge H \overset{\{\!|\mathsf{resell}.\mathsf{r}.*|\!\}_{sH}}{\Longrightarrow} R)$ then $g' = g \cup [R \overset{\{\!|\mathsf{resell}.r.v|\!\}_{sH}}{\Longrightarrow_g} C]$. Consider a state h' where $g' \approx_R h'$. It follows from the definition of equivalence that $[R \overset{\{\!|\mathsf{resell}.r.v|\!\}_{sH}}{\Longrightarrow_g} C] \in h'$, and thus if $h = h'/[R \overset{\{\!|\mathsf{resell}.r.v|\!\}_{sH}}{\Longrightarrow_g} C]$ then $g \approx_R h$. Therefore, given the inductive hypothesis, $Policy1(h)$ holds. Given $g' \approx_R h'$ then a rate statement in g' appears in h' and if the precondition of `newRoom1` holds in g' then it also holds in h'. Therefore, if the resell action is valid in g' it will also be valid in h' and $Policy1(h')$ holds.