Postulates for Revocation Schemes Technical Report

Marcos Cramer, Giovanni Casini

University of Luxembourg

Abstract. In access control frameworks with the possibility of delegating permissions and administrative rights, delegation chains can form. There are different ways to treat these delegation chains when revoking rights, which give rise to different revocation schemes. Hagström et al. [11] proposed a framework for classifying revocation schemes, in which the different revocation schemes are defined graph-theoretically. At the outset, we identify multiple problems with Hagström et al.'s definitions of the revocation schemes, which can pose security risks. This paper is centered around the question how one can systematically ensure that improved definitions of the revocation schemes do not lead to similar problems. For this we propose to apply the axiomatic method originating in social choice theory to revocation schemes. Our use of the axiomatic method resembles its use in belief revision theory. This means that we define postulates that describe the desirable behaviour of revocation schemes, study which existing revocation frameworks satisfy which postulates, and show how all defined postulates can be satisfied by defining the revocation schemes in a novel way.

1 Introduction

In ownership-based frameworks for access control, it is common to allow principals (users or processes) to grant both permissions and administrative rights to other principals in the system. Often it is desirable to grant a principal the right to further grant permissions and administrative rights to other principals. This may lead to delegation chains starting at a *source of authority* (the owner of a resource) and passing on certain permissions to other principals [12, 14, 5, 15].

Furthermore, such frameworks commonly allow a principal to revoke a permission that she granted to another principal [11, 16, 5, 2]. Depending on the reasons for the revocation, different ways to treat the delegation chain can be desirable [11, 1, 7]. For example, if one is revoking a permission given to an employee because he is moving to another position in the company, it makes sense to keep in place the permissions she previously granted; but if one is revoking a permission from a user who has abused his rights and is hence distrusted, it makes sense to delete the permissions she previously issued. Any algorithm that determines which permissions to keep intact and which ones to delete when revoking a permission is called a *revocation scheme*. Revocation schemes are usually defined in a graph-theoretical way.

Hagström et al. [11] have presented a framework for classifying possible revocation schemes along three different dimensions: the extent of the revocation to other grantees (propagation), the effect on other grants to the same grantee (dominance), and the permanence of the negation of rights (resilience). This classification was based on revocation schemes implemented in database management systems [10, 9, 4, 3]. The framework's design decisions are carried over from these database management systems and are often not fully motivated.

We identify a number of problems with Hagström et al.'s framework and the definitions of the revocation schemes included in the framework. Some of these problems pose security risks. In order to avoid that an improved framework turns out to have similar undesirable properties as those we identified in Hagström et al.'s framework, we propose to formally study the merits and demerits of various definitions of revocation schemes using the *axiomatic method*. This methodology originates in social choice theory, and is used in a way akin to ours in belief revision theory (see [13] for an overview of this methodology in belief revision and its connections to social choice theory). We will state formal properties, called *postulates*, which formalize our intuitions about the desired behaviour of the revocation schemes. We will study which postulates are satisfied by the existing revocation frameworks, and show how all of them can be satisfied by defining the revocation schemes in a novel way.

The idea to use this methodology in the study of delegation revocation was first put forward in Cramer et al. [7] (the main author of which is also the main author of the current paper). The main goal of Cramer et al. [7] was to state postulates that fully characterize all the revocation schemes. This could only be achieved by introducing a dedicated logic, called *Trust Delegation Logic*, that allows to formalize the reasons that principals have for delegating and revoking. However, this logic is highly complex and has many non-trivial design choices, so that this approach leaves open the question whether the logic really correctly formalizes our intuitions about the desired behaviour of revocation schemes. In this paper, we instead define simpler postulates, whose meaning can be understood more readily. This way of applying the axiomatic method is more in line with standard applications of this methodology in social choice theory and belief revision. We show that one of the simple postulates that we introduce in this paper is not satisfied by the framework that was introduced in Cramer et al. [7]. This means that the approach of the present paper, based on simpler postulates, can help to detect problems that the approach from [7] cannot detect.

The rest of the paper is structured as follows: In Section 2 we discuss the work of Hagström et al. [11] that the present paper is heavily based on. After specifying some formal preliminaries in Section 3.1, we motivate and define four postulates for revocation schemes in Section 3.2, and show which of these postulates are satisfied by which existing delegation-revocation frameworks in Section 3.3. Sections 4-6 are dedicated to defining a delegation-revocation framework that satisfies all the defined postulates. This is done in a stepwise way: First we define in Section 4 the framework *Dom*, which only covers the distinction made in the dominance dimension. Section 5 extends this framework to *DR*, which also covers the resilience dimension, which is

further extended in Section 6 to the framework *DPR* that covers all three dimensions. In Section 7, we conclude the paper and discuss some possible further research.

2 Related work

The only existing work on delegation revocation that takes the same methodological approach as the present paper is Cramer et al. [7]. The relation between the present paper and [7] has already been sketched in the Introduction, and will be discussed further throughout the rest of the paper. In the present section we discuss the work of Hagström et al. [11] that both the present paper and [7] are heavily based on, and explain a terminological issue.

2.1 Hagstöm et al.'s framework

Hagström et al. [11] have introduced three dimensions according to which revocation schemes can be classified: *dominance*, *propagation* and *resilience*.

Dominance. This dimension deals with the case when a principal losing a permission in a revocation still has permissions from other grantors. If these other grantors' revocation rights are dependent on the revoker, the revoker can dominate over these grantors and revoke the permissions from them. This is a *strong* revocation. The revoker can also choose to make a *weak* revocation, where permissions from other grantors to a principal losing a permission are kept.

Propagation. The decision of a principal i to revoke an authorization previously granted to a principal j may either affect only the direct recipient j or propagate and affect all the other users in turn authorized by j. In the first case, we say that the revocation is *local*, in the second case that it is *global*.

Resilience. This dimension distinguishes revocation by removal (deletion) of positive authorizations from revocation by issuing a negative authorization which just inactivates positive authorizations. In the first case another principal may grant a similar authorization to the one that had been revoked, so the effect of the revocation does not persist in time. In the second case a negative authorization will overrule any (new) positive permission given to the same principal, so its effect will remain until the negative permission is revoked. We call a revocation of the first kind a *delete* or *non-resilient* revocation, and a revocation of the second kind a *negative* or *resilient* revocation.

Since there are two possible choices along each dimension, Hagström et al.'s framework allows for eight different revocation schemes. The behaviour of the revocation schemes is defined differently depending on whether precedence is given to positive or negative authorizations. Cramer et al. [7] have argued for integrating this precedence into the dominance dimension, thereby replacing the binary distinction along the dominance dimension by a ternary distinction between *strong*, *predecessor-takes-precedence* (*p-t-p*) and *weak* revocations. Here *p-t-p* has the meaning that Hagström et al. give to *strong*, while a *strong* revocation dominates over all other grantors' authorizations, no matter whether the principal targeted by the revocation is dependent on the principal performing the revocation or not. This

design decision and terminological decision are motivated in Section 3.1 of Cramer et al. [7].

2.2 Problems with Hagström et al.'s framework

In this section we analyze some problems with the revocation framework by Hagström et al. [11], and informally sketch how we propose to solve them.[1] As many of these problems amount to a principal having access right in a situation where the intended meaning of the used revocation scheme implies that the principal should not have access right, these problems can pose security risks.

(1) In Hagström et al.'s framework, the relative timing of a granting a permission and a Strong Global Delete revocation influences the effect of the revocation in an undesirable way. Let us illustrate this problem with an example.

Example 1. User A issues an authorization to users B and C. B plans to grant this authorization to C. At the same time A plans to perform a Strong Global Delete revocation of B's rights. Depending on which user performs the planned action first, the outcome will be different in Hagström et al.'s framework. If the Strong Global Delete is performed first, user C will be unaffected. But if B first delegates to C, then user C will also lose his access right as a consequence of the Strong Global Delete from A to B.

One way to explain why this behaviour is problematic is to note that if the revocation was a Weak Global Delete instead of a Strong Global Delete, C would be unaffected even if B first delegated to C. But the difference between a Strong Global Delete and a Weak Global Delete is supposed to be only about the dominance of the revocation, i.e. about what happens when others have delegated to B. But as no one else has delegated to B, there should be no difference between the two revocations.

Another way to explain why this behaviour of the Strong Global Delete is problematic is to note that whether B attempts to delegate to A shortly before or shortly after the Strong Global Delete should not make a difference. The timing of a delegation with respect to a Strong Global Delete should only matter if it is a delegation of a right to B, as the revocation is non-resilient. But since the revocation is global, the timing of a delegation performed by B should not matter.

(2) A similar problem is faced by the Strong Local Negative revocations in Hagström et al.'s framework:

Example 2. The SOA delegates a right to user A, who delegates it further to user B, who delegates it further to user C. Now A plans to delete the authorization she

[1] In Cramer et al. [7] five problems with Hagström et al.'s framework are discussed. As problems (4) and (5) from [7] are also relevant to the present paper, we have taken them over into the present paper, where they are listed as problems number (3) and (4) respectively. Problem (1) below is based on problem (1) from [7], but the explanation of the problem has been significantly reworked and extended. Problem (2) below has not been presented in print before. Two further problems with Hagström et al.'s framework not relevant to the present paper were presented in sections III.C and III.D of Cramer et al. [6].

has issued to user B, and at the same time, the SOA plans to perform a Strong Local Negative revocation of B's rights. Depending on which user performs the planned action first, the outcome will be different in Hagström et al.'s framework. If the Strong Local Negative is performed first, C will conserve his access right even after the deletion of the authorization from A to B. But if A deletes the authorization to B first, then user C will lose his access right.

One way to explain why this behaviour of the Strong Local Resilient is problematic is to note that the deletion of the authorization from A to B is a non-resilient revocation. Hagström et al. say about non-resilient revocations that after the revocation, "no trace remains of the fact that the authorization has been granted and then revoked". But in Example 2, there does remain a trace of the authorization from A to B, namely the fact that C has access right (which materializes through an auxiliary authorization from the SOA to C, which is created only because there exists an authorization from A to B at the moment of the local revocation).

Another way to explain why this behaviour of the Strong Local Resilient is problematic is to note that while for a Strong Local Resilient revocation of B's rights the timing of delegations performed by B with respect to the revocation is relevant (as it is a local revocation), the timing of other actions that affect B with respect to the revocation should not make a difference, as the revocation is strong and resilient.

(3) Hagström et al. motivate the distinction between delete and negative revocations mainly through the notion of resilience as defined in Section 2.1. However, this definition renders the notion of a weak resilient revocation contradictory, since a weak revocation does not affect authorizations issued by others than the revoker. (Hagström et al. motivate the usage of weak negatives by pointing out that they are useful for temporary revocations, but as discussed in Cramer et al. [7], a better way to make temporary revocations possible is to not delete the forward chain in a delete revocation.)

Furthermore, p-t-p and strong deletes would have undesirable effects, as illustrated by the following example:

Example 3. User A issues an authorization to user B, and gives user C the right to perform strong revocations. User C performs a Strong Global Delete on B, removing without traces the authorization provided to B by A. Later A realizes that C cannot be trusted to perform strong revocations, and takes away B's right to do so through a Strong Global Delete revocation. Even though C can no longer perform strong revocations, the effect of his strong delete persist: B does not have the right originally issued to him by A until someone issues a new authorization to him.

Hence we do not have a p-t-p or strong delete revocation in our framework, but instead have the distinction between a resilient and a non-resilient negative for p-t-p and strong revocations. To conclude, if the dominance of a revocation is p-t-p or strong, there are two options along the resilience dimension, non-resilient and resilient, both of which are defined through negative authorizations. But if the dominance is weak, the value of the resilience dimension has to be "non-resilient". A weak

non-resilient revocation is defined through the deletion of a positive authorization, and is therefore also called a "weak delete".

(4) Hagström et al. do not allow negative authorizations to be inactivated. The reason they give is that they "do not want a revocation to result in a subject having more permissions than before the revocation". However, the deletion of negative authorizations is allowed, even though it may have the same effect. We do allow negative authorizations to be inactivated, but the only kind of revocation that can result in a subject having more permissions than before is a revocation of someone's right to perform strong revocations, and in this case this is a desirable property.

2.3 Revocations and denials

A revocation of a principal's rights removes rights that the principal already has. A denial of rights on the other hand can be issued even when the principal does not yet have the concerning rights, and has the effect that other principals will no longer be able to effectively grant rights to the affected principal.

Negative authorizations can function as a form of denial. When, for example, j does not yet have the rights in question and i issues a negative authorization for those rights to j, this negative authorization functions like a denial rather than like a revocation. The work in this paper applies to negative authorizations independently of whether they are used to revoke existing rights or deny rights. We will for the rest of this paper only use the term "revocation" and not "denial", in order to be consistent with the terminology used in the papers that we extensively refer to.

3 Postulates for Delegation and Revocation

In this section we formally define four postulates for delegation and revocation that formalize desirable properties of a delegation-revocation framework. The postulates are justified on the basis of the intended meaning of the possible values along the three revocation dimensions. Our justification of the postulates is partially based on the discussion of the problems considered in Section 2.2.

From a formal point of view, the role of a delegation-revocation framework is to specify which users will have access given that certain delegations and revocations have been performed in a certain temporal order. In order to make this more precise, we first introduce some notation.

3.1 Preliminaries

Let **S** be the set of principals (subjects) in the system, let **O** be the set of objects in the system and let **A** be the set of access types. For every object $o \in \mathbf{O}$, there is a *source of authority* (SOA), i.e. the manager of object o.

For any $\alpha \in \mathbf{A}$ and $o \in \mathbf{O}$, the SOA of o can grant the right to access α on object o to other principals in the system. Secondly, the SOA can delegate this granting right further. Thirdly, the SOA can grant the right to perform strong revocations and to delegate this right further. Accordingly we have three *permissions*: *access*

right (A), *delegation right* (D) and *strong revocation right* (S). We assume that delegation right implies access right. The set $\{A, D, S\}$ of permissions is denoted by **P**.

There is no interaction between the rights of principals concerning different access-object pairs (α, o). For this reason, we can consider α and o to be fixed for the rest of the paper, and no longer explicitly mention them. We use W, P, S, L, G, N and R as abbreviations for *weak*, *p-t-p (predecessor-takes-precedence)*, *strong*, *local*, *global*, *non-resilient*, *resilient* and *delete* respectively. We define Σ^* to be the set $\{\mathsf{W}, \mathsf{P}, \mathsf{S}\} \times \{\mathsf{L}, \mathsf{G}\} \times \{\mathsf{N}, \mathsf{R}\}$, i.e. the set of all conceivable combinations of revocation dimension values (assuming that there are three possible values for the dominance dimension as explained at the end of Section 2.1).

Let i and j be two principals, and let π be a permission π. We write grant(i, j, π) for i's action of granting permission π to j. Given $(\mathfrak{d}, \mathfrak{p}, \mathfrak{r}) \in \Sigma^*$, we write revoke$(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$ for i's action of revoking permission π from j with dominance $\mathfrak{d}$, propagation $\mathfrak{p}$ and resilience $\mathfrak{r}$. We say that the actions grant(i, j, π) and revoke$(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$ are *performed by* the principal i and *targeted at* the principal j.

Since delegation right implies access right, an action grant(i, j, D) can only be performed in combination with the action grant(i, j, A). By taking the contrapositive, the connection is reversed for revocations: The action revoke$(i, j, A, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$ can only be performed in combination with the action revoke$(i, j, D, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$.

We define a *delegation-revocation profile* to be a sequence of delegation and revocation actions such that directly before any action of the form grant(i, j, D) there is an action of the form grant(i, j, A), and directly before any action of the form revoke$(i, j, A, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$ there is an action of the form revoke$(i, j, D, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$. For example, the profile

$$\langle \mathsf{grant}(\mathsf{A},\mathsf{B}, A), \mathsf{grant}(\mathsf{A},\mathsf{C}, S), \mathsf{revoke}(\mathsf{C},\mathsf{B}, \alpha, \mathsf{S},\mathsf{G},\mathsf{N}), \mathsf{revoke}(\mathsf{A},\mathsf{C}, S, \mathsf{S},\mathsf{G},\mathsf{N}) \rangle$$

formally expresses the delegation and revocation actions that were taken in Example 3 in Section 2.2 as well as there temporal ordering. Given two delegation-revocation profiles Π_1 and Π_2, we write $\Pi_1 \oplus \Pi_2$ for the profile resulting from concatenating the sequence Π_1 with the sequence Π_2.

Let $\Sigma \subseteq \Sigma^*$ be some set of revocation dimension combinations. We say that a profile Π is *over* Σ if for every revocation action revoke$(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathfrak{r})$ in Π, $\{\mathfrak{d}, \mathfrak{p}, \mathfrak{r}\} \in \Sigma$. A *delegation-revocation framework over* Σ is a function F that takes as input a delegation-revocation profile Π over Σ, and outputs a set $F(\Pi)$ of principals that encodes the information which principals have access and which ones do not have access if delegation and revocation actions have been performed as specified by Π.

For example, the Hagström et al. [11] define two delegation-revocation frameworks: The one that describes the behaviour of the revocations when positive revocations have precedence is a delegation-revocation framework over $\{\mathsf{W}, \mathsf{P}\} \times \{\mathsf{L}, \mathsf{G}\} \times \{\mathsf{N}, \mathsf{R}\}$ (even though they use the terms "strong", "delete" and "negative" instead of "p-t-p", "non-resilient" and "resilient"), while the framework that describes the behaviour of the revocations when negative authorizations have precedence is in

place is a delegation-revocation framework over $\{S\} \times \{L, G\} \times \{\text{N, R}\}$. Below we call these two delegation-revocation frameworks H^+ and H^- respectively. In Cramer et al. [7] a delegation-revocation framework (called C below) over the set $\Sigma' := (\{\text{W}\} \times \{\text{L, G}\} \times \{\text{N}\}) \cup (\{\text{P, S}\} \times \{\text{L, G}\} \times \{\text{N,R}\})$ is defined, whereas in Cramer et al. [6], the restriction of this framework over $\{(\text{P,G,R})\}$ is defined (i.e. the only revocation considered is P-t-p Global Resilient). The set Σ' is also the most extensive set over which we define a delegation-revocation framework in this paper. The reason for not defining a delegation-revocation framework over the full set Σ^* of conceivable revocation dimension combinations is that weak resilient revocations do not make sense, as discussed under point (3) in Section 2.2.

Delegation-revocation frameworks are usually defined with the help of a *delegation-revocation graph*, i.e. a graph whose nodes are principals and whose labelled edges encode relevant information about the granting and revocation actions taken by principals. The delegation-revocation framework specifies how the graph is to be modified given a certain action, and how to determine who has access given a certain graph.

One might be tempted to think that delegation-revocation profiles are practically the same thing as delegation-revocation graphs. However, the distinction between them is central to our methodology. It is a distinction akin to the distinction between the syntax and the semantics of a formal logical language. The delegation-revocation profiles play the role of the syntax: They encode the observable granting and revocation action of the principals, independently of how we decide to interpret these actions. One could be tempted to think that the semantics of a delegation-revocation profile should just be the set of principals that get access based on that profile. But that information is not enough as a semantic structure, because two profiles that lead to the same principals having access can nevertheless behave differently: Further actions that are added to one of these two profiles can lead to different access rights depending on which profile the actions were added to. The delegation-revocation graphs give us the additionaly structural information that is needed to semantically distinguish profiles that behave differently over time: They allow us to interpret what a sequence of actions means, both in the sense of allowing us to determine who has access after that sequence of actions, as well as allowing us to determine who will have access if certain further actions are taken.

3.2 The four postulates

Given that any function from the set of delegation-revocation profiles to the powerset of the set of principals counts as a delgation-revocation framework, there are many different ways of defining delegation-revocation frameworks. However, we are not really interested in arbitrary delegation-revocation frameworks, but only in those frameworks that behave in a way that meets our expectations of what it means to grant a permission and to revoke a permission with a certain combination of revocation dimension values. The goal of the axiomatic approach that we take is to formalize some of these expectations so that we can study which graph-theoretic definitions of delegation-revocation frameworks meet which expectations. Following

the belief revision literature, whose methodological approach we follow, we call the formalized formulation of these expectations *postulates*.

We should stress that in this paper we are not aiming at formalizing all our expectations about what granting and the revocation dimensions mean, nor to specify a set of postulates that uniquely determines a delegation-revocation framework. The latter aim was achieved by Cramer et al. [7], but at the expense of specifying a very complicated postulate based on a dedicated logic (Trust Delegation Logic) with many non-trivial design choices. The present paper complements that approach by formulating simpler postulates, whose meaning can be understood more readily.

The first postulate that we consider is called *Locality*, as it formalizes a central desirable feature of local revocation schemes: a local revocation should only affect the principal at which it is targeted. Formally, the fact that the delegation-revocation framework F satisfies Locality can be expressed as follows:

> **Locality.** Let $\Sigma \subseteq \Sigma^*$ be a set of revocation dimension combinations. Then for any delegation-revocation profile Π over Σ and any $i, j \in \mathbf{S}$, $\pi \in \mathbf{P}$, $\mathfrak{d} \in \{\mathsf{W},\mathsf{P},\mathsf{S}\}$ and $\mathfrak{r} \in \{\mathsf{N},\mathsf{R}\}$ such that $(\mathfrak{d}, \mathsf{L}, \mathfrak{r}) \in \Sigma$,
>
> $$F(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle) \cup \{j\} = F(\Pi) \cup \{j\}.$$

The second postulate that we consider is called *Resilience Indifference*, as it formalizes the idea that when a revocation is at then end of a delegation-revocation profile, it does not make a difference whether it is a resilient or a non-resilient revocation. Formally:

> **Resilience Indifference.** Let $\Sigma \subseteq \Sigma^*$ be a set of revocation dimension combinations. Then for any delegation-revocation profile Π over Σ and any $i, j \in \mathbf{S}$, $\pi \in \mathbf{P}$, $\mathfrak{d} \in \{\mathsf{W},\mathsf{P},\mathsf{S}\}$ and $\mathfrak{p} \in \{\mathsf{L},\mathsf{G}\}$ such that $(\mathfrak{d}, \mathfrak{p}, \mathsf{N}) \in \Sigma$ and $(\mathfrak{d}, \mathfrak{p}, \mathsf{R}) \in \Sigma$,
>
> $$F(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathsf{N})\rangle) = F(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathsf{R})\rangle).$$

The motivation for this postulate is that the intended difference between a resilient and a non-resilient revocation is that the non-resilient revocation can be overridden by a later granting action, whereas a resilient revocation cannot be overridden in this way. As this difference only plays a role when there is some granting action after the revocation, it cannot make a difference when the revocation is the last action that has been performed.

The third postulate is called *Access from Revocation*, and formalizes the idea that the only revocation that can lead to any principal having more access than before the revocation is a revocation of permission S (the right to perform a strong revocation). Formally:

> **Access from Revocation.** Let $\Sigma \subseteq \Sigma^*$ be a set of revocation dimension combinations, and let Π be a delegation-revocation profile over Σ. Let a be a revocation action concerning a permission other than S. Then
>
> $$F(\Pi \oplus \langle a \rangle) \subseteq F(\Pi).$$

As explained in the discussion of problem (4) in Section 2.2, this postulate is a weakening of an idea of Hagström et al., who "do not want a revocation to result in a subject having more permissions than before the revocation", but who nevertheless define delete revocations that do not satisfy this property.

The fourth and last postulate that we consider is called *Timing Indifference*, as it formalizes ideas about the conditions under which the relative timing of two actions does not make a difference. The explanations of problems (1) and (2) in Section 2.2 were partially based on considerations of timing indifference. Those explanations suggest the following characterization of timing indifference between a revocation and another action:

- For a global non-resilient revocation targeted at principal l, the temporal ordering between this revocation and any action targeted at a principal other than l does not matter.
- For a local resilient revocation targeted at principal l, the temporal ordering between this revocation and any action performed by a principal other than l does not matter.
- For a global resilient revocation, the temporal ordering between the revocation and another action does not matter.
- For a local non-resilient revocation targeted at principal l, the temporal oredering between this revocation and any action that performed by and targeted at a principal other than l does not matter.

If both actions considered for timing indifference are revocations, the above conditions need to be satisfied in both directions. If both actions are granting actions, the timing between them should never make a difference.

The above criteria for timing indifference can be formalized in a single postulate as follows:

> **Timing Indifference.** Let $\Sigma \subseteq \Sigma^*$ be a set of revocation dimension combinations, and let Π_1 and Π_2 be delegation-revocation profiles over Σ. Suppose that a_1 is a granting or revocation action performed by i and targeted at j, and that a_2 is a granting or revocation action performed by k and targeted at l such that the following properties are satisfied:
> 1. a_1 is either a granting action or a global revocation action, or $k \neq j$.
> 2. a_1 is either a granting action or a resilient revocation action, or $l \neq j$.
> 3. a_2 is either a granting action or a global revocation action, or $i \neq l$.
> 4. a_2 is either a granting action or a resilient revocation action, or $j \neq l$.
>
> Then
> $$F(\Pi_1 \oplus \langle a_1, a_2 \rangle \oplus \Pi_2) = F(\Pi_1 \oplus \langle a_2, a_1 \rangle \oplus \Pi_2).$$

3.3 The postulates applied to existing frameworks

Both H^+ and H^- (the two delegation-revocation frameworks by Hagström et al. depending on the precedence of positive or negative authorizations) as well as C (the delegation-revocation framework by Cramer et al. [7]) satisfy the Locality postulate, because in a local revocation these three frameworks add auxiliary authorizations

from the principal performing to the revocation to any principal not targeted by the revocation that would otherwise be affected by the revocation.

While H^+ and C satisfy Resilience Indifference, H^- does not satisfy it, due to problem (4) from Section 2.2. Suppose the SOA gives A access right and gives B the right to issue negative authorizations (i.e. to perform strong revocations), and B uses this right to revoke A's access right through a Strong Global Negative revocation. Suppose further that after this the SOA revokes the right to issue negative authorizations from B. If this revocation is a delete revocation (i.e. non-resilient), it will according to Hagström et al. also delete the negative authorization from B to A, thus giving back access to A. But if this revocation is a negative authorization (i.e. resilient), it will not inactivate the authorization from B to A due to Hagström et al.'s principle that negative authorizations cannot get inactivated, so A will not get back access right. So A's access right depends on whether the final action is a resilient or non-resilient revocation, thus contradicting Resilience Indifference. Note that modifying H^- by allowing negative authorizations to get inactivated will ensure satisfaction of Resilience Indifference.

H^+ and H^- fail to satisfy Timing Indifference in multiple ways. For example, problem (1) from Section 2.2 shows how they fail to satisfy it for a Strong Global Delete (i.e. Non-Resilient) revocation, and problem (2) shows how they fail to satisfy it for a Strong Local Negative (i.e. Resilient) revocation. C also does not satisfy Timing Indifference, because it behaves in the same way as H^+ and H^- on the example from problem (2) in Section 2.2. But unlike in H^+ and H^-, the global revocations in C do satisfy Timing Indifference. More formally, the restriction of C to a delegation-revocation graph over $(\{\mathsf{W}\} \times \{\mathsf{G}\} \times \{\mathsf{N}\}) \cup (\{\mathsf{P}, \mathsf{S}\} \times \{\mathsf{G}\} \times \{\mathsf{N},\mathsf{R}\})$ satisfies Timing Indifference.

To conclude, H^- only satisfies two of the four postulates that we have defined, while H^+ and C satisfy the first three of them. C only fails Timing Indifference in the case of local revocations. This suggests that it might be possible to define a delegation-revocation framework that satisfies all four postulates by modifying the treatment of local revocations in C. This is what we will do by defining the framework *DPR* in Section 6. To build up to that task, we first define a basic delegation-revocation framework over $\{(\mathsf{S},\mathsf{G},\mathsf{R}), (\mathsf{P},\mathsf{G},\mathsf{R}), (\mathsf{W},\mathsf{G},\mathsf{N})\}$ called *Dom*, which we then extend stepwise.

4 The basic framework Dom

In this section we define the basic delegation-revocation framework *Dom* that distinguishess three revocations based on the dominance dimension. *Dom* will be extended to delegation-revocation frameworks incorporating first the Resilience dimension (Section 5), and then the Propagation dimension (Section 6).

The three revocations in *Dom* are *Strong Global Resilient* (SGR), *P-t-p Global Resilient* (SGR) and *Weak Global Delete* (WGD). In other words, the value of the propagation dimension is fixed to *Global*, and the value of the resilience dimension is fixed to *Resilient* when possible (as explained in Section 2.2, it does not make sense to have weak resilient revocations). So formally *Dom* is a delegation-revocation

framework over the set $\{(\mathsf{S},\mathsf{G},\mathsf{R}), (\mathsf{P},\mathsf{G},\mathsf{R}), (\mathsf{W},\mathsf{G},\mathsf{N})\}$ of revocation dimension combinations.

As the delegation-revocation frameworks defined by Hagström et al. [11] and Cramer et al. [7], *Dom* is defined in a graph-theoretical way, where the nodes of the graph are the principals, and the labelled edges of the graph are *authorizations* that principals have granted to each other. *Dom* admits for one kind of positive authorization, denoted $+$, and two kinds of negative authorization, denoted $-_{\mathsf{SR}}$ and $-_{\mathsf{PR}}$ (the R in the subscript means "resilient"; it is used here as we will define extensions of *Dom* that have non-resilient negative authorizations). The set $\{+, -_{\mathsf{SR}}, -_{\mathsf{PR}}\}$ of authorization types is denoted by $\mathbf{T}_{Dom}$.

Definition 1. *An* authorization *is a tuple* (i, j, τ, π)*, where* $i, j \in \mathbf{S}$*,* $\tau \in \mathbf{T}_{Dom}$*,* $\pi \in \mathbf{P}$*.*

From a graph-theoretical point of view, an authorization is an edge from i to j labelled τ, π. The graph consisting of the principals and the authorizations is called the *authorization specification*. As the set of principals is constant, we also use the term *authorization specification* to refer to the set of authorizations that are in place.

In *Dom*, i's action of granting a permission π to j corresponds to adding $(i, j, +, \pi)$ to the authorization specification. i's action of revoking permission π from j through an SGR or PGR revocation corresponds to adding $(i, j, -_{\mathsf{SR}}, \pi)$ or $(i, j, -_{\mathsf{PR}}, \pi)$ respectively to the authorization specification. i's action of revoking permission π from j through a WGN revocation corresponds to deleting $(i, j, +, \pi)$ from the authorization specification. These correspondences inductively define a function $\mathbf{A}_{Dom}$ that maps any delegation-revocation profile Π over $\{(\mathsf{S},\mathsf{G},\mathsf{R}), (\mathsf{P},\mathsf{G},\mathsf{R}), (\mathsf{W},\mathsf{G},\mathsf{N})\}$ to an authorization specification (the base case is that $\mathbf{A}_{Dom}(\langle\rangle)$ is the empty authorization specification).

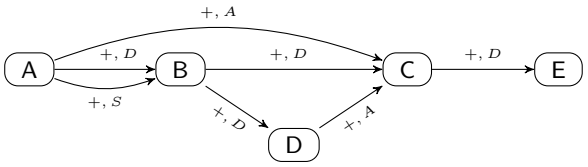
Since in a delegation-revocation profile Π a granting action of a delegation right can only occur directly after a granting action of a corresponding access right (see Section 3.1), an authorization $(i, j, +, D)$ can only be present in an authorization specification $\mathbf{A}_{Dom}(\Pi)$ if the authorization $(i, j, +, A)$ is also present. Conversely, an authorization (i, j, τ, A) for $\tau \in \{-_{\mathsf{SR}}, -_{\mathsf{PR}}\}$ can only be present if an authorization $(i, j, -_{\mathsf{SR}}, D)$ is present.

We visualize an authorization specification as in Example 4, in which A is the SOA. For every authorization (i, j, τ, π) in the authorization specification, this graph contains an edge from i to j labelled τ, π. We refrain from showing the authorizations that can be implied to exist by the considerations explained in the previous paragraph (for example, additionally to the depicted authorization $(A, B, +, D)$, there must also be an authorization $(A, B, +, A)$, which is not depicted).

Example 4. An authorization specification

We define a relation $\mathcal{R}$ on $\mathbf{P} \times (\mathbf{T}_{Dom} \times \mathbf{P})$ such that $\mathcal{R}(\pi, (\tau, \pi'))$ formalizes the notion that permission π is a prerequisite for being a legal grantor of an authorization of type τ and permission π':

Definition 2. $\mathcal{R}(\pi, (\tau, \pi'))$ *holds iff one of the following conditions is satisfied:*



- $\pi = D$, $\tau \neq -_{\mathsf{SR}}$ *and either* $\pi' = A$ *or* $\pi' = D$.
- $\pi = S$, $\tau \neq -_{\mathsf{SR}}$ *and* $\pi' = S$.
- $\pi = S$ *and* $\tau = -_{\mathsf{SR}}$.

In order to evaluate which principals have access given a certain authorization specification, we need to consider which authorizations are active and which ones are inactivated. For an authorization to be active, one prerequisite is that it must be connected back to the SOA through a chain of active authorizations that ensure that each principal along the chain is a legal grantor of the authorization in the chain granted by that principal. Additionally, a negative authorization $(i, j, -_{\mathsf{SR}}, \pi)$ inactivates every positive authorization from some principal k to j (as this negative authorization means that i has performed a Strong Global Resilient revocation onto j).

In order to formally specify which authorizations get inactivated when issuing a negative authorization, we define through a simultaneous inductive definition the notions of an authorization being *active* and an authorization being *directly inactivated* in Definitions 3 and 4.[2] The auxiliary notion of a directly inactivated authorization captures the idea of an authorization from k to j being inactivated by a negative authorization from i to j.

Definition 3. *Let* $\mathbf{A}$ *be an authorization specification. An authorization* (i, j, τ, π) *is* active *in* $\mathbf{A}$ *if it is not* directly inactivated *in* $\mathbf{A}$ *and there are nodes* $p_1, \ldots, p_n, p_{n+1}$ *satisfying the following properties:*

- $p_1 = SOA$, $p_n = i$ *and* $p_{n+1} = j$.
- *For* $1 \leq l < n$ *there is an authorization* $(p_l, p_{l+1}, +, \pi')$ *in* $\mathbf{A}$ *that is not* directly inactivated*, where* $\mathcal{R}(\pi', (\tau, \pi))$.
- *There do not exist* l, m *such that* $1 \leq l \leq m \leq n$ *and an authorization* $(p_l, p_{m+1}, -_{\mathsf{PR}}, \pi')$ *in* $\mathbf{A}$ *such that* $\tau = +$ *and* $\pi' = \pi$ *if* $m = n$*, and such that* $\mathcal{R}(\pi', (\tau, \pi))$ *otherwise.*

[2] These definitions inductively depend on each other. They should be read as an inductive definition with the well-founded semantics [8]. As discussed in Appendix A of Cramer et al. [7], there are exist paradoxical cases in which the well-founded semantics is three-valued rather than two-valued, so that for some authorizations it is undecided whether they are active or not. Such paradoxical cases only arise when strong revocation of the permission S depend on each other in a circular way. For the purpose of this paper we stipulate that *undecided* is treated as *false*, so that the principals directly affected by such a paradoxical situation will not have access until the paradoxical situation is resolved.

Definition 4. *Let* $\mathbf{A}$ *be an authorization specification. An authorization* $(i, j, +, \pi)$ *is* directly inactivated *in* $\mathbf{A}$ *if there is an* active *authorization* $(k, j, -_{\mathsf{SR}}, \pi)$ *in* $\mathbf{A}$.

The notion of an active authorization is used in the definition of access right:

Definition 5. *Let* $\mathbf{A}$ *be an authorization specification. A principal* j *has access right in* $\mathbf{A}$ *iff* j *is the* SOA *or there is an active authorization of the form* $(i, j, +, A)$ *for some node* i.

Now we are in a position to define the delegation-revocation framework *Dom*:

Definition 6. *Given a delegation-revocation profile* Π *over* $\{(\mathsf{S},\mathsf{G},\mathsf{R}), (\mathsf{P},\mathsf{G},\mathsf{R}), (\mathsf{W},\mathsf{G},\mathsf{N})\}$*, we define*

$$\mathit{Dom}(\Pi) := \{i \in \mathbf{S} \mid i \text{ has access right in } \mathbf{A}_{\mathit{Dom}}(\Pi)\}.$$

Example 5. Consider the authorization specification in Example 4. Let the principal B perform an action revoke(B,C, A, S,G,R), that is, a global revocation of access rights targeting the principal C (see Figure 1). The result of the action in the graph we add a negative authorization $(\mathsf{B},\mathsf{C}, -_{\mathsf{SR}}, A)$ (that implies also the negative authorization $(\mathsf{B},\mathsf{C}, -_{\mathsf{SR}}, D)$). Such a negative authorization is active, making the positive authorizations targeting C directly inactivated, and consequently making also the authorization previously issued by C, $(\mathsf{C},\mathsf{E},+, D)$, inactive.

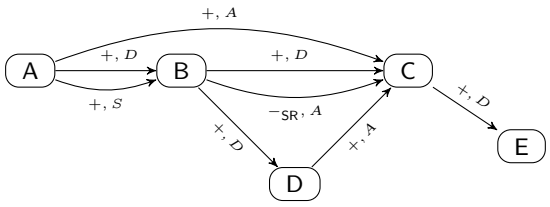


Fig. 1. Example 5

Dom satisfies all four postulates from Section 3.2. Locality and Resilience Indifference are satisfied vacuously, as *Dom* does not support any local revocation nor any pair of revocations that differ only in the resilience dimension.

Theorem 1. *Dom satisfies* Locality, Resilience Indifference, Access from Revocation, *and* Timing Indifference.

Proof. As explained in Section 4, *Dom* vacuously satisfies Locality and Resilience Indifference. The following reasoning sketches a proof of the claim that *Dom* satisfies the Access from Revocation postulate: The only way that a principal j could gain access right through a revocation action is that some previously inactive authorization

of the form $(i, j, +, A)$ gets reactivated. The only way for such an authorization to be reactivated is if some authorization $(k, l, +, D)$ in a chain of authorizations linking the SOA to i gets reactivated, or if $(i, j, +, A)$ stops being directly inactivated. But $(k, l, +, D)$ can only get reactivated if either some authorization $(k', l', +, D)$ in a chain of authorizations linking the SOA to k gets reactivated, or if $(k, l, +, A)$ stops being directly inactivated. Applying this reasoning inductively, we can conclude that the only way for $(i, j, +, A)$ to get reactivated is for it or some authorization of the form $(k, l, +, D)$ to stop being directly inactivated. But the only way in which an authorization of the form $(i, j, +, \pi)$ can stop to be directly inactivated is for an authorization of the form $(k, j, -_{\mathsf{SR}}, \pi)$ to be inactivated. The only way in which such an authorization can be inactivated is if an authorization $(k', j', +, S)$ in a chain of authorizations linking the SOA to k gets inactivated. The only way this can happen is if either an authorization of the form $(k'', j'', +, S)$ in a chain of authorizations linking the SOA to k' gets inactivated, or if $(k', j', +, S)$ starts to be directly inactivated. Applying this reasoning inductively, we can conclude that the only way in which an authorization of the form $(k, j, -_{\mathsf{SR}}, \pi)$ can get inactivated is if some authorization of the form $(k', j', +, S)$ starts to be directly inactivated. The only way this can happen is if some authorization of the form $(p, k', -_{\mathsf{SR}}, S)$ either gets added or gets reactivated. As the case of $(p, k', -_{\mathsf{SR}}, S)$ getting reactivated is analogous to the case of $(i, j, +, A)$ getting reactivated, we can see that $(p, k', -_{\mathsf{SR}}, S)$ can only get reactivated if some authorization of the form $(p', k'', -_{\mathsf{SR}}, S)$ either gets added or gets reactivated. Applying induction once more, we can conclude that the only way that j to get access is for some authorization of the form $(p, k', -_{\mathsf{SR}}, S)$ to get added, i.e. for some strong revocation of permission S to be performed, just as the Access from Revocation postulate claims.

To see why *Dom* satisfies Timing Indifference, note that the only way in which

$$\mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_1, a_2 \rangle \oplus \Pi_2) = \mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_2, a_1 \rangle \oplus \Pi_2)$$

can fail to hold is if a_1 is a granting action and a_2 is a WGN revocation between the same two principals. But in this case the preconditions of Timing Indifference are not satisfied. So whenever these preconditions are satisfied, the above equation holds, and thus

$$Dom(\Pi_1 \oplus \langle a_1, a_2 \rangle \oplus \Pi_2) = Dom(\Pi_1 \oplus \langle a_2, a_2 \rangle \oplus \Pi_2)$$

holds too, as required. □

5 Adding Non-Resilient revocation: DR

In this section we extend *Dom* to a delegation framework *DR* that introduces into the framework also the possibility of performing non-resilient revocations.

DR is defined by making the following modifications to *Dom*:

– New negative authorization types $-_{\mathsf{SN}}$ and $-_{\mathsf{PN}}$ are introduced i.e. $\mathbf{T}_{DR} := \{+, -_{\mathsf{SR}}, -_{\mathsf{PR}}, -_{\mathsf{SN}}, -_{\mathsf{PN}}\}$.

– We introduce a new element in the authorization specification, the *shields*, which protect a positive authorization from being inactivated by an earlier non-resilient negative authorization (see below).
– We redefine how performing a granting action modifies the authorization specification, introducing also the possibility of the shields.
– We modify the definition of *active* and *directly inactivated* in order to account for the shields.

The addition of the authorization types $-_{\mathsf{SN}}$ and $-_{\mathsf{PN}}$ means that $\mathbf{T}_{Dom}$ gets replaced by $\mathbf{T}_{DR}$ in the definitions of *authorization* and $\mathcal{R}$, and that "$\tau \neq -_{\mathsf{SR}}$" gets replaced by "$\tau \notin \{-_{\mathsf{SR}}, -_{\mathsf{SN}}\}$" in the definition of $\mathcal{R}$.

The behaviour of the global non-resilient revocations is the same in *DR* as in the delegation-revocation framework C from Cramer et al. [7]. However, Cramer et al. defined this behaviour without reference to shields. Instead, they included time stamps on the authorizations that indicate when an authorization was issued, and that were used to get the same effect as we get in *DR* through the use of shields. The reason why we use shields instead of time stamps is that time stamps include a lot of additional information into the authorization specification that is not relevant for determining access rights. By using shields we encode in the authorization specification only that part of the information about the temporal ordering of actions that is needed to correctly define access right.

A shield is a pair $((i, j, +, \pi), (k, j, -_{\mathfrak{p}N}, \pi'))$ for $\mathfrak{p} \in \{S, P\}$, i.e. a pair consisting of a positive authorization and a non-resilient negative authorization that target the same principal. In order to have the shields in the authorization specification, we need to redefine the authorization specification to be a more complex structure than a graph: An authorization specification is a structure consisting of a graph (with vertices and edges as in Section 6) plus a binary relation $\mathcal{S}$ on the edges of the graph, where we require that $\mathcal{S}((i, j, \tau, \pi), (k, l, \tau', \pi'))$ can only hold if $\tau = +$, $\tau' \in \{-_{\mathsf{PR}}, -_{\mathsf{SR}}\}$ and $l = j$.

A shield $((i, j, +, \pi), (k, j, -_{\mathfrak{p}N}, \pi'))$ represents the fact that $(i, j, +, \pi)$ results from a granting action performed after the revocation action that gave rise to $(k, j, -_{\mathfrak{p}N}, \pi')$, which by the intuitive meaning of *non-resilient* means that $(i, j, +, \pi)$ cannot be inactivated by $(k, j, -_{\mathfrak{p}N}, \pi')$. In order to ensure that the right shields are in the authorization specification, we need to modify the effect that performing a granting action has on the authorization specification. Whenever a granting action grant(i, j, π) is performed:

– $(i, j, +, \pi)$ is added to the authorization specification.
– If in the authorization specification there is a non-resilient negative authorization $(k, j, -_{\mathsf{SN}}, \pi')$, then add $\mathcal{S}((i, j, +, \pi), (k, j, -_{\mathsf{SN}}, \pi'))$ to the authorization specification.

The last step in the definition of *DR* is to modify the definition of *active* and *directly inactivated* in order to account for the shields:

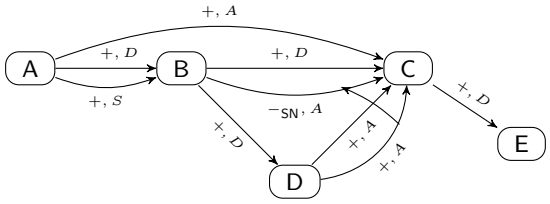
Definition 7. *Let* $\mathbf{A}$ *be an authorization specification with shield relation* $\mathcal{S}$*. An authorization* (i, j, τ, π) *is* active *in* $\mathbf{A}$ *if it is not* directly inactivated *in* $\mathbf{A}$ *and there are nodes* $p_1, \ldots, p_n, p_{n+1}$ *satisfying the following properties:*

– $p_1 = SOA$, $p_n = i$ *and* $p_{n+1} = j$.
– *For* $1 \leq l < n$ *there is an authorization* $(p_l, p_{l+1}, +, \pi')$ *in* $\mathbf{A}$ *that is not* directly inactivated, *where* $\mathcal{R}(\pi', (\tau, \pi))$.
– *There do not exist* l, m *such that* $1 \leq l \leq m \leq n$ *and an authorization* $(p_l, p_{m+1}, \tau', \pi')$ *in* $\mathbf{A}$ *such that*
  - $\tau' \in \{-_{\mathsf{PR}}, -_{\mathsf{PN}}\}$,
  - $((p_m, p_{m+1}, +, \pi''), (p_l, p_{m+1}, \tau', \pi')) \notin \mathcal{S}$, *where* $\pi'' = \pi$ *if* $m = n$, *and* $\mathcal{R}(\pi'', (\tau, \pi))$ *otherwise*,
  - $\tau = +$ *and* $\pi' = \pi$ *if* $m = n$, *and*
  - $\mathcal{R}(\pi', (\tau, \pi))$ *if* $m \neq n$.

Definition 8. *Let* $\mathbf{A}$ *be an authorization specification with shield relation* $\mathcal{S}$. *An authorization* $(i, j, +, \pi)$ *is* directly inactivated *in* $\mathbf{A}$ *if there is an* active *authorization* (k, j, τ, π) *in* $\mathbf{A}$ *such that* $\tau \in \{-_{\mathsf{SR}}, -_{\mathsf{SN}}\}$ *and* $((i, j, +, \pi), (k, j, \tau, \pi)) \notin \mathcal{S}$).

Example 6. The starting point is the graph in Example 4. B issues a non-resilient strong revocation of A targeted at C.

Let D re-issue the positive authorization (D,C, $+, A$); since there is a non-resilient negative authorization (B,C, $-_{\mathsf{SN}}, A$) targeting C, a shield ((D,C, $+, A$), (B,C, $-_{\mathsf{SN}}, A$)) is issued.



Following Definition 8, the authorization (D,C, $+, A$) is not directly inactivated by (B,C, $-_{\mathsf{SN}}, A$) since there is a shield from the former to the latter. (D,C, $+, A$) is actually active, and C's access rights are restored.

The framework *DR* satisfies all four postulates defined in Section 3.2 (Locality is satisfied vacuously due to the lack of local revocations):

Theorem 2. *DR satisfies* Locality, Resilience Indifference, Access from Revocation, *and* Timing Indifference.

Proof. *DR* vacuously satisfies Locality, as it does not support any local revocations.

One can easily see that *DR* satisfies Resilience Indifference by noting that $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathfrak{p},\mathsf{R}) \rangle}$ cannot contain a shield whose second element is a negative authorization that originates from the action revoke$(i, j, \pi, \mathfrak{d}, \mathfrak{p}, \mathsf{R})$.

Given that the resilient revocation behave the same way in *Dom* and *DR*, the fact that *DR* satisfies Access from Revocation directly follows from the facts that *Dom* satisfies Access from Revocation and that *DR* satisfies Resilience Indifference.

To see why *DR* satisfies Timing Indifference, first note that if neither of the actions a_1 and a_2 mentioned in the Timing Indifference postulate is a strong non-resilient or p-t-p non-resilient revocation, the postulate can be shown to be satisfied by *DR* in the same way as for *Dom*. Furthermore, note that the only way in which

$$\mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_1, a_2 \rangle \oplus \Pi_2) = \mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_2, a_1 \rangle \oplus \Pi_2)$$

can fail to hold when at least one of a_1 and a_2 is a strong non-resilient or p-t-p non-resilient revocation is if a shield gets added in one of the action pairs $\langle a_1, a_2 \rangle$ and $\langle a_2, a_1 \rangle$, but not in the other. The only way this can happen is if a_1 and a_2 target the same principal. But in this case the preconditions of Timing Indifference are not satisfied. □

6 Adding local revocations: DPR

In this section we extend the framework *DR* to a delegation-revocation framework *DPR* over Σ^*. In other words, *DPR* fully covers all three revocation dimensions, i.e. it can handle all ten revocation actions defined in Section 3.1. For this, we need to specify how *DR* gets modified so as to support local revocations.

As seen at the end of Section 4, the definition of local revocations presented by Cramer et al. [7] does not satisfy the postulate of Timing Indifference. The goal of this section is to define the local revocations in such a way that this postulate is satisfied. We do this by extending the framework *Dom* in the following way:

- We add a new set of nodes to the graph, the set **B** of bridges. A bridge can be used in delegation chains in order to preserve the effect of authorizations issued by a principal targeted by a local revocation.
- We introduce a new class of actions, *Local Revocations*.
- We appropriately modify the definition of the *authorization specifications* as well as the definition of when an authorization is *active*.

We define the set of *bridges* to be

$$\mathbf{B} := \{\textit{bridge}(i, j, \mathfrak{d}, \mathfrak{r}, \pi) | i, j \in \mathbf{S}, (\mathfrak{d}, \mathfrak{r}) \in (\{\text{S,P,W}\} \times \{\text{R,N}\}) \setminus \{\text{W,R}\} \text{ and } \pi \in \mathbf{P}\}.$$

Following a local revocation action revoke$(i, j, \mathfrak{d}, L, \mathfrak{r}, \pi)$, the purpose of *bridge*$(i, j, \mathfrak{d}, \mathfrak{r}, \pi)$ is to be a substitute for j in the delegation chains that ensure that the principals whose access right previously depended on j is preserved.

We extend the definition of an authorization-specification from Section 5 by allowing bridges to be nodes of the graph as well, and adding shields to this extended notion of a graph using the same definition that was used to add shields in Section 5, only that i, j, and k now refer to the new notion of a node (a principal or a bridge) rather than to the old notion of a node (just a principal). The main distinguishing factor between a principal and a bridge is that *a bridge cannot perform any action*, as bridges cannot be mentioned in delegation-revocation profiles. We say that a bridge *bridge*$(i, j, \mathfrak{d}, \mathfrak{r}, \pi)$ is a bridge *for* the principal j, and we indicate with $\mathbf{B}_j$ the set of the bridges for j.

We change the definition of how the authorization specification gets modified when a granting action or a global revocation action targeting a principal j is performed by adding not only an authorization ending in j, but also analogous authorizations ending in the bridges in $\mathbf{B}_j$. More precisely, the action grant(i, j, π) results in adding not only $(i, j, +, \pi)$, but also $(i, b, +, \pi)$ for any $b \in \mathbf{B}_j$ to the authorization specification; and the action revoke$(i, j, \mathfrak{d}, G, \mathfrak{r}, \pi)$ results in adding not only $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$, but also $(i, b, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ for any $b \in \mathbf{B}_j$ to the authorization specification.

In what follows we need to distinguish in the set $\mathbf{B}_j$ the bridges that are actually playing an active role in the graph, since they are associated to some active negative authorizations, from the ones that are not relevant. We call the former ones the *active* bridges for j, and denote the set of the active bridges for j by $\mathbf{B}_j^a$ (see Definition 9 below). Informally, the main idea is the following: Given a principal j, its bridges in $\mathbf{B}_j$ record all the global authorizations targeting j. In the moment a local revocation is performed by a principal i toward j, resulting into a negative authorization $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$, all the authorizations issued by j up to that point are 'copied' in the bridge *bridge*$(i, j, -_{\mathsf{SR}}, \pi)$, i.e. for every (j, k, τ', π') in the authorization specification, an authorization $(\textit{bridge}(i, j, \mathfrak{d}, \mathfrak{r}, \pi), k, \tau', \pi')$ is added to the authorization specification. In such a way, for every authorization (j, k, τ', π') that was active before the performing of a local revocation targeting j, we introduce a new authorization $(\textit{bridge}(i, j, -_{\mathsf{SR}}, \pi), k, \tau', \pi')$ that is active in the new graph. This ensures that whatever rights were granted by j before the local revocation are still supported by an active delegation chain that 'bypasses' the principal j through a bridge for j.

Performing a local revocation revoke$(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})$ has the following effects on an authorization specification:

1. For every principal k and every authorization (j, k, τ', π') in the authorization specification, an authorization $(\textit{bridge}(i, j, \mathfrak{d}, \mathfrak{r}, \pi), k, \tau', \pi')$ is added to the authorization specification.
2. For every principal k and every authorization (k, j, τ', π') in the authorization specification, an authorization $(k, \textit{bridge}(i, j, \mathfrak{d}, \mathfrak{r}, \pi), \tau', \pi')$ is added to the authorization specification.[3]
3. $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is added to the authorization specification.

The constraints defining which authorization are *active* and which are *inactive* must be changed in order to consider also the bridges, but only the active ones. Apart from reinterpreting the meaning of the word *node* and the domain of quantification of the variable $i, j, p_1, \ldots, p_{n+1}$ to include bridges as well as principals, Definition 8 remains unchanged, while we change Definition 7 simply adding the following condition:

– For $1 < l \leq n$, if $p_l \in \mathbf{B}_j$ for some principal j, then $p_l \in \mathbf{B}_j^a$.

[3] We add such a condition even though every authorization from k to j created due to a granting or global revocation action already has a copy from k to any bridge for j, because there can be authorizations from k to j created due to local revocations that must be added at this point.

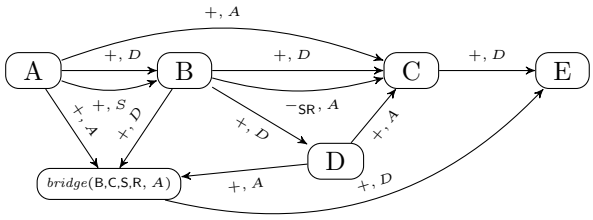
Note that the latter condition refers to the set of active bridges. So instead of building a *simultaneous inductive definition* consisting of Definitions 3 and 4 as in Section 4, here we build an analogous simultaneous inductive definition using Definition 4, the modified version of Definition 3, and a third component, Definition 9:

Definition 9. *Given a principal j, the set $\mathbf{B}^a_j$ is defined as follows: For every bridge$(i, j, \mathfrak{d}, \mathfrak{r}, \pi) \in \mathbf{B}_j$, bridge$(i, j, \mathfrak{d}, \mathfrak{r}, \pi) \in \mathbf{B}^a_j$ if and only if $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is an active authorization.*

According to the above constraints, when a local revocation revoke$(i, j, \mathfrak{d}, L, \mathfrak{r})$ is performed, a negative authorization $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is issued and a node *bridge*$(i, j, \mathfrak{d}, \mathfrak{r}, \pi)$ is associated to (i, j, τ, π). In case $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is inactive, also *bridge*$(i, j, \mathfrak{d}, \mathfrak{r}, \pi)$ is inactive and its presence is irrelevant. But if $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is active, *bridge*(i, j, τ, π) is active and ensures that all rights that were granted by j before the revocation are preserved.

Example 7. Consider the authorization specification in Example 4. Let the principal B perform an action revoke(B,C, A, S,L,R), i.e. a local revocation of access A targeting the principal C. If in the graph we simply added a negative authorization (B,C, $-_{\mathsf{SR}}, A$) as in Example 5, this would have the effect of a global revocation, inactivating also the authorization (C,E, $+, D$) previously issued by B. Now we use bridges to model the locality of the revocation. In the visualization of the graph, we depict only the bridge that is relevant for the performed local revocation revoke(B,C, A, S,L,R), namely *bridge*(B,C,S,R, A).

In step 1 of the three steps describing the effects of the local revocation revoke(B,C, A, S,L,R), we add an authorization (*bridge*(B,C,S,R, A), j, τ, π) for every authorization $(\mathsf{C}, j, \tau, \pi)$. In this case we only have to replicate the authorization (C,E, $+, D$) as (*bridge*(B,C,S,R, A), E, $+, D$). In step 2, we do not need to add anything, because previously only non-local actions have been performed, and all the non-local actions targeting C have already given rise to authorizations targeting *bridge*(B,C,S,R, A) (see footnote 3). Finally, we add the negative authorization (B,C, $-_{\mathsf{SR}}, A$). The resulting graph is the following.



As the negative authorization (B,C, $-_{\mathsf{SR}}, A$) is active, the node *bridge*(B,C,S,R, A) is also active. Then it is easy to check that the principal E obtains through the bridge the delegation right that C had previously granted to E, while C itself no longer has access or delegation right.

The framework *DPR* satisfies all four postulates defined in Section 3.2:

Theorem 3. *DPR satisfies* Locality, Resilience Indifference, Access from Revocation, *and* Timing Indifference.

Proof. First we prove that *DPR* satisfies Locality.

Suppose that the preconditions of the Locality postulate are satisfied. The idea behind the proof is that the treatment of *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$ in the local revocation does not give access to any principal that did not have access before the revocation, but ensures that for every active delegation chain that gave access to a principal $k \neq j$ before the loval revocation, there is a corresponding active delegation chain (possibly going through *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$ instead of through j) that gives access to k after the local revocation. To formalize this idea, we need to prove the equality statement of Locality by proving a set inclusion in both direction.

First suppose that $k \in \mathit{DPR}(\Pi) \cup \{j\}$. If $k = j$ or $k = \mathrm{SOA}$, then trivially $k \in \mathit{DPR}(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle) \cup \{j\}$. If $k \in \mathit{DPR}(\Pi)$ and $k \neq \mathrm{SOA}$, then there is an authorization $(l, k, +, \pi')$ with $\pi' = A$ or $\pi' = D$ that is active in $\mathbf{A}_\Pi$. So there is a chain of authorizations in $\mathbf{A}_\Pi$ satisfying the properties specified in Definition 3 (the definition of *active* in *Dom*) plus the additional property that gets added to the definition of *active* in Section 6. Any authorization in $\mathbf{A}_\Pi$ is also in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$, so this chain also exists in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$. If either this chain does not go through j or the negative authorization $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is not active, then this chain also satisfies the properties of the definition of "active" in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$, so that $(l, k, +, \pi')$ is active in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$, i.e. $k \in \mathit{DPR}(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle)$, as required. If this chain does go through j and $(i, j, -_{\mathfrak{d}\mathfrak{r}}, \pi)$ is active, then the chain formed from this chain by replacing j by *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$ satisfies the properties of the definition of "active" in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$, so that $(l, k, +, \pi')$ is active in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$, i.e. $k \in \mathit{DPR}(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle)$, as required.

Now suppose that $k \in \mathit{DPR}(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle) \cup \{j\}$. If $k = j$ or $k = \mathrm{SOA}$, then trivially $k \in \mathit{DPR}(\Pi)$. If $k \in \mathit{DPR}(\Pi \oplus \langle \mathsf{revoke}(i, j, \pi, \mathfrak{d}, \mathsf{L}, \mathfrak{r})\rangle)$ and $k \neq \mathrm{SOA}$, then there is an authorization $(l, k, +, \pi')$ with $\pi' = A$ or $\pi' = D$ that is active in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$. So there is a chain of authorizations in $\mathbf{A}_{\Pi \oplus \langle \mathsf{revoke}(i,j,\pi,\mathfrak{d},\mathsf{L},\mathfrak{r})\rangle}$ satisfying the properties of the definition of "active". If this chain does not go through *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$, then its authorizations also exists in $\mathit{DPR}(\Pi)$, and the chain satisfies the properties of the definition of "active" also in $\mathit{DPR}(\Pi)$, so that $(l, k, +, \pi')$ is active in $\mathit{DPR}(\Pi)$, i.e. $k \in \mathit{DPR}(\Pi)$, as required. If this chain does go through *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$, then the chain formed from this chain by replacing *bridge*$(i, j, \mathfrak{d}, \mathfrak{r})$ by j satisfies the properties of the definition of "active" in $\mathit{DPR}(\Pi)$, so that $(l, k, +, \pi')$ is active in $\mathit{DPR}(\Pi)$, i.e. $k \in \mathit{DPR}(\Pi)$, as required. This finishes the proof that *DPR* satisfies Locality.

One can prove that *DPR* satisfies Resilience Indifference in a similar way as *DR* was shown to satisfy Resilience Indifference in Theorem 2.

To see why *DPR* satisfies Access from Revocation, note that if a is a local revocation, we can show that $\mathit{DPR}(\Pi \oplus \langle a\rangle) \subseteq F(\Pi)$ in a similar way as we showed Locality above, whereas if a is a global revocation, we can show that $\mathit{DPR}(\Pi \oplus \langle a\rangle) \subseteq F(\Pi)$ in a similar way as in the proof that *Dom* satisfies Access from Revocation in Theorem 1.

To see why *DPR* satisfies Timing Indifference, first note that if neither of the actions a_1 and a_2 mentioned in the Timing Indifference postulate is a local revocation, the postulate can be shown to be satisfied by *DPR* in the same way as for *DR*. Furthermore, note that the only way in which

$$\mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_1, a_2 \rangle \oplus \Pi_2) = \mathbf{A}_{Dom}(\Pi_1 \oplus \langle a_2, a_1 \rangle \oplus \Pi_2)$$

can fail to hold when at least one of a_1 and a_2 is a local revocation is if the target of a local revocation in $\{a_1, a_2\}$ is the principals that performs the other action, because only in this case can the creation of authorizations coming out of $\mathit{bridge}(i, j, \mathfrak{d}, \mathfrak{r}, \pi)$ based on authorizations coming out of j depend on the temporal ordering of a_1 and a_2. But in this case the preconditions of Timing Indifference are not satisfied. □

Note that of the delegation-revocation frameworks that we have defined, *DPR* is the only one which satifies all four postulates in a non-vacuous way, and the only one which supports all ten revocation actions defined in Section 3.1.

7 Conclusion and Future Work

Following an idea first proposed in Cramer et al. [7], we analyse delegation revocation using the *axiomatic method*. In contrast to Cramer et al. [7], we define relatively simple and readily understandable postulates. This way, our use of the axiomatic method resembles more closely the standard way it is used in social choice theory and belief revision. The four postulates that we define formalize desirable features of revocation scheme, i.e. expactations about the behaviour of various revocation schemes that are based on the intended meaning of the three revocation dimensions first identified by Hagström et al. [11].

We have shown that none of the existing frameworks satisfies all four defined postulates. Even the framework defined in Cramer et al. [7] fails to satify one of the postulates in the case of local revocations. In order to define the delegation-revocation framework *DPR* that satisfies all four postulates while supporting all meaningful revocation schemes, we first defined the simple basic delegation-revocation framework *Dom* that supports only three simple revocation schemes, which we extended in a stepwise way first to *DR* and finally to *DPR*.

We believe that the approach taken in this paper can be a fruitful foundation for future research. Concerning the specific topic of this paper, further research should study the possibility of defining further postulates for relegation revocation frameworks and of proving representation results similar to those in belief revision (see Rott [13]). Furhermore, the approach from the present paper based on simple postulates could be combined with the approach from Cramer et al. [7] that formulated a complex postulate based on a dedicated logic called Trust Delegation Logic. Combining these approaches could lead to an improved variant of Trust Delegation Logic that fully characterizes a delegation-revocation framework that additionally satisfies all the desirable simple postulates.

Finally, we consider the work presented in this paper as a proof of concept showing the fruitfulness of applying the axiomatic method to problems in computer security.

We believe that other problems sudied in computer security could also profit from being analyzed using the axiomatic method.

Acknowledgements

The work of Marcos Cramer was supported by the *Fonds National de la Recherche*, Luxembourg, via the INTER project *Specification logics and Inference tools for verification and Enforcement of Policies*. The work of Giovanni Casini has been supported by the *Fonds National de la Recherche*, Luxembourg, and cofunded by the *Marie Curie Actions* of the European Commission (FP7-COFUND) (AFR/9181001).

References

1. Aucher, G., Barker, S., Boella, G., Genovese, V., van der Torre, L.: Dynamics in Delegation and Revocation Schemes: A Logical Approach. In: Li, Y. (ed.) Data and Applications Security and Privacy XXV, Lecture Notes in Computer Science, vol. 6818, pp. 90–105. Springer Berlin (2011)
2. Barker, S., Boella, G., Gabbay, D., Genovese, V.: Reasoning about delegation and revocation schemes in answer set programming. Journal of Logic and Computation (2014)
3. Bertino, E., Samarati, P., Jajodia, S.: An extended authorization model for relational databases. Knowledge and Data Engineering, IEEE Transactions on 9(1), 85–101 (Jan 1997)
4. Bertino, E., Jajodia, S., Samarati, P.: A Non-timestamped Authorization Model for Data Management Systems. In: Proceedings of the 3rd ACM Conference on Computer and Communications Security. pp. 169–178. CCS '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/238168.238211
5. Chander, A., Dean, D., Mitchell, J.C.: Reconstructing trust management. Journal of Computer Security (2004)
6. Cramer, M., Hertum, P.V., Lapauw, R., Dasseville, I., Denecker, M.: Resilient Delegation Revocation with Precedence for Predecessors Is NP-Complete. In: 2016 IEEE 29th Computer Security Foundations Symposium (CSF). pp. 432–442 (June 2016)
7. Cramer, M., Ambrossio, D.A., van Hertum, P.: A Logic of Trust for Reasoning about Delegation and Revocation. In: Proceedings of the 20th ACM Symposium on Access Control Models and Technologies. pp. 173–184 (2015), http://doi.acm.org/10.1145/2752952.2752968
8. Denecker, M.: The Well-Founded Semantics Is the Principle of Inductive Definition. In: Dix, J., del Cerro, L., Furbach, U. (eds.) Logics in Artificial Intelligence, Lecture Notes in Computer Science, vol. 1489, pp. 1–16. Springer Berlin Heidelberg (1998)
9. Fagin, R.: On an Authorization Mechanism. ACM Trans. Database Syst. 3(3), 310–319 (Sep 1978), http://doi.acm.org/10.1145/320263.320288
10. Griffiths, P.P., Wade, B.W.: An Authorization Mechanism for a Relational Database System. ACM Trans. Database Syst. 1(3), 242–255 (Sep 1976), http://doi.acm.org/10.1145/320473.320482
11. Hagström, Å., Jajodia, S., Parisi-Presicce, F., Wijesekera, D.: Revocations – A Classification. In: Proceedings of the 14th IEEE Workshop on Computer Security Foundations. pp. 44–. CSFW '01, IEEE Computer Society, Washington, DC, USA (2001), http://dl.acm.org/citation.cfm?id=872752.873508

12. Li, N., Grosof, B.N., Feigenbaum, J.: Delegation Logic: A Logic-based Approach to Distributed Authorization. ACM Transaction on Information and System Security (2003)
13. Rott, H.: Change, Choice and Inference: a study of belief revision and nonmonotonic reasoning. Oxford University Press, Oxford, UK (2001)
14. Tamassia, R., Yao, D., Winsborough, W.H.: Role-Based Cascaded Delegation. In: Proceedings of the 9th ACM symposium on Access control models and technologies (2004)
15. Yao, D., Tamassia, R.: Compact and Anonymous Role-Based Authorization Chain. ACM Transactions on Information and System Security (2009)
16. Zhang, L., Ahn, G.J., Chu, B.T.: A rule-based framework for role-based delegation and revocation. ACM Transactions on Information and System Security (2003)