Advance Access publication on 25 August 2015
doi:10.1093/comjnl/bxv060

A Novel Permission Hierarchy for RBAC for Dealing with SoD in MAC Models

Simeon Veloudis[1]* and Nimal Nissanke[2]

[1]*South East European Research Centre (SEERC), International Faculty of the University of Sheffield, CITY College, Thessaloniki, Greece*
[2]*Emeritus Professor, London South Bank University, London, UK*
**Corresponding author: sveloudis@seerc.org*

Separation of duty (SoD) is a fundamental principle of computer security that has not been addressed sufficiently in multi-level security (MLS) mandatory access control (MAC) models, as realized through the adoption of the Bell-LaPadula (BLP) model. This is due to the lack of means at present to express SoD constraints in MAC. The primary objective of this paper is to overcome this but within a framework that allows for rigour and linguistic features to express SoD constraints, while retaining the core security properties of BLP, namely the Simple Security Property and ★-Property. To this end, we propose a formal framework which bridges the BLP model with the more general hierarchical role-based access control (RBAC) model. Our framework is based on a *hierarchy of permissions* that is founded on a novel concept of *permission capacity*, determined on the basis of the security levels that characterize objects in MLS models. Such a hierarchy naturally provides a solid basis for defining role seniority and deriving a hierarchical ordering on roles within MLS models. SoD constraints are expressed by means of conflicting permissions that give rise to *mutually exclusive* roles.

Keywords: mandatory access control; multi-level security models; Bell-LaPadula security model; role-based access control

Received 7 September 2014; revised 20 July 2015
Handling editor: Andrew Martin

1. INTRODUCTION

Access control methodologies fall into two broad categories: one catering for the assurance of specific security goals—such as confidentiality and integrity—directly by virtue of the underlying security models, and the other for delegating the assurance of such goals to security administrators and information owners, by providing the means for expressing relevant security measures. Notable exemplars of the former category include mandatory access control (MAC) methodologies such as the Bell-LaPadula (BLP) [1], Biba [2] and Clark–Wilson security models [3] for assuring, respectively, confidentiality, integrity and avoidance of certain forms of conflict of interest. A characteristic advantage of these models is the consistency in the interpretation of security goals in arriving at access control decisions, once the security clearances of subjects and the security classifications of objects are appropriately established. Exemplars of the latter category, i.e. which delegate the assurance of security goals to security administrators and information owners, include the Role-Based Access Control (RBAC) [4, 5] and Discretionary Access Control (DAC) [6–8] methodologies. An outline of BLP and RBAC models follow later in Section 3.

While DAC leaves decisions controlling access to named objects at the discretion of their owners, RBAC assumes greater control over such decisions by taking an institutional standpoint, that is, by adopting the institutional practices as the basis for formulating security policies and by ensuring that such policies are enforceable on an institution-wide scale. However, recognizing the complexity arising as a result in the management of security policies, especially in the face of potentially

frequent changes in institutional security requirements, RBAC relies on the notion of a *role* as a core concept in order to simplify the overall design and incorporate such changes with minimal effect on the underlying security architecture. With the adoption of the notion of a role, RBAC shares a general standpoint with the former category of MAC security methodologies, namely the ability to consistently interpret security requirements, in this case not with respect to a particular security goal, but with respect to the roles that the individuals play in an organization.

Through the adoption of the notion of a role, RBAC is capable of efficiently accommodating another important common practical necessity, namely, the need for specifying *Separation of Duty* (SoD) constraints [3]. SoD is a security principle used to formulate multi-person control policies. It maintains that if a sensitive task requires n distinct operations for its completion, k of these operations are considered conflicting and must be performed by different users [9]. An outline of SoD constraints follows later in Section 2.2. Its aim is to distribute the responsibility and authority for a task to two or more people and thus significantly mitigate the risk of inadvertent mistakes, fraud or sabotage.

Although SoD has been studied extensively within the context of role-based models in mainly commercial settings [4, 9–11], it has not found sufficient attention in MLS MAC models. Nevertheless, SoD constraints may be equally significant in contexts in which fundamental security properties such as confidentiality and integrity are typically assured through the adoption of the BLP and Biba (MLS) models [1, 2]. For example in a military context, where confidentiality may be assured through the BLP model, we might additionally require that two different individuals must independently arm and launch a nuclear missile [12].

Motivated and driven by this shortcoming, the work reported in this paper proposes a novel formal framework for: (i) dealing with permissions, in particular, assigning permissions to roles based on security levels that conform with those in BLP, (ii) dealing with *conflicting* permissions and satisfying SoD constraints through the adoption of a novel view of role seniority and permission inheritance and (iii) providing an alternative definition of role seniority as understood in RBAC. The framework is underpinned by confidentiality objectives as advocated by the BLP model, i.e. as expressed by its Simple Security and ★-Security policies. The main benefit of our approach is its ability to incorporate confidentiality assurances within a hierarchical RBAC setting while, at the same time, addressing SoD constraints in MLS systems.

Unlike previous related works [13–15], our framework is based on a *hierarchy of permissions* which is in turn founded on the novel concept of *permission capacity*. This concept is closely related to the security levels that characterize objects in MLS systems; it may serve, for example, as a measure of competences required in, or security vulnerabilities posed by, the performance of the operation associated with a given permission. Permission hierarchies lie at the foundation of how role hierarchies are constructed in MLS systems; their relation to real-world scenarios is demonstrated through the example in Section 4.6. In our view, the determination of the seniority of a role on the basis of the capacities of its associated permissions[1] is intuitive and natural in an MLS context. More importantly, it has a significant practical benefit, namely, in that it greatly simplifies authorization management in large systems by absolving security administrators from the burden of having to explicitly assign seniority levels to roles. In addition, it allows a role to inherit permissions *directly*, i.e. dispensing with the requirement for these permissions to be also assigned to more junior roles as in conventional RBAC. It is emphasized here that our work does not aspire to devise yet another approach for the specification of SoD constraints. Nor does it claim that it is a simpler substitute for either RBAC or BLP, each taken on its own, as it relies on features of both of them taken together.

It is worth mentioning here that the Simple Security and ★-Security policies are not unique to the BLP model; the same terminology is encountered in the Chinese Wall security model due to Brewer and Nash [17]. However, despite the similarity of terms, these notions are fundamentally different in the two security models. In the Chinese Wall security model, the two policies are designed to prevent conflicts of interest arising from having prior access to information sources by controlling access to the relevant objects depending on access rights that a given subject has had or already has. In contrast, in BLP the access rights to objects are determined solely by clearance levels possessed by subjects relative to the security levels attributed to the objects. Safeguarding confidentiality being our primary security objective, BLP suits our purpose better.

This paper is structured as follows. Section 2 describes related work and highlights their major differences with the approach presented in this paper. Section 3 presents an overview of the BLP model from the perspective of our development and introduces some of the relevant RBAC notation. Section 4 presents a setting where objects and permissions can be meaningfully associated with each other, akin to the object-oriented view and introduces the concept of permission capacity; it demonstrates how the latter concept can form the basis for the definition of a hierarchical ordering on permissions in the MLS context. Section 5 introduces *explicit* assignment of permissions to roles and defines a hierarchical ordering on roles. It also proposes a formal framework for the *implicit* assignment of permissions to roles and introduces the concept of conflicting permissions and roles from a SoD perspective. Section 6 defines a suitable user-to-role assignment relation that limits the role membership of users to non-conflicting roles. Section 7 demonstrates that the simple security property and ★-property are satisfied in a SoD-aware BLP model—one which by design prevents users from performing conflicting operations. Finally, Section 8 presents conclusions and future work.

[1] An RBAC role is a named collection of permissions [16].

2. RELATED WORK

The purpose of this section is to set the broader context for our work by shedding some light on existing relevant research. In this respect, it briefly summarizes works in the realm of RBAC and MAC and highlights their major differences from our approach. It also briefly presents research on the incorporation of SoD in role-based systems.

2.1. RBAC and MAC

Hierarchical role-based access control models [4, 16, 18] have attracted considerable research interest in recent years due to their inherent ability to reflect organizational structure and to interpret security requirements across different roles in a consistent manner, and thereby to facilitate easier administration of institutional security. Delegation of permissions and roles is an important aspect of RBAC and has been studied widely [19–21]. The growing practical interest in RBAC, as well as the wide-ranging research on the subject, is well reflected in the initiative to draw a standard on a unified model for RBAC [16].

In [13], the authors describe an approach whereby RBAC is configured for accommodating the BLP model and hence demonstrate the generality of the former with respect to the latter. Despite the significance of its contribution and the clarity of its basic idea, it provides an abstract and purely mechanistic treatment of roles within the context of MLS: one in which roles are used to model labels of the lattice derived from the security levels. Although suitable for demonstrating the generality of hierarchical RBAC over BLP, such an abstract view can only characterize roles on the basis of the effect—either purely observational or alterational-only—that they may bring about on *sets* of objects that lie at certain security levels. Such a view fails, in essence, to take into account a crucial aspect of the underlying semantics of a role, namely, the effect that it brings about on *individual* objects. Thus, such an approach overlooks the inherent nature of a role, as determined by the operations permitted on the objects that the role is associated with. A clear disadvantage of such an isolated treatment of roles is that it precludes from the outset the expression of SoD constraints. In contrast, the work reported in this paper provides a different view of a role in an MLS context, one which fully conforms with a role's nature in the RBAC methodology, in essence as a collection of permissions. Thus, whereas in [13] roles derive their seniority *directly* from the security classification provided by the underlying BLP model, in our work roles derive their seniority from the permissions that they entail. Consequently, our work provides the ability to express SoD constraints in an MLS context.

The work reported in [22] attempts to develop a mapping algorithm to transform BLP security policies to equivalent RBAC policies. Nevertheless, the authors employ certain unconventional notations that hinder a proper understanding of their approach. In [23], a generic framework for the formal comparison of different access control models is presented. The approach is based on the notion of simulations: one access control model is more restrictive than another iff, for any implementation of the former, an analogous implementation of the latter can be devised. The authors use their approach to compare BLP with RBAC and conclude that the former is strictly more restrictive than the latter.

As already mentioned, the framework presented in this work is founded upon the notion of permission capacity which determines the security level of a permission and thus allows for the definition of a permission hierarchy. It is not the first time that security levels are assigned to RBAC permissions. In [14], the security level of a permission is determined on the basis of the roles to which the permission has been assigned; however, as the author concedes, this definition is problematic as it requires that for a user to obtain a certain permission, the user must also acquire all roles to which that permission has been assigned. In our view, such a definition of a permission's security level is cumbersome and counter-intuitive, especially from a MAC standpoint.

Crampton [14] defines a seniority relation on permissions which, however, is completely detached from the concept of permission security levels: for any two permissions p and p', p' is senior to p (in the sense that p' subsumes p) iff the former enjoys a richer set of access modes[2] than the latter on the same underlying object o. Nevertheless, we consider this to be a rather limited view of permission seniority from an MLS standpoint as it completely overlooks object classifications and thus restricts seniority comparisons with permissions that are defined on the same object.

In [15], the Oriented Permission RBAC (OP-RBAC) model is presented, which is founded upon Crampton's approach to permission inheritance proposed in [14]. In this approach, permissions are characterized as either 'up-oriented', 'down-oriented' or 'neutral'. An up-oriented permission can be inherited by any role that has a higher seniority than any of the roles to which the permission is *explicitly* assigned; similarly, a down-oriented permission can be inherited by any role that has a lower seniority than any of the roles to which the permission is explicitly assigned; in contrast, neutral permissions are not inherited by any roles to which they have not been explicitly assigned. OP-RBAC is specifically preconfigured to conveniently accommodate the BLP model without the need of introducing a dual role hierarchy as, for example, in [13]. This is demonstrated in [15] by using OP-RBAC to implement various different versions of the BLP model. This work differs from ours in the following important aspects. Firstly, no explicit reference is made pertaining to the incorporation of SoD constraints in the BLP model.

[2] In [14], a permission takes the form $(o, \{m_1, \ldots, m_k\})$ where o is an object and m_i (for each $i \in 1, \ldots, k$) is an *access mode*. For abitrary finite sets of access modes M and M', if $p = (o, M)$ and $p' = (o, M')$, then p' is senior to p iff $M \subseteq M'$.

Secondly, no hierarchical ordering is identified between permissions in the MLS context which precludes the derivation of a role's seniority from its constituent permissions.

It is widely accepted that assigning permissions to users indirectly, via roles, in RBAC may have potentially negative repercussions on the appropriateness of the operations that are exposed to users. This may be caused, for example, by inadvertently affecting the role seniority relation. In an attempt to address this shortcoming, Ref. [24] discerns a fragment of RBAC called '*bi-sorted role-based access control*'. In this approach, two distinct objects of indirection are identified: the *proper role*, which applies only to users, and the *demarcation*, which applies only to permissions. These two objects are linked up via a 'grant' relation which enables users assigned to proper roles to acquire appropriate demarcations, hence permissions. The authors maintain that such a decoupling of roles enhances organizational scalability, e.g. by preventing roles from dynamically becoming accessible to a wider, or narrower, set of users when permissions are added to, or removed from, roles.

2.2. SoD

The concept of SoD has long existed in the realm of information security as a core mechanism for preventing fraud and mitigating errors [3]. As reported in [25], there are at least two approaches to enforce SoD policies. One is the history-based approach which takes into account all the actions that a user has performed on the system in order to dynamically determine whether the user can be granted permission to perform a certain operation. The second is the static permission-assignment approach which a priori assigns to each user an appropriate set of permissions such that a minimum number of users is required to collectively perform a sensitive task.

SoD-motivated constraints have been studied widely in the relevant literature, especially within the context of RBAC-protected systems. In [26], the authors formally specify a wide range of static and dynamic SoD properties in secure RBAC systems and identify their interrelationships. These interrelationships are intended to aid the implementation of SoD policies which are defined as conjunctions of SoD properties. On a similar note, albeit in a less formal manner, Ref. [9] introduces the notion of history-based SoD and identifies different kinds of SoD variations and the mechanisms required to implement them.

The works reported in [11, 12, 27] propose formal languages for specifying a variety of increasingly sophisticated SoD constraints. In [27], the authors present the elements, syntax and semantics of *RCL 2000*, a formal language for specifying role-based authorization constraints and show how this language can be used to express SoD constraints as well as obligation constraints, i.e. constraints that require a user to be assigned certain role combinations. In [11], a graphical access control model is proposed which ultimately aims at simplifying the process of constraint specification by expressing SoD constraints as binary relationships. The authors demonstrate that their approach is capable of expressing a variety of constraints. Crampton [12] proposes a set-theoretic approach for the specification of SoD constraints which, as the author claims, is simpler than other similar approaches. The relationship between static, dynamic and historical SoD constraints is identified and discussed. Different flavours of SoD—such as role-based, permission-based and object-based—are specified and enforced through the use of a hypothetical 'constraint monitor'.

In [25], the authors stress the distinction between static SoD policies and mutually exclusive role constraints (the latter are simply a mechanism for enforcing the former) and demonstrate that the verification problem—i.e. the problem of verifying that a set of constraints implements a certain SoD policy in RBAC–is intractable. They also propose an approach for the efficient generation of the minimum set of constraints to enforce a given SoD policy. In [28], a framework is developed for specifying SoD constraints in the generalized temporal RBAC (GTRBAC) [28]. The work reported in [29] proposes a genetic algorithm for solving the minimum user assignment problem under multiple static mutually exclusive role (SMER) constraints by relating it to the chromatic number problem in graph theory.

In [30], a language for the specification of static and dynamic SoD constraints in role-based workflow systems is proposed. The problem of checking whether a workflow with such constraints has a valid user-to-task assignment is discussed and a relevant algorithm is proposed. A similar problem is discussed in [31]. Staying within the realm of workflow systems, but on a more abstract note, Ref. [32] proposes a novel algebra for the specification of high-level policies that combines requirements on the number of users motivated by SoD considerations with requirements in users' attributes. In a similar spirit, Refs. [33, 34] extend the work in [32] by increasing its expressiveness through the incorporation of multisets and by allowing for changing role assignments during workflow execution. In addition, they take advantage of CSP's [35] operational semantics in order to bridge the rift between abstract algebraic specifications on the one hand, and run-time enforcement mechanisms on the other.

Habib *et al.* [36] identifies several flaws that are related to dynamic SoD and that arise when mutually conflicting permissions are dealt with at the role level; for instance: (i) any permissions that are associated with mutually exclusive roles are effectively rendered conflicting (whereas, in fact, they may not be); (ii) users often have to switch between entire sessions to prevent them from obtaining conflicting permissions; (iii) security administrators have no means other than roles for exercising control over conflicting permissions. To overcome these flaws, the authors propose a solution whereby a role is divided into an 'inner' and an 'outer' part comprising, respectively, conflicting and non-conflicting permissions; users are allowed to activate the outer parts of mutually exclusive roles.

In a similar vein, the work in [37] recognizes drawbacks inherent in manipulating SoD requirements at the role level,

e.g. through static and dynamic mutually exclusive role (SMER and DMER) constraints. Further, they identify the need for treating SoD policies at the task level, rather than at an individual step level, as well as the need for distinguishing between 'risky' and 'credible' users. They formally demonstrate that the enforceability of a SoD policy is mathematically intractable and assess the safety of SoD requirements as a satisfiability problem.

3. THE BLP AND RBAC MODELS—PRELIMINARIES

This section outlines the BLP and RBAC security models focusing only on those features used in this paper. This is standard material [5, 7] and is not relegated to an appendix since an understanding of the basic concepts and familiarity with the notation are essential for an uninterrupted presentation.

3.1. BLP—common notations

In addition to types S and O, denoting, respectively, the sets of subjects and objects, the BLP security model considers a partial order $(L, \leq)$, where L denotes a set of *security levels* and $\leq$ an ordering relation on L. A security level in our work is determined solely on the basis of the security classification of an object.[3] Security levels are associated with both objects and subjects (in the latter case, they are often referred to as the security clearance of the subjects).

The BLP model also considers a set of access rights, $\{rd, wr, ap, ex\}$: rd for '`read`' (observational rights only), wr for '`write`' (alterational and obervational rights), ap for '`append`' (alterational rights only) and ex for '`execute`'. In this paper, we shall be focusing on a reduced set of similar access rights, namely $A = \{rd, ap\}$, with the same semantics as in BLP. BLP's '`write`' access right is modelled here as $\{rd, ap\}$, while the '`execute`' access right is not considered in this work.

The BLP model includes an 'access permission matrix' M with indices s and o, and a set of tuples $B \in \mathbb{P}(S \times O \times \mathbb{P}A \setminus \emptyset)$ in the form of a 'table' with the meaning that, for any $s \in S, o \in O$ and $x \in \mathbb{P}A \setminus \emptyset$, $(s, o, x) \in B$ iff s is currently performing an operation x on o.

BLP uses three functions

$$f_S, f_C : S \to L, \quad f_O : O \to L \tag{1}$$

such that $f_O(o)$ represents the security level assigned to each object o from the point of view of confidentiality (i.e. sensitivity level), $f_S(s)$ gives the maximum clearance level enjoyed by each subject s and $f_C(s)$ the current security level of s:

$$\forall s \in S \bullet f_C(s) \leq f_S(s) \tag{2}$$

The triple (f_S, f_C, f_O) is usually denoted by $\mathbf{f}$. In the BLP security model, the ss- and $\star$-properties may be stated as follows; see [7].

ss-Property (simple security property): a state $(B, M, \mathbf{f})$ satisfies the ss-Property if

$$\forall s \in S; o \in O \bullet (s, o, v) \in B \Rightarrow f_S(s) \geq f_O(o) \tag{3}$$

for $v \in \mathbb{P}A \setminus \emptyset$, $v = \{rd\}$ or $v = \{rd, ap\}$. This property ensures that a subject can only read objects of a sensitivity level lower than the subject's maximum clearance.

$\star$-Property: a state $(B, M, \mathbf{f})$ satisfies the $\star$-Property if

$$\begin{aligned} &\forall s \in S; o \in O \bullet (s, o, m) \in B \Rightarrow \\ &f_C(s) \leq f_O(o) \wedge \\ &(\forall o' \in O \bullet (s, o', v) \in B \Rightarrow \\ &f_O(o') \leq f_O(o)) \end{aligned} \tag{4}$$

for $m, v \in \mathbb{P}A \setminus \emptyset$, $m = \{ap\}$ or $m = \{ap, rd\}$, and $v = \{rd\}$ or $v = \{rd, ap\}$. In the interest of absolute clarity, we have represented m and v as sets and explicitly enumerated their elements. Visualizing the lattice defined by the partial order $(L, \leq)$ as a directed graph with arcs running upwards from lower nodes to higher nodes (in a vertically laid visual representation), the two properties in (4) ensure that any information flow would take place only upwards, that is, from entities of lower levels of confidentiality to those of higher levels.

3.2. RBAC—common notations

Though there are many formalizations of RBAC, ours below concentrates on the essentials in a manner to suit our purpose. Given are the types U, R, O and Op, denoting, respectively, the sets of all possible users, roles, objects and operations applicable on objects. The elements of these types are typically referred to here as $u \in U, r \in R, o \in O$ and $op \in Op$, with subscripts to distinguish between elements of the same type. Following the conventional practice, we introduce a set P for permissions, but define it as a suitably chosen subset of $O \times Op$, rather than as the full Cartesian product. A precise definition of P will follow later on in Section 4.5. We introduce a relation UA (user-to-role assignment) from U to R

$$UA : U \leftrightarrow R \tag{5}$$

such that, for all $u \in U$ and $r \in R$, $(u, r) \in UA$ holds true if and only if the user u is assigned the role r. Let us also introduce a

[3] In the literature (e.g. [7]), security levels often take into account the notion of *compartments*, alongside that of a security classification. Compartments are typically used as a means of implementing the '*need to know*' principle—a principle that we do not consider in this work, hence our decision not to consider compartments and equate security levels with security classifications. As indicated in Section 8, the consideration of compartments is a possible avenue of future work.

relation *PA* (role-to-permission assignment)

$$PA : R \leftrightarrow P \tag{6}$$

from R to P such that, for all $r \in R$, $PA(r) \neq \emptyset$ and,[4] for all $p \in P$, $(r, p) \in PA$ holds true if and only if the role r is *explicitly*[5] assigned the permission p. The relations *UA* and *PA* will be appropriately constrained in later sections in order to conveniently suit the purposes of our approach. Users enjoy clearance levels in the same way as subjects do in the BLP model.[6] In this respect, the mappings f_S, f_C of Section 3.1 may apply to users too.[7] A formal treatment of the concept of RBAC sessions is deferred until Section 6.2.

4. CONFIDENTIALITY-DRIVEN PERMISSION INHERITANCE

This section introduces the concept of permission capacity in MLS contexts and demonstrates how it can be used to determine the seniority level of a permission. A hierarchical ordering on permissions, one that is based on permission capacity, is formally defined. A novel view of permissions—one which is akin to methods in the object-oriented paradigm—is adopted.

4.1. Permission capacity

In RBAC, a permission is regarded as an '*approval to perform an operation on one or more* RBAC-*protected objects*' [5]. Depending on the importance of the objects with which it is associated, a permission may be assumed to carry a certain *capacity*: the more highly an object is valued, the greater the capacity of the permission. For example in a military setting, a permission to issue a command to launch an airstrike carries a greater capacity than a permission to merely compile an intelligence report: the former grants access to a data object[8] which is naturally considered of far greater significance, hence more highly valued than the former.

In the realm of MAC, and specifically in the BLP model, permission capacity can be quantified on the basis of the security levels assigned to objects. It may thus form the basis for defining *security levels* for permissions and, consequently, a *permission seniority* relation. Nevertheless, a brief account of the effect of operations on objects is first in order.

4.2. Object operations

At the level of abstraction where security issues are addressed from the perspective of confidentiality using the BLP model, operations are considered not with their full functionality but based on their eventual effect on the actual state of a set of affected objects. In such models, operations are thus abstractly categorized into those with observational-only capabilities ('`read`' operations) and those with alterational-only capabilities ('`append`' operations). However, although this categorization is adequate from a confidentiality or integrity point of view, it proves too coarse-grained in cases when these fundamental security properties are not the only concern: it may often be the case that *SoD* constraints must be *additionally* imposed in order to prevent certain sensitive combinations of operations from being performed by the same user. As an example, consider in a military context the operations arm_missile and launch_missile. Although these operations may be indistinguishable at a certain level of abstraction from the eventual effect they have on the state of the affected object (i.e. they may both be categorized as alterational-only operations), it may be required that they are not delegated to the same user. Such a SoD requirement, however, cannot be expressed unless the richer functional capabilities of the two operations are considered as in the RBAC model.

4.3. An object-oriented perspective

In both the MAC and RBAC models, objects as well as operations are uniformly treated in isolation, i.e. as independent entities, as if each operation can be applied to every object irrespective of the kind of the object concerned or the nature of the operation. This view may be adequate when dealing with security concerns in a preliminary manner, but it is hardly a realistic view when considering design issues in detail. In order to remove this anomaly, let us group together objects that share common operations. This is akin to grouping 'objects' into 'classes' in the object-oriented approach. Fitting in with such a paradigm that is widely used in the design and implementation of secure systems is an added motivation for grouping objects in this manner. However, no claim is made here that it is a fully fledged adoption of the object-oriented approach, as this is beyond the scope of this paper.

Let us introduce two equivalence relations,[9] R_C on O and R_M on Op, respectively,[10] such that, given any two objects o_1 and o_2, and given any two operations op_1 and op_2, $(o_1, o_2) \in R_C$ is true iff o_1 and o_2 belong to the same class of objects, and $(op_1, op_2) \in$

[4] We exclude here trivial cases of roles that do not entail any permissions. To reduce notational clutter, we refrain ourselves from using the notation $PA(\!|\{r\}|\!)$ to denote the relational image of *PA* through the set $\{r\}$; instead, we resort to the functional notation $PA(r)$ for denoting the same image; this convention is used throughout this paper.

[5] *Implicit* permission-to-role assignments will be formally introduced in Section 5.3.

[6] In fact, the clearance level of a subject is derived from the clearance level of the user on whose behalf the subject acts.

[7] To reduce notational clutter, we overload the symbols f_S, f_C such that each denotes two 'sibling' mappings: one from the set of all subjects S to L (see Section 3.1), and one from the set of all users U to L.

[8] It is assumed here that issuing a command to launch an airstrike programmatically amounts to altering the value of a data object, e.g. a system attribute, or variable.

[9] Note that our choice of equivalence relations here is purely for grouping objects sharing common operations, though there can be other equivalence relations that can capture other common characteristics.

[10] C indicating the notion of 'class' and M the notion of 'method'.

R_M is true iff op_1 and op_2 are methods applicable to objects in the same class. The partition induced by R_C in O thus represents the set of 'classes' (strictly speaking a set of sets of objects), with each element c in the quotient $O\backslash R_C$ being interpreted as a class c. Likewise, the partition induced by R_M in Op represents the set of methods (strictly speaking a set of sets of operations), with any element m in the quotient $Op\backslash R_M$ being the set of 'methods' belonging to a specific class. We require that $O\backslash R_C$ and $Op\backslash R_M$ are equinumerous and that there exists a bijective function $\mathcal{M}$

$$\mathcal{M} : O\backslash R_C \rightarrowtail\!\!\!\!\!\rightarrow Op\backslash R_M \tag{7}$$

such that, for any $c \in O\backslash R_C$, $\mathcal{M}(c)$ represents the set of methods applicable to objects in class c.

As an example, in a military context, all report data objects (e.g. intelligence reports, compilations of intelligence reports, 'recommendation of action' reports etc.) may be considered to belong to a single class c with $\mathcal{M}(c)$ representing the set of operations defined on these objects (e.g. CRUD[11] operations).

4.4. Permissions

In this work, we choose to view permissions as *atomic* in the sense that each permission can only approve a *single* operation on a *single* object.[12] Formally, the set of relevant permissions is defined as

$$P = \{(o, op) \mid \exists c \in O\backslash R_C; m \in Op\backslash R_M \bullet \mathcal{M}(c) = m \wedge o \in c \wedge op \in m\} \tag{8}$$

It is to be noted here that the fact that two or more objects belong to the same class, and are thus amenable to the same set of operations, does not necessarily imply that the objects bear the same security level. Consider, for example, the case of two database objects: although they may belong to the same class, they may store data of varying classification—a fact that is inevitably reflected by the objects' security level. Objects of the same class may thus potentially give rise to permissions of differing capacities.

It is also to be noted here that the introduction of atomic permissions does not impose an actual restriction on the number of objects on which a subject may gain approval to operate: a subject can always operate on a multitude of objects (from the same or from different classes) by obtaining all the corresponding individual permissions. Our choice to view permissions as atomic can be justified on the following grounds. Firstly, it facilitates the derivation of the capacity of a permission (see Section 4.5) and thus allows for a neater treatment of permission seniorities. Secondly, it extends the object-oriented perspective to permissions: atomic permissions are inevitably bundled with individual object classes in the same manner as operations are bundled with individual object classes. This is in contrast to the conventional RBAC view whereby a single permission may approve a multitude of operations on objects irrespective of their type or characteristics.

We provide the following mapping to determine the eventual effect that a permission has on the state of the affected object:

$$\mathcal{A} : Op \rightarrow (\mathbb{P}A \setminus \emptyset) \tag{9}$$

where the set A was defined in Section 3.1. Note that an operation always has a certain mode of access to the object concerned (as understood in Section 3.1 with or without leaving an effect on the object's state), hence the exclusion of the empty set from the range of $\mathcal{A}$. For example, for any $op \in Op$, $\mathcal{A}(op) = \{ap\}$ means that op alters, but does not observe, the object with which it is associated; similarly, $\mathcal{A}(op) = \{rd\}$ means that op observes, but does not alter, the associated object and $\mathcal{A}(op) = \{ap, rd\}$ means that op alters and observes the associated object.

4.5. Permission security levels

We propose that the security level of a permission be defined by its *capacity*, i.e. by the security level of the object that the permission is associated with. Formally, we have the following definition.

DEFINITION 4.1. $\ell^P : P \rightarrow L$ *such that*

$$\forall p \in P; o \in O; op \in Op \bullet p = (o, op) \Rightarrow \ell^P(p) = f_O(o) \tag{10}$$

We may now proceed with the definition of permission seniority. For any two permissions p and p', p' is considered at least as senior as p (denoted $p \preccurlyeq p'$) iff:

(i) p and p' are defined on objects o, o' of the same security level and p' grants a set of access modes at least as rich as the ones granted by p. For example, in a military setting, a permission that allows a read-only operation on an intelligence report is considered less senior than a permission that grants both `read` and `append` access to such a report.

(ii) p and p' are defined on objects o, o' of distinct security levels, p grants only `read` access to o, p' grants at least `read` access to o' and the security level of p' is *greater* than that of p. For instance, considering again a military context, a permission that grants `read`-only access to an individual intelligence report is considered less senior than a permission that grants `read` access to a compilation of intelligence reports.[13]

(iii) p and p' are defined on objects o, o' of distinct security levels, p grants `append`-only access to o, p' grants at least `append` access to o' and the security level of p' is

[11] Create, Read, Update, Delete.

[12] A similar, albeit less restrictive, view of permissions is adopted in [14].

[13] The compilation of intelligence reports naturally carries a higher security level than a single intelligence report.

less than that of p. For instance, a permission that grants `append`-only access to a compilation of intelligence reports is considered more senior than a permission that grants `append` access to a corresponding 'recommendation of action' report, one that is invariably edited on the basis of the compilation of intelligence reports.[14]

Formally:

Definition 4.2. *Let $p, p' \in P$ such that $p = (o, op)$ and $p' = (o', op')$ for any $o, o' \in O$, $op, op' \in Op$. Then*

$$p \preccurlyeq p' \Leftrightarrow (\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op')) \quad (11)$$
$$\vee$$
$$(\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op')) \quad (12)$$
$$\vee$$
$$(\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op')) \quad (13)$$

Two permissions p and p' are of equal seniority, denoted by $p \approx p'$, iff they share the same security level and they grant an identical set of access modes; formally, we have the following definition.

Definition 4.3. *Let $p, p' \in P$ such that $p = (o, op)$ and $p' = (o', op')$ for any $o, o' \in O$, $op, op' \in Op$. Then*

$$p \approx p' \Leftrightarrow \ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) = \mathcal{A}(op')$$

Note that permission seniority equality does not necessarily imply permission equality (the latter being denoted as $p = p'$).

Theorem 4.1. *The permission seniority relation $(P, \preccurlyeq)$ is a partial order.*

Proof. The proof is provided in Appendix A.1. □

Definition 4.4. *Any two distinct permissions are comparable iff one of them is considered senior, or of equal seniority, to the other.*

Note that our definition of permission seniority constitutes a departure from the one provided by Crampton [14] whereby, for any two permissions p and p', p' is 'senior' to p, iff the former enjoys a richer set of access modes on an underlying object o (see Section 2.1). The reason for this departure is as follows. In our work, permission seniority is introduced as a device that lays the ground for the introduction of *permission inheritance* (see Section 5.3). Let $p = (o, op)$ and $p' = (o, op')$ and suppose that $\mathcal{A}(op) \subset \mathcal{A}(op')$. Given that $A = \{rd, ap\}$ and on the strength of (9), this means that $\mathcal{A}(op) = \{rd\}$ or $\mathcal{A}(op) = \{ap\}$ and that $\mathcal{A}(op') = \{rd, ap\}$. In either case, although p' can be considered senior to p in Crampton's standpoint, such a seniority signifies a superfluous form of permission inheritance: any user who has p' can clearly access o in any access mode, irrespective of any privilege to inheritance, as he is endowed with the full set of access modes A. Crampton's definition of the seniority relation on permissions has thus certain limitations. Our Definition 4.2, captured through predicates (11–13), not only subsumes Crampton's notion of permission seniority, but also incorporates within it the security levels of the objects that the permissions are associated with. Thus, the resulting permission seniority hierarchy is more meaningful as it encompasses more needed and relevant aspects for defining permission seniority in MLS settings.

The seniority relation can be extended to *sets* of permissions: for any two such sets S, S', such that each set comprises solely mutually non-comparable permissions,[15] S' is considered at least as senior as S (denoted by $S \preccurlyeq S'$) iff each permission in S has at least one 'at least as' senior permission in S'. Formally,[16] for any $S \in \mathbb{P}P$, let the predicate $IP(S)$ hold true iff S comprises solely non-comparable permissions:

$$\forall S \in \mathbb{P}P \bullet IP(S) \Leftrightarrow (\forall p, p' \in S \bullet \neg(p \preccurlyeq p') \vee p = p') \quad (14)$$

Then, continuing with the extension of the seniority relation to sets of permissions:

$$\forall S, S' \in (\mathbb{P}P \backslash \emptyset) \bullet S \preccurlyeq S' \Leftrightarrow (IP(S) \wedge IP(S') \wedge \forall p \in S \bullet \exists p' \in S' \bullet p \preccurlyeq p') \quad (15)$$

Similarly, S and S' are considered of equal seniority, denoted by $S \approx S'$, iff each permission in S has at least one permission of equal seniority to it in S' and vice versa; formally,

$$\forall S, S' \in (\mathbb{P}P \backslash \emptyset) \bullet S \approx S' \Leftrightarrow (\forall p \in S \bullet \exists p' \in S' \bullet p \approx p') \wedge (\forall p' \in S' \bullet \exists p \in S \bullet p' \approx p) \quad (16)$$

Any two distinct permission *sets* are *comparable* iff one is considered senior, or of equal seniority, to the other.

[14] The 'recommendation of action' report naturally carries a higher security level than the compilation of intelligence reports.

[15] As will be explained in Section 5.1, roles in our model are defined mathematically as sets of mutually non-comparable permissions; here we are interested in extending the seniority relations to sets of permissions that could potentially be considered roles.

[16] Note that in order to reduce notational clutter, we overload the symbol $\preccurlyeq$ to denote three different types of seniority: seniority between permissions, seniority between sets of permissions and seniority between roles (for the latter type see Section 5.2). The specific type of seniority to which any given instance of $\preccurlyeq$ refers is unambiguously derived from its operands. An analogous note holds for the symbol $\approx$.

4.6. Example

The purpose of this example is to illustrate the usefulness of our model by demonstrating through a realistic application that, under certain circumstances, confidentiality and SoD requirements are interwoven and may hence need to be simultaneously satisfied. In the current section, however, we restrict ourselves to merely illustrating how the concepts of permission capacity and permission seniority apply to a realistic application. The scenario described below is a simplified version of a case study reported in [38]. The Maritime Intelligence Center (MIC) is an organization responsible for bringing together military intelligence operations. Suppose that MIC receives intelligence about a vessel carrying illegal cargo which must be interdicted. The intelligence is received by the Signals Intelligence (SIGINT) and Electronic Intelligence (ELINT) Officers who are responsible for composing two relevant reports, modelled here by the objects denoted by o_{SI} and o_{EI}, respectively (see Fig. 1). The security levels applicable to this application are identified as $c_i, i = 1, \ldots, 4$, and ordered as following:

$$c_4 < c_3 < c_2 < c_1 \tag{17}$$

This total order replaces the more general lattice of security levels typically encountered in BLP models.

We assume that the objects o_{SI} and o_{EI} have the same security level:

$$f_O(o_{SI}) = f_O(o_{EI}) = c_4 \tag{18}$$

We also assume the existence of a Tactical Analyst (TA) Officer who is responsible for compiling the SIGINT and ELINT reports into a single tactical analysis report, modelled here by the object denoted by o_{TA} (see Fig. 1). The security level of o_{TA} equals c_3:

$$f_O(o_{TA}) = c_3 \tag{19}$$

The tactical analysis report is accessible to an Intelligence Watch Officer (IWO) who acts as a linchpin in providing data to his/her superiors and initiating action. More specifically, based on the tactical analysis, the IWO submits an 'interdiction recommendation report' to the Command Duty Officer (CDO) who is responsible for deciding whether (and when) to issue a command for vessel interdiction. We thus discern two additional objects: 'recommendation for vessel interdiction', denoted by o_{IR}, and 'command for vessel interdiction', denoted by o_{IC}. The security levels of these objects are as follows:

$$f_O(o_{IR}) = c_2 \tag{20}$$

$$f_O(o_{IC}) = c_1 \tag{21}$$

All objects of relevance to this example, along with the operations to which they are amenable, the corresponding types of access and the relevant permissions, are summarized in Table 1.[17]

It follows from Definition 4.1 that the capacity, and hence the security level, of each permission becomes:

$$\ell^P(p_1^r) = \ell^P(p_1^a) = c_1 \tag{22}$$

$$\ell^P(p_2^r) = \ell^P(p_2^a) = c_2 \tag{23}$$

$$\ell^P(p_3^r) = \ell^P(p_3^a) = c_3 \tag{24}$$

$$\ell^P(p_4^r) = \ell^P(p_4^a) = \ell^P(p_5^r) = \ell^P(p_5^a) = c_4 \tag{25}$$

As an illustration, consider the operations r_{IR} and r_{IC} in our running example. Both operations grant read-only access, through permissions p_2^r and p_1^r, to objects o_{IR} and o_{IC}, respectively. It follows from predicates (17), (22) and (23) that p_1^r has a greater capacity than p_2^r. From predicate (12) of Definition 4.2 it then follows that p_1^r is senior to p_2^r, i.e. $p_2^r \prec p_1^r$, depicted in Fig. 2 with an arrow from p_2^r to p_1^r.[18] Consider now another pair of operations a_{IC} and a_{IR}, granting append-only access to objects o_{IC} and o_{IR}, respectively, through permissions p_1^a and p_2^a. It follows from (13) of Definition 4.2 and, as in the previous case, from (17), (22) and (23) that $p_1^a \prec p_2^a$. This corresponds to an arrow from p_1^a to p_2^a in Fig. 2. The reader may note that there are two disjoint sub-graphs in the graph of Fig. 2 and, further, that one sub-graph corresponds to 'append-only' permissions p_i^a, while the other to 'read-only' permissions p_j^r, for $i, j = 1 \ldots 5$. This is because, given any two permissions $p_i^r = (o_i, op_m)$ and $p_j^a = (o_j, op_n)$, $\mathcal{A}(op_m) \cap \mathcal{A}(op_n) = \emptyset$. This means that, according to Definition 4.2, p_i^r and p_j^a are not related through $\preccurlyeq$, hence the absence of arrows between them.

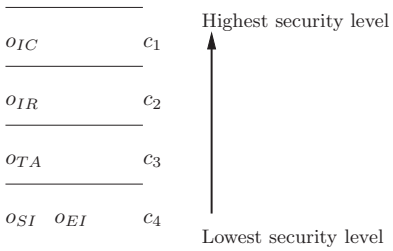

FIGURE 1. Security levels of objects.

[17] All permissions used in this example are either `read`-only or `append`-only (this is, respectively, indicated by the superscripts r and a with which permissions are decorated). Clearly, any two permissions—such as, for example, p_1^r and p_1^a, that grant `read`-only and `append`-only access to the *same* object, have the same effect as a single permission that grants both `read` and `append` access to that object. Nevertheless, as will become apparent in later sections, by discerning `read`-only and `append`-only permissions, we are able to exercise finer control as to which specific operations a role provides access—a feature particularly relevant when SoD constraints are considered. This is the reason why no permissions that provide combined `read` and `append` access are illustrated in this example. However, it is to be noted that, for the sake of generality, our definition of the eventual effect that a permission has on the state of the affected object (see Definition (9) of mapping $\mathcal{A}$), and hence the permission seniority relation of Definition 4.2, are not restricted to `read`-only and `append`-only permissions but also take into account permissions that provide combined `read` and `append` access to objects.

[18] The following convention is used in Fig. 2: for any two permissions p, p', there exists an arrow from p to p' iff p' is at least as senior as p.

TABLE 1. Objects, operations and permissions.

Objects		Operations			
Symbol	Description	Symbol	Description	Type of access	Permissions
o_{IC}	vessel interdiction command	r_{IC}	read [command]	$\mathcal{A}(r_{IC}) = \{rd\}$	$p_1^r = (o_{IC}, r_{IC})$
		a_{IC}	alter/issue [command]	$\mathcal{A}(a_{IC}) = \{ap\}$	$p_1^a = (o_{IC}, a_{IC})$
o_{IR}	vessel interdiction recommendation	r_{IR}	read [recommendation]	$\mathcal{A}(r_{IR}) = \{rd\}$	$p_2^r = (o_{IR}, r_{IR})$
		a_{IR}	alter/create [recommendation]	$\mathcal{A}(a_{IR}) = \{ap\}$	$p_2^a = (o_{IR}, a_{IR})$
o_{TA}	tactical analysis	r_{TA}	read [analysis]	$\mathcal{A}(r_{TA}) = \{rd\}$	$p_3^r = (o_{TA}, r_{TA})$
		a_{TA}	alter/create [analysis]	$\mathcal{A}(a_{TA}) = \{ap\}$	$p_3^a = (o_{TA}, a_{TA})$
o_{SI}	SIGINT report	r_{SI}	read [report]	$\mathcal{A}(r_{SI}) = \{rd\}$	$p_4^r = (o_{SI}, r_{SI})$
		a_{SI}	alter/create [report]	$\mathcal{A}(a_{SI}) = \{ap\}$	$p_4^a = (o_{SI}, a_{SI})$
o_{EI}	ELINT report	r_{EI}	read [report]	$\mathcal{A}(r_{EI}) = \{rd\}$	$p_5^r = (o_{EI}, r_{EI})$
		a_{EI}	alter/create [report]	$\mathcal{A}(a_{EI}) = \{ap\}$	$p_5^a = (o_{EI}, a_{EI})$

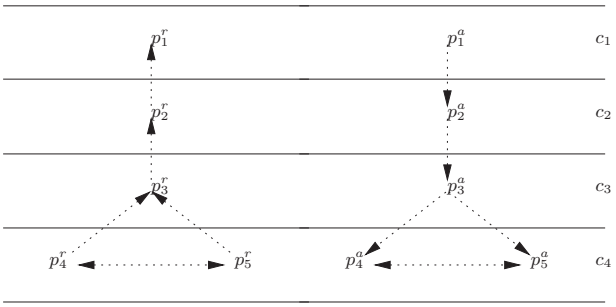

FIGURE 2. Permission seniorities.

The opposite directions of the arrows in the two sub-graphs follow as a consequence of the fact that seniority in 'read-only' permissions increases with increasing permission capacity, whereas seniority in 'append-only' permissions decreases with increasing permission capacity—this is formally captured through predicates (12) and (13) of Definition 4.2.

5. SOD AND EXPLICIT PERMISSION-TO-ROLE ASSIGNMENT

This section begins with an introduction to *explicit* assignment of permissions to roles. It then defines a hierarchical ordering on roles and proposes a formal framework for the *implicit* assignment of permissions to roles. Section 5.4 introduces conflicting permissions from a SoD perspective and Section 5.5 investigates how such permissions affect permission inheritance and thus implicit permission-to-role assignments. A relevant example is presented in Section 5.6 and, in Section 5.7, the notion of mutually exclusive roles is formally defined.

5.1. Explicit assignment of permissions to roles

Mathematically, a role is a named collection of permissions [16]. This provides a sound basis for determining the security level of a role from the security levels, and thus the capacities, of the permissions assigned to it. We require that such permissions are all of the same security level. Formally, we have the following definition.

DEFINITION 5.1. $\ell^R : R \to L$ *such that*

$$\forall r \in R; p, p' \in P \bullet \{p, p'\} \subseteq PA(r) \Rightarrow \ell^P(p') = \ell^P(p) = \ell^R(r) \tag{26}$$

As will become clear later from Definition 5.3 on implicit inheritance of junior permissions, it is futile to explicitly assign two or more comparable permissions to a role. In the interest of eliminating such futile assignments from our model, we impose the following restriction:

$$\forall r \in R; p, p' \in P \bullet p, p' \in PA(r) \Rightarrow \neg(p \preccurlyeq p') \vee p = p' \tag{27}$$

Note that, by virtue of Definition 4.2, any two permissions that both grant (at least) a `read` or an `append` access are comparable. A role can thus only be assigned a *single* permission that grants a `read`-only access, and/or a *single* permission that grants an `append`-only access.[19] This is analogous to the approach taken in [13] where exactly two roles reside at each security level x: one for reading (xR) and one for writing (xW) at level x. Formally, we have the following corollary.

[19] Of course, if a role is assigned a permission that grants both a `read` and an `append` access, then that role cannot be explicitly assigned any other permissions.

COROLLARY 5.1. *Let $p, p' \in P$ such that $p = (o, op)$ and $p' = (o', op')$ for any $o, o' \in O$, $op, op' \in Op$. Then,*

$$\forall r \in R \bullet \#PA(r) \leq 2 \wedge (p, p' \in PA(r) \wedge p \neq p' \Rightarrow \#\mathcal{A}(op) = \#\mathcal{A}(op') = 1 \wedge \mathcal{A}(op) \neq \mathcal{A}(op'))$$

Proof. The proof is provided in Appendix A.2. □

Note that in our model, a multitude of roles may coexist at the same security level. This constitutes a departure from the approach in [13] and, as already mentioned, it is a prerequisite for the incorporation of SoD constraints. Also, a role in our model may only comprise a single permission that grants a `read`-only or an `append`-only access. This also constitutes a departure from the approach in [13] whereby each role necessarily has read and append access to all objects that reside at a certain security level. Our approach is more general as it allows, for example, cases in which one or more objects are read-only and are thus not amenable to alteration.

5.2. Role hierarchy

The hierarchical model of RBAC is founded on a role hierarchy that mirrors the organizational role hierarchy. Mathematically, it is based on a partial order on the set of applicable roles. In our case, any hierarchical ordering between two (or more) roles is naturally based on the corresponding sets of permissions that these roles make available: the seniority of a role is directly proportional to the seniority, and thus the capacity, of the permissions that it entails. It follows that, for any two roles r_1 and r_2, r_2 is at least as senior as r_1 (denoted by $r_1 \preccurlyeq r_2$) iff the security level of r_1 is less than, or equal to, the security level of r_2. Formally, we have the following definition.

DEFINITION 5.2.

$$\forall r_1, r_2 \in R \bullet r_1 \preccurlyeq r_2 \Leftrightarrow \ell^R(r_1) \leq \ell^R(r_2)$$

Two roles r_1 and r_2 are of equal seniority, denoted by $r_1 \approx r_2$, iff their corresponding sets of permissions are of equal seniority and thus have the same capacity.

COROLLARY 5.2. *The role seniority relation $(R, \preccurlyeq)$ is a partial order.*

Proof. The proof follows directly from the fact that $(L, \leq)$ is a partial order. □

Note that role seniority equality does not necessarily imply role equality (the latter being denoted as $r = r'$).

In hierarchical RBAC role seniority implies role inheritance in the sense that senior roles acquire the permissions of junior roles [5]. This, however, is not generally the case in MLS systems as the following example demonstrates.

5.2.1. *Example*

Continuing the example of Section 4.6, consider the roles *IWO* and *CDO* and assume that they are explicitly assigned the following permissions:

$$PA(CDO) = \{p_1^r, p_1^a\} \quad (28)$$

$$PA(IWO) = \{p_2^r, p_2^a\} \quad (29)$$

Note that (28) constitutes a valid permission-to-role assignment since, according to (22), p_1^r and p_1^a have the same security level and are also incomparable with each other. An analogous argument holds for (29). *CDO* enjoys a greater security level than *IWO* (since $\ell^P(p_2^r) < \ell^P(p_1^r)$) and thus, according to Definition 5.2, *CDO* is more senior than *IWO*. Nevertheless, *CDO* cannot acquire p_2^a for this would violate the 'no write down' policy leading to a potential confidentiality breach.

As will be seen in Section 5.3, role inheritance is, in our model, being replaced by a more fine-grained *direct permission inheritance* whereby *individual* permissions may be inherited by a role r irrespective of whether these permissions are assigned to any roles junior to r. Such a direct permission inheritance avoids potential confidentiality breaches such as the one outlined above.

5.2.2. *Role seniority—an alternative formulation*

An alternative definition of role seniority, one which by design avoids confidentiality breaches, can be constructed as follows. For any two roles r_1 and r_2, r_2 is at least as senior as r_1 (denoted $r_1 \preccurlyeq r_2$) iff each and every permission that r_1 entails has at least one 'at least as' senior permission entailed by r_2. Formally[20]:

$$\forall r_1, r_2 \in R \bullet r_1 \preccurlyeq r_2 \Leftrightarrow PA(r_1) \preccurlyeq PA(r_2). \quad (30)$$

Such a formulation does imply role inheritance as in hierarchical RBAC for it can be formally shown that, under predicate (30), senior roles do acquire the permissions of junior roles. In addition, it can be shown (see relevant theorem in Appendix A.3) that such a role seniority relation is a partial order.

Nevertheless, as the following example illustrates, such a formulation may be incompatible with the view of role hierarchy in MLS systems.

5.2.3. *Example*

Consider the roles *CDO* and *IWO*. Although *CDO* is, from a military standpoint, senior to *IWO*, the set $PA(CDO)$ cannot be considered senior to $PA(IWO)$ as it contains at least one permission (p_1^a) which is less senior than p_2^a contained in $PA(IWO)$. Thus *CDO* cannot be considered senior to *IWO* in the sense of predicate (30).

Generalizing, this view of role seniority is of limited use for performing role comparisons in MLS settings: in such settings

[20] Recall that the permission seniority relation is extended to sets of permissions through predicates (15) and (16).

a role will usually comprise read and append permissions to objects of a security level equal to the role's security level; such permissions cannot both be junior or, for that matter, senior to the read and append permissions comprised by any other role of a different security level.

In this work, we choose to retain Definition 5.2 for role seniority which accurately depicts the organizational view of role hierarchy and ignore the alternative formulation provided by predicate (30).

5.3. Implicit role permissions

The set of permissions allocated to a given role are defined hitherto solely on the basis of explicit permission-to-role assignments as described in Section 5.1. Here we extend it to additionally take into account the inheritance of junior permissions: if a role r is assigned a permission p, then r is also *implicitly* assigned any permission p' for which $p' \preccurlyeq p$ holds. In order to accommodate this requirement, we introduce a relation $PA^{\preccurlyeq} : R \leftrightarrow P$ which forms a superset of PA. Formally, we have the following definition.

Definition 5.3.

$$PA^{\preccurlyeq} : R \leftrightarrow P$$
$$\forall r \in R; p, p' \in P \bullet p \preccurlyeq p' \wedge p' \in PA(r) \Rightarrow p \in PA^{\preccurlyeq}(r) \tag{31}$$
$$\forall r \in R; p \in P \bullet p \in PA^{\preccurlyeq}(r) \Rightarrow p \in PA(r) \vee \exists p' \in PA(r) \bullet p \preccurlyeq p' \tag{32}$$

The predicate (32) requires that $PA^{\preccurlyeq}$ is set-theoretically the smallest superset of PA satisfying (31). A distinctive feature of our model is that every role r inherits implicitly all permissions that are junior to the permissions explicitly assigned to r. We refer to this as *direct permission inheritance*—a notion that fundamentally differs from RBAC's conventional view of permission inheritance whereby r can only inherit permissions that have been assigned to roles junior to r. Direct permission inheritance is not definable in conventional RBAC as the latter lacks any hierarchical ordering of permissions, similar to that defined here on the basis of permission capacity.

Note that direct permission inheritance is the reason for requiring in Section 5.1 that no two comparable permissions be *explicitly* assigned to the same role—see predicate (27). Suppose, for instance, that the permissions p, p' are explicitly assigned to the role r, and let p' be comparable to p. Then, by Definition 4.4, either $p \preccurlyeq p'$ or $p' \preccurlyeq p$. In either case, the more junior permission is assigned to r implicitly, by virtue of direct permission inheritance, rendering any explicit assignment of comparable permissions to the same role pointless.

However, there may be situations where full direct permission inheritance may not be desirable. In order to accommodate this requirement, we introduce a relation $PA^{\preccurlyeq}_{Excl} : R \leftrightarrow P$ designed to exclude inheritance of any undesirable permissions. More specifically, consider the relation $PExcl : R \leftrightarrow P$. For any role r, $PExcl(r)$ comprises all permissions that must be excluded from among those directly (implicitly) inherited by r. We may then define $PA^{\preccurlyeq}_{Excl}$ as:

Definition 5.4.

$$PA^{\preccurlyeq}_{Excl} : R \leftrightarrow P$$
$$\forall r \in R \bullet PA^{\preccurlyeq}_{Excl}(r) = PA^{\preccurlyeq}(r) \setminus PExcl(r) \tag{33}$$

The task of determining which permissions comprise $PExcl(r)$, i.e. which permissions (if any) must be excluded from a role r is, of course, an application-specific one and, therefore, is not dealt with at the current level of abstraction.

5.3.1. *Example*

Consider the role TA from the example of Section 4.6 and suppose the following explicit permission assignment:

$$PA(TA) = \{p^r_3, p^a_3\} \tag{34}$$

Taking into account inheritance of junior permissions, TA is implicitly assigned the permissions p^a_1, p^a_2, p^r_4 and p^r_5 (see Fig. 2 for the relevant permission seniority relations):

$$PA^{\preccurlyeq}(TA) = \{p^r_3, p^a_3, p^a_1, p^a_2, p^r_4, p^r_5\} \tag{35}$$

From Definition 5.1 it follows that TA has a security level equal to c_3. Suppose now that TA is disallowed to alter the objects o_{IC} and o_{IR} (recall that these objects reside at higher security levels than the security level of TA—we refer the reader to Table 1 for more details). Consequently, the permissions p^a_1 and p^a_2 belong to the set $PExcl(r)$ and must therefore be excluded from the set $PA^{\preccurlyeq}(TA)$:

$$PA^{\preccurlyeq}_{Excl}(TA) = \{p^r_3, p^a_3, p^r_4, p^r_5\} \tag{36}$$

5.4. SoD and conflicting permissions

SoD is a fundamental principle in computer security [3]. It states that if a sensitive task requires n distinct operations for its completion, k (for $k < n$) of these operations—and hence the corresponding *permissions*—are considered *conflicting* and must be performed by different users.

One approach to enforce an SoD policy is through static SoD (SSoD) policies. As reported in [25], each SSoD policy is an *objective* that specifies the minimum number of users that are allowed to collectively possess all permissions for the completion of a task. Such an objective is typically enforced in RBAC, and thus in this work too, via mutual exclusion constraints that limit the role membership of users and ultimately ensure adequate permission-to-user assignments. In [25], such constraints are termed SMER constraints. Although SMER constraints are

suitable for enforcing SoD policies, the same cannot be claimed about a second kind of constraints, the so-called dynamic mutually exclusive role (DMER) constraints [25]. The reason for this is that DMER constraints do not limit the role membership of users but, instead, exclude mutually exclusive roles from RBAC sessions; as discussed in Section 6.2, this is not appropriate for the enforcement of SoD policies. Hence we limit ourselves to SMER constraints.

The ability of SMER constraints to successfully enforce an SSoD policy crucially depends upon how permissions are assigned to roles: for instance, conflicting permissions may never be assigned to the same role, for otherwise they may be possessed by a single user. Let us introduce the relation *PConf* to formally capture conflicting permissions:

$$PConf : P \leftrightarrow P \tag{37}$$

For any $p \in P$, the set $PConf(p)$ contains exactly those permissions that are in conflict with p. It is assumed here that two permissions are considered conflicting iff their corresponding operations are considered conflicting.[21] Clearly, *PConf* is an irreflexive and symmetric relation:

$$\forall p \in P \bullet p \notin PConf(p) \tag{38}$$

$$\forall p, p' \in P \bullet p \in PConf(p') \Rightarrow p' \in PConf(p) \tag{39}$$

Crampton [12] claims that the existence of a role hierarchy facilitates the specification of SoD constraints because it reflects the structure of the underlying organization. We believe that the existence of a permission hierarchy, such as the one proposed in this work, further reinforces this facilitation by allowing SMER constraints to be derived from corresponding static mutually exclusive *permission* constraints. It is also to be noted here that in our approach, permission conflicts are *task-agnostic*; i.e. any two permissions—and hence the operations that they entail—are considered conflicting independently of the task as part of which these operations are being performed. This is in line with BLP where there is generally no notion of a task and any restrictions are placed directly on individual actions.

5.5. Mutually exclusive permissions and inheritance

The introduction of conflicting permissions naturally affects the manner in which permissions are allocated to roles. We are interested in ensuring that any two conflicting permissions are *mutually exclusive*, i.e. that they cannot be made available through the same role. The permissions (explicitly or implicitly) allocated to a role must thus not be conflicting. To accommodate this requirement, we introduce the function $\mathcal{CFP}^{\preccurlyeq}$ which associates with any role r the sets of permissions drawn from the powerset $\mathbb{P}PA^{\preccurlyeq}_{Excl}(r)$ that do not contain any conflicting permissions. More specifically, $\mathcal{CFP}^{\preccurlyeq}(r)$ comprises all those elements of $\mathbb{P}PA^{\preccurlyeq}_{Excl}(r)$ that are set-theoretically the largest subsets of $PA^{\preccurlyeq}_{Excl}(r)$ that comprise solely non-conflicting permissions. Formally, we have the following definition.

DEFINITION 5.5. *$\mathcal{CFP}^{\preccurlyeq} : R \to \mathbb{P}\mathbb{P}P$ such that, for any $r \in R$,*

$$\begin{aligned}&\mathcal{CFP}^{\preccurlyeq}(r) = \\ &\{S \in \mathbb{P}PA^{\preccurlyeq}_{Excl}(r) \mid \forall p \in PA^{\preccurlyeq}_{Excl}(r) \bullet \\ &p \in S \Leftrightarrow \forall p' \in S \bullet (p, p') \notin PConf\}\end{aligned} \tag{40}$$

For any role r, the sets of permissions that are made available through r is drawn from the set $\mathcal{CFP}^{\preccurlyeq}(r)$. These are essentially all the permissions explicitly or implicitly assigned to r after the subtraction of any conflicting permissions. In the case of there being two or more distinct elements in $\mathcal{CFP}^{\preccurlyeq}(r)$, the one that will be used for specifying the permissions entailed by r must include the set $PA(r)$, i.e. the complete set of permissions *explicitly* assigned to r. This is because the inheritance of junior permissions by a role should by no means undermine the role's ability to perform the core operations explicitly assigned to it. Of course, this is only possible if there are no conflicting permissions in $PA(r)$. To satisfy this requirement, let us introduce the set $\mathcal{IEP}$ (stands for 'incomplete explicit permissions') which:

(1) equals the empty set if the set of permissions explicitly assigned to r contains any conflicting permissions;
(2) comprises exactly those elements of $\mathcal{CFP}^{\preccurlyeq}(r)$ that do *not* include $PA(r)$, otherwise.

Formally, we have the following definition.

DEFINITION 5.6. *$\mathcal{IEP} : R \to \mathbb{P}\mathbb{P}P$ such that, for any $r \in R$,*

$$\begin{aligned}\mathcal{IEP}(r) &= \emptyset, \quad \text{if } \exists p, p' \in PA(r) \bullet (p, p') \in PConf \\ &= \{S \in \mathcal{CFP}^{\preccurlyeq}(r) \mid PA(r) \not\subseteq S\}, \quad \text{otherwise.}\end{aligned}$$

The role of the function $\mathcal{IEP}$ in determining an appropriate set of permissions is further demonstrated through the example of Section 5.6.

In case there are two or more elements in $\mathcal{CFP}^{\preccurlyeq}(r)$ that subsume $PA(r)$, the choice as to which one is selected for specifying the permissions entailed by r is non-deterministic and dealt with at an application-specific level by a security administrator. Similarly, in case there are no elements in $\mathcal{CFP}^{\preccurlyeq}(r)$ that subsume $PA(r)$, some element of $\mathcal{CFP}^{\preccurlyeq}(r)$ is non-deterministically chosen for defining the permissions allocated to r. Consequently, at the current level of abstraction, we merely assert that the set of permissions assigned to r is drawn from $\mathcal{CFP}^{\preccurlyeq}(r) \setminus \mathcal{IEP}(r)$. In this respect, we introduce the relation $PA^{\preccurlyeq}_{SoD}$ to associate with each role an appropriate set of non-conflicting permissions; formally, we have the following definition.

DEFINITION 5.7. *$PA^{\preccurlyeq}_{SoD} : R \leftrightarrow P$ such that*

$$\forall r \in R \bullet PA^{\preccurlyeq}_{SoD}(r) \in \mathcal{CFP}^{\preccurlyeq}(r) \setminus \mathcal{IEP}(r) \tag{41}$$

[21] Recall from Section 4.4 that permissions in our work are atomic and thus each permission entails a single operation.

The case of $\mathcal{CFP}^{\preccurlyeq}(r)$ being a singleton set signifies that $PA^{\preccurlyeq}_{Excl}(r)$ contains no conflicting permissions in which case $\mathcal{CFP}^{\preccurlyeq}(r)$ clearly encompasses $PA^{\preccurlyeq}_{Excl}(r)$.

Note that it is impossible for the empty set to be a member of $\mathcal{CFP}^{\preccurlyeq}(r)$. This is because, by definition (see Section 3.2), each role is assigned a non-empty set of permissions and in the event of there being any conflicting permissions among them, $\mathcal{CFP}(r)$ is made up of permissions assigned to r by removing just one permission at a time from each conflicting pair of permissions. Therefore, elements in $\mathcal{CFP}^{\preccurlyeq}(r)$ always contain at least one permission. Formally, we have the following corollary.

Corollary 5.3. $\forall r \in R \bullet PA^{\preccurlyeq}_{SoD}(r) \neq \emptyset$.

Proof. The proof follows directly from Definitions 5.5 and 5.7. □

Finally, note that SoD-derived mutual exclusion constraints are substantially different from the permission exclusion restrictions captured through the *PExcl* relation of Section 5.3. The latter merely specify sets of permissions that are beyond the scope of a role and must thus be excluded from the permissions implicitly assigned to it. The former specify sets of permissions that are excluded from the permissions assigned to a role, not because they are beyond the role's scope, but because they cannot coexist with other permissions that the role entails.

5.6. Example

Continuing the example of Section 4.6, consider the roles *IWO*, *CDO*, *TA*, *SIGINT*, *ELINT* and assume that they are explicitly assigned the following permissions (see Table 1 for the relevant permission definitions):

$$PA(CDO) = \{p^r_1, p^a_1\} \quad (42)$$
$$PA(IWO) = \{p^r_2, p^a_2\} \quad (43)$$
$$PA(TA) = \{p^r_3, p^a_3\} \quad (44)$$
$$PA(SIGINT) = \{p^r_4, p^a_4\} \quad (45)$$
$$PA(ELINT) = \{p^r_5, p^a_5\} \quad (46)$$

From Definition 5.1 it follows that these roles have the following security levels:

$$\ell^R(CDO) = c_1 \quad (47)$$
$$\ell^R(IWO) = c_2 \quad (48)$$
$$\ell^R(TA) = c_3 \quad (49)$$
$$\ell^R(SIGINT) = \ell^R(ELINT) = c_4 \quad (50)$$

Taking into account inheritance of junior permissions, each role is implicitly assigned the following additional permissions (see Fig. 2 for the relevant permission seniority relations):

$$PA^{\preccurlyeq}(CDO) = \{p^r_1, p^a_1, p^r_2, p^r_3, p^r_4, p^r_5\} \quad (51)$$
$$PA^{\preccurlyeq}(IWO) = \{p^r_2, p^a_2, p^a_1, p^r_3, p^r_4, p^r_5\} \quad (52)$$
$$PA^{\preccurlyeq}(TA) = \{p^r_3, p^a_3, p^a_1, p^a_2, p^r_4, p^r_5\} \quad (53)$$
$$PA^{\preccurlyeq}(SIGINT) = PA^{\preccurlyeq}(ELINT) = \{p^r_4, p^a_4, p^a_1, p^a_2, p^a_3, p^a_5, p^r_5\} \quad (54)$$

Assume the following permission–exclusion requirements:

$$PExcl(CDO) = \emptyset \quad (55)$$
$$PExcl(IWO) = \emptyset \quad (56)$$
$$PExcl(TA) = \{p^a_1, p^a_2\} \quad (57)$$
$$PExcl(SIGINT) = PExcl(ELINT) = \{p^a_1, p^a_2, p^a_3\} \quad (58)$$

Taking into account the above permission exclusions, each role is implicitly assigned the following permissions:

$$PA^{\preccurlyeq}_{Excl}(CDO) = \{p^r_1, p^a_1, p^r_2, p^r_3, p^r_4, p^r_5\} \quad (59)$$
$$PA^{\preccurlyeq}_{Excl}(IWO) = \{p^r_2, p^a_2, p^a_1, p^r_3, p^r_4, p^r_5\} \quad (60)$$
$$PA^{\preccurlyeq}_{Excl}(TA) = \{p^r_3, p^a_3, p^r_4, p^r_5\} \quad (61)$$
$$PA^{\preccurlyeq}_{Excl}(SIGINT) = PA^{\preccurlyeq}(ELINT) = \{p^r_4, p^a_4, p^a_5, p^r_5\} \quad (62)$$

Assume now the following SoD constraints:

(1) A subject that composes a recommendation for vessel interdiction (operation a_{IR} of Table 1) must not be able to issue a command for vessel interdiction.
(2) A subject that composes the SIGINT report (operation a_{SI} of Table 1) must not be able to compose an ELINT report.

These constraints give rise to the following permission conflicts:

$$p^a_1 \in PConf(p^a_2) \quad (63)$$
$$p^a_5 \in PConf(p^a_4) \quad (64)$$

We are now ready to apply the function $\mathcal{CFP}^{\preccurlyeq}$ in order to determine the allowable permissions that can be allocated to each role:

$$\mathcal{CFP}^{\preccurlyeq}(CDO) = \{\{p^r_1, p^a_1, p^r_2, p^r_3, p^r_4, p^r_5\}\} \quad (65)$$
$$\mathcal{CFP}^{\preccurlyeq}(IWO) = \{\{p^r_2, p^a_2, p^r_3, p^r_4, p^r_5\}, \{p^r_2, p^a_1, p^r_3, p^r_4, p^r_5\}\} \quad (66)$$
$$\mathcal{CFP}^{\preccurlyeq}(TA) = \{\{p^r_3, p^a_3, p^r_4, p^r_5\}\} \quad (67)$$
$$\mathcal{CFP}^{\preccurlyeq}(SIGINT) = \mathcal{CFP}^{\preccurlyeq}(ELINT) = \{\{p^r_4, p^a_4, p^r_5\}, \{p^r_4, p^a_5, p^r_5\}\}. \quad (68)$$

Finally, for each role r we need to select an appropriate element of $\mathcal{CFP}^{\preccurlyeq}(r)$ which will represent the set of permissions made available through r. Recall from Section 5.5 that such a set of permissions should contain the core permissions that have been *explicitly* assigned to r.[22] To this end, we first apply the function $\mathcal{IEP}$ to r in order to discern all those elements

[22] Providing, of course, that the permissions explicitly allocated to r do not comprise any conflicting permissions.

of $\mathcal{CFP}^{\preccurlyeq}(r)$ that do *not* include the full set of permissions explicitly assigned to r. By removing these elements from $\mathcal{CFP}^{\preccurlyeq}(r)$, we ensure that the element of $\mathcal{CFP}^{\preccurlyeq}(r)$ that will be ultimately chosen to represent the set of permissions offered through r necessarily contains all those permissions explicitly assigned to r.[23] Applying $\mathcal{IEP}$ to the roles of this example thus yields:

$$\mathcal{IEP}^{\preccurlyeq}(CDO) = \emptyset \tag{69}$$

$$\mathcal{IEP}^{\preccurlyeq}(IWO) = \{\{p_2^r, p_1^a, p_3^r, p_4^r, p_5^r\}\} \tag{70}$$

$$\mathcal{IEP}^{\preccurlyeq}(TA) = \emptyset \tag{71}$$

$$\mathcal{IEP}^{\preccurlyeq}(SIGINT) = \{\{p_4^r, p_5^a, p_5^r\}\} \tag{72}$$

$$\mathcal{IEP}^{\preccurlyeq}(ELINT) = \{\{p_4^r, p_4^a, p_5^r\}\} \tag{73}$$

Note that in (69) and (71) above, the application of $\mathcal{IEP}$ yields the empty set not because the permissions explicitly assigned to *CDO* and *TA* contain conflicting permissions, but because $\mathcal{CFP}^{\preccurlyeq}(CDO)$ and $\mathcal{CFP}^{\preccurlyeq}(TA)$ comprise a single element that includes all permissions explicitly assigned to *CDO* and *TA*, respectively.

From the sets of allowable permissions that each role is entitled to, the following final allocations are selected, after substracting the sets of permissions returned by the application of the function $\mathcal{IEP}$:

$$PA_{SoD}^{\preccurlyeq}(CDO) = \{p_1^r, p_1^a, p_2^r, p_3^r, p_4^r, p_5^r\} \tag{74}$$

$$PA_{SoD}^{\preccurlyeq}(IWO) = \{p_2^r, p_2^a, p_3^r, p_4^r, p_5^r\} \tag{75}$$

$$PA_{SoD}^{\preccurlyeq}(TA) = \{p_3^r, p_3^a, p_2^a, p_4^r, p_5^r\} \tag{76}$$

$$PA_{SoD}^{\preccurlyeq}(SIGINT) = \{p_4^r, p_4^a, p_5^r\} \tag{77}$$

$$PA_{SoD}^{\preccurlyeq}(ELINT) = \{p_5^r, p_5^a, p_4^r\} \tag{78}$$

Note that, since all the above roles are explicitly assigned solely non-conflicting permissions, the sets of permissions ultimately made available through them include these explicitly assigned permissions.

5.7. Conflicting and mutually exclusive roles

Two roles are considered *conflicting* iff they entail mutually exclusive permissions. Formally, we devise the mapping *RConf* which, for any role r, returns all those roles that are conflicting with r.

DEFINITION 5.8. *RConf* $: R \leftrightarrow R$ *such that*

$$\forall r, r' \in R \bullet (r, r') \in RConf \Leftrightarrow \exists p \in PA_{SoD}^{\preccurlyeq}(r); p' \in PA_{SoD}^{\preccurlyeq}(r') \bullet (p, p') \in PConf$$

COROLLARY 5.4. *RConf is an irreflexive and symmetric relation:*

$$\forall r \in R \bullet r \notin RConf(r) \tag{79}$$

$$\forall r, r' \in R \bullet r \in RConf(r') \Rightarrow r' \in RConf(r) \tag{80}$$

Proof. The proof is provided in A.4 . □

Referring to our running example (see Section 5.6), it follows from predicates (74) and (75) that the roles *CDO* and *IWO* are conflicting since they share the conflicting permissions p_1^a and p_2^a. The same applies to the roles *SIGINT* and *ELINT*.

6. USER-ROLE INVOCATION

This section formally defines a suitable user-to-role assignment relation that limits the role membership of users to non-conflicting roles. A formal treatment of RBAC sessions is also presented and the unsuitability of DMER constraints for the enforcement of SoD policies is demonstrated.

6.1. Role-to-user assignment

A user u is eligible to invoke a role r if u's maximum clearance $f_S(u)$ dominates the security level of r. Formally, we have the following definition.

DEFINITION 6.1.

$$\forall u \in U; r \in R \bullet r \in UA(u) \Rightarrow \ell^R(r) \leq f_S(u) \tag{81}$$

We are interested in ensuring that any two conflicting roles are *mutually exclusive*, i.e. they cannot be made available to the same user.[24] As argued in [25], SMER constraints enforce an SoD policy by restricting the roles that can be assigned to a user u via the *UA* relation. To accommodate such constraints, we follow an approach analogous to the one described in Section 5.1 and introduce the function $\mathcal{CFR}$, which associates with any user u the sets of roles drawn from the powerset $\mathbb{P}UA(u)$ that do not contain any conflicting roles. More specifically, $\mathcal{CFR}(u)$ comprises all those elements of $\mathbb{P}UA(u)$ that are set-theoretically the largest subsets of $UA(u)$ that comprise solely non-conflicting roles. Formally, we have the following definition.

[23] Note that, in the case that the permissions explicitly assigned to r contain conflicting permissions, then the full set of permissions explicitly assigned to r cannot be made available through r. In such a case, none of the elements of $\mathcal{CFP}^{\preccurlyeq}(r)$ contains the full set of permissions explicitly assigned to r (recall from Definition 5.5 that $\mathcal{CFP}^{\preccurlyeq}(r)$ comprises solely sets of permissions that do not contain any conflicting permissions). The application of $\mathcal{IEP}$ to r thus returns the empty set as there are no elements of $\mathcal{CFP}^{\preccurlyeq}(r)$ to be excluded from the final choice of permissions made available through r.

[24] In this work, we only consider *pairwise* conflicting roles. Generalized t-out-of-m ($t, m \in \mathbb{Z}$) constraints such as the ones considered in [25], in which a user cannot be a member of t or more roles in a set $\{r_i | i = 1, \ldots, m\}$ of roles, are deferred for future work.

Definition 6.2. $\mathcal{CFR} : U \rightarrow \mathbb{PP}R$ *such that for any* $u \in U$*:*

$$\mathcal{CFR}(u) = \{S \in \mathbb{P}UA(u) \mid (\forall r \in S \bullet S \cap RConf(r) = \emptyset) \wedge \forall r \in UA(u) \bullet r \notin S \Rightarrow \exists r' \in S \bullet (r, r') \in RConf\}$$

For any user u, the set of roles that are made available to u is drawn from the set $\mathcal{CFR}(u)$. In the case of there being two or more distinct elements in $\mathcal{CFR}(u)$, the choice as to which one will be used for specifying the roles assigned to u is non-deterministic and dealt with at an application-specific level by a security administrator. Consequently, at the current level of abstraction, we merely assert that the set of roles assigned to u is drawn from the set $\mathcal{CFR}(u)$, without being specific about which element of $\mathcal{CFR}(u)$ that is. In this respect, we introduce the relation UA_{SoD} to associate with each user an appropriate set of non-conflicting roles; formally, we have the following definition.

Definition 6.3. $UA_{SoD} : U \leftrightarrow R$ *such that:*

$$\forall u \in U \bullet UA_{SoD}(u) \in \mathcal{CFR}(u) \tag{82}$$

6.1.1. Example

Continuing the example of Section 5.6, assume a user u whose maximum clearance is equal to c_2:

$$f_S(u) = c_2 \tag{83}$$

From Definition 6.1 it follows that u can potentially be assigned the following roles:

$$UA(u) = \{IWO, TA, SIGINT, ELINT\} \tag{84}$$

Nevertheless, as discussed in Section 5.7, certain of these roles are pairwise conflicting and thus mutually exclusive. Taking into account such role conflicts, the function $\mathcal{CFR}$ yields

$$\mathcal{CFR}(u) = \{\{IWO, TA, SIGINT\}, \{IWO, TA, ELINT\}\} \tag{85}$$

It follows that $UA_{SoD}(u)$ equals either $\{IWO, TA, SIGINT\}$ or $\{IWO, TA, ELINT\}$.

6.2. Sessions

RBAC sessions are defined through two functions

$$user : Session \rightarrow U \tag{86}$$

$$roles : Session \leftrightarrow R \tag{87}$$

such that

$$\text{dom } user = \text{dom } roles \tag{88}$$

where *Session* denotes the set of all possible session identifiers. The above are defined such that, for any session $s \in Session$, $user(s)$ denotes the user on whose behalf the session s is run and $roles(s)$ denotes the roles invoked by the session s. They are subject to the following constraint:

$$\forall s \in \text{dom } user; u : U \bullet user(s) = u \Rightarrow roles(s) \subseteq UA_{SoD}(u) \tag{89}$$

As reported in [13], a user acting through a session is a subject in the BLP sense. In this respect, for any session s, $user(s)$ is a member of the set of all subjects S:

$$\forall s \in \text{dom } user \bullet user(s) \in S \tag{90}$$

In any session s, a user u can only activate roles whose security levels are identical to u's *current* clearance $f_C(u)$:

$$\forall s \in \text{dom } user \bullet \forall r \in roles(s) \bullet \ell^R(r) = f_C(user(s)) \tag{91}$$

6.2.1. Example

Continuing the example of Section 6.1.1, let $UA_{SoD}(u) = \{IWO, TA, SIGINT\}$. It follows that the user u can activate three separate sessions, say s_1, s_2 and s_3, such that

$$roles(s_1) = \{IWO\} \tag{92}$$

$$roles(s_2) = \{TA\} \tag{93}$$

$$roles(s_3) = \{SIGINT\} \tag{94}$$

None of the roles *IWO*, *TA*, *SIGINT* can coexist in the same session as they are of differing security levels. Note that for sessions s_2 and s_3 to be activated, u's current clearance must be downgraded from c_2 (of equation (83)) to c_3 and c_4, respectively.[25]

As we would expect, no single user session can comprise conflicting roles.

Corollary 6.1.

$$\forall s \in Session; r \in R \bullet r \in roles(s) \Rightarrow RConf(r) \cap roles(s) = \emptyset \tag{95}$$

Proof. The proof follows directly from the fact that, according to (89), $roles(s)$ is a subset of $UA_{SoD}(user(s))$ and that, according to Definition 6.3, $UA_{SoD}(user(s))$ comprises solely non-conflicting roles. □

The same holds for the roles activated *across* user sessions:

Corollary 6.2.

$$\forall u \in U; r_1, r_2 \in R \bullet r_1, r_2 \in \bigcup_{s \in user^{-1}(u)} roles(s) \Rightarrow r_1 \notin RConf(r_2) \tag{96}$$

[25] In this work we do not examine administrative operations such as the ones performed for downgrading or, for that matter, upgrading a user's clearance.

Proof. The proof follows directly from the fact that, according to (89), $roles(s)$ (for all $s \in user^{-1}(u)$) is a subset of $UA_{SoD}(user(s))$ and, according to Definition 6.3, $UA_{SoD}(user(s))$ comprises solely non-conflicting roles. □

As stated in Section 5.4, DMER constraints are, in contrast to SMER constraints, unsuitable for enforcing SoD policies [25]. DMER constraints essentially stipulate that any two roles activated within, or across, a user's currently activated sessions must not be conflicting. As described in [25], consider the case in which an SoD policy states that the operations op_1 and op_2, accessible through the permissions p_1 and p_2, respectively, must not be performed by the same user u. Let p_1 and p_2 be made available through the roles r_1 and r_2, respectively. Clearly, p_1 and p_2 are conflicting, as are the roles r_1 and r_2; see Definition 5.8. Suppose now that r_1 is present in one of u's currently activated sessions, whereas r_2 is not. There is nothing that prevents u from ending the session comprising r_1 (as well as any other sessions comprising roles conflicting with r_2) and then starting a session that comprises r_2. This way u potentially violates the SoD policy (by obtaining the right to perform both op_1 and op_2) while not violating any DMER constraint. As stated [25], DMER constraints are motivated by, and thus suitable for the expression of, the least-privilege principle.

7. CONFIDENTIALITY-DRIVEN SMER IN MAC

This section formally bridges the BLP and RBAC security models. It also demonstrates that the Simple Security and ★-properties are satisfied by the SoD-aware construction presented in the previous sections. In addition, it demonstrates that the proposed model precludes any single user from possessing conflicting permissions, hence from performing conflicting operations.

7.1. Bridging BLP and RBAC

Given $u \in U$, $o \in O$ and $v \in \mathbb{P}A \setminus \emptyset$, BLP uses the predicate $(u, o, v) \in B$ to signify the state in which a user u is performing an operation with access rights v on an object o; see Section 3.1. In our RBAC construction incorporating SoD, such a predicate acquires the following meaning.

DEFINITION 7.1.

$$\begin{aligned}&\forall u \in U; o \in O; v \in \mathbb{P}A \setminus \emptyset \bullet \\ &(u, o, v) \in B \Leftrightarrow \exists s \in Session; r \in R; op \in Op \bullet \\ &\qquad u = user(s) \wedge r \in roles(s) \wedge \\ &\qquad \mathcal{A}(op) = v \wedge (r, (o, op)) \in PA^{\preccurlyeq}_{SoD}\end{aligned} \tag{97}$$

In other words, $(u, o, v) \in B$ is true iff a user u is running a session s in the capacity of a role r, which permits the operation op on object o—an operation that has the same effect on the state of o as v. It is to be noted here that the pair (o, op) may not necessarily form a permission that has been explicitly assigned to r through the relation PA_{SoD}; indeed, (o, op) may be a permission that is implicitly assigned to r by virtue of the relation $PA^{\preccurlyeq}_{SoD}$. The above definition accurately interprets our understanding of the predicate $(u, o, v) \in B$ in terms of the elements in the theory we have been constructing so far.

7.1.1. *Example*

Continuing the example of Section 6.2.1, let the user u activate the session s_1 in order to:

(i) read the tactical analysis report modelled by the object o_{TA} (see Section 4.6 and Table 1);
(ii) create a vessel interdiction recommendation object o_{IR} (see Section 4.6 and Table 1).

Through s_1, the user invokes the role *IWO* and hence, according to (75), obtains the permissions p^r_3, p^a_2 and p^r_2 defined in Table 1 by the pairs (o_{TA}, r_{TA}), (o_{IR}, a_{IR}) and (o_{IR}, r_{IR}), respectively. Given that $\mathcal{A}(r_{TA}) = \mathcal{A}(r_{IR}) = \{rd\}$ and $\mathcal{A}(a_{IR}) = \{ap\}$ (see Table 1), from Definition 7.1 it follows that the triples $(u, o_{TA}, \{rd\})$, $(u, o_{IR}, \{ap\})$ and $(u, o_{IR}, \{rd\})$ belong to the set B, signifying that u is reading the object o_{TA} while modifying and reading the object o_{IR}.

7.2. Fundamental security properties

We now present three important theorems. Our objective is to demonstrate formally that the SoD-aware confidentiality driven RBAC construction presented in this paper meets certain fundamental security properties, specifically: (a) that it satisfies the Simple Security and ★-Confidentiality properties, as understood in BLP, and (b) that it prevents a user from performing conflicting operations. The context of these theorems is the RBAC version developed here, as the sets U, O, Op, A and the state B appearing in these theorems are shared by both RBAC and BLP.

THEOREM 7.1 (ss-Property).

$$\begin{aligned}&\forall u \in U; o \in O; v \in \mathbb{P}A \setminus \emptyset \bullet (u, o, v) \in B \wedge rd \in v \Rightarrow \\ &\qquad f_O(o) \leq f_S(u)\end{aligned}$$

Proof. The proof is provided in A.5. □

THEOREM 7.2 (★-Property).

$$\begin{aligned}&\forall u \in U; o \in O; v \in \mathbb{P}A \setminus \emptyset \bullet (u, o, v) \in B \wedge ap \in v \Rightarrow \\ &\qquad f_C(u) \leq f_O(o)\end{aligned} \tag{98}$$

$$\begin{aligned}&\forall u \in U; o, o' \in O; v, v' \in \mathbb{P}A \setminus \emptyset \bullet \\ &(u, o, v) \in B \wedge ap \in v \wedge (u, o', v') \in B \wedge rd \in v' \Rightarrow \\ &f_O(o') \leq f_O(o)\end{aligned} \tag{99}$$

Proof. The proof is provided in A.6. □

Theorem 7.3 (SoD property).

$$\forall u \in U; o, o' \in O; v, v' \in \mathbb{P}A \setminus \emptyset; op, op' \in Op \bullet \left.\begin{array}{l}(u, o, v) \in B \wedge \mathcal{A}(op) = v \wedge \\ \mathcal{A}(op') = v' \wedge \\ (o', op') \in PConf((o, op))\end{array}\right\} \Rightarrow (u, o', v') \notin B \quad (100)$$

Proof. The proof is provided in A.7. □

7.2.1. *Example*

Continuing the example of Section 7.1.1, recall from equation (83) that the maximum clearance of user u is c_2. From Theorem 7.1 it follows that u can only read objects whose security level is not higher than c_2. This is indeed the case as, according to (75), the user can only obtain read access to the objects o_{IR}, o_{TA}, o_{SI} and o_{EI} through the permissions p^r_2, p^r_3, p^r_4 and p^r_5, respectively; according to Fig. 1, none of the security levels of these objects exceeds c_2. The same holds for the rest of the observational accesses that u can potentially be granted by invoking the roles in sessions s_2 and s_3.

From predicate (98) of Theorem 7.2 it follows that u can only alter objects whose security level is not lower than the current clearance c_2 of u.[26] This is indeed the case as, according to (75), the user can only obtain append access[27] to o_{IR}—an object whose security level is, according to Fig. 1, equal to c_2. The same applies to the rest of the append accesses that u can be potentially granted through the invocation of the roles in sessions s_2 and s_3.[28]

From predicate (99) of Theorem 7.2 follows that u can only alter objects whose security level is not lower than the security levels of any objects that u can observe. This is indeed the case as, according to (75), u can only obtain append access to the object o_{IR}, while any observational accesses are confined to the objects o_{IR}, o_{TA}, o_{SI} and o_{EI}. According to Fig. 1, none of the security levels of the latter objects exceeds the security level of o_{IR}. The same applies to the rest of the objects that u can access by invoking the roles in sessions s_2 and s_3.

Finally, Theorem 7.3 maintains that if u performs an operation op, then u cannot also perform any other operation that is conflicting with op. This is indeed the case as, according to (63) and (64), there are two pairs of conflicting permissions: (p^a_1, p^a_2) and (p^a_4, p^a_5). According to (75), u only obtains permission p^a_2 from the first pair, whereas none of the permissions of the second pair are obtainable by u. This means that u does not perform any of the conflicting operations a_{SI} and a_{EI} (accessed, according to Table 1, through permissions p^a_4 and p^a_5, respectively), whereas u can perform the operation a_{IR} but not its conflicting operation a_{IC} (these operations are accessed through permissions p^a_2 and p^a_1, respectively).

[26] We are assuming here that u acts through session s_1 and thus u's current clearance $f_C(u)$ is equal to u's maximum clearance c_2.

[27] Through the permission p^a_2.

[28] Recall that, for u to activate s_2 or s_3, u's current clearance must be downgraded to c_3 or c_4, respectively.

8. CONCLUSIONS AND FUTURE WORK

SoD is widely considered to be a seminal principle in computer security [3]. Although it has been studied extensively within the context of role-based access control models [4, 9–11], the same cannot be claimed for Multi-Level MAC Security models. One reason for this shortcoming is the fact that the expression of SoD constraints requires an approach whereby the operations that a user is allowed to perform are determined on the basis of the effect that each operation has on *individual* objects. Such an approach does not conform with the abstract view of operations typically encountered in MAC models—a view whereby the operations that a user is permitted to perform on objects are determined on the basis of a comparison between the user's clearance, on the one hand, and the security level of the objects, on the other. Such a view overlooks, in essence, the effect that these operations bring about on individual objects. In order to enable the expression of SoD constraints, this work adopts a slightly detailed view of RBAC whereby objects are associated with relevant operations with applicable functional capabilities. Naturally, this view fits in well with the conventional object-oriented perspective whereby operations are defined as part of individual classes of objects.

Additionally, this work takes advantage of one of RBAC's important virtues, namely its generality in being able to emulate the BLP security model [13], thus enabling Multi-level MAC Security models to express SoD constraints. An important feature of our framework is that each permission is assumed to possess a certain capacity, determined by the BLP security level of the object with which it is associated. This helps characterize permissions with security levels and, in turn, in defining a partial order on permissions and, thus, a permission hierarchy. Such a hierarchy serves two purposes:

(i) On the one hand, it provides a formal basis for defining role seniority and thus a hierarchical ordering on roles. Owing to its derivation on the basis of the capacity of the permissions that it comprises, the seniority of a role reflects the security level of objects accessible through those permissions. This provides an intuitive definition of role seniority and promotes the view whereby roles are not by themselves carriers of seniority values but a named collection of permissions [16], i.e. an auxiliary mechanism introduced for simplifying security administration by facilitating the assignment of permissions to users.

(ii) On the other hand, it provides the means to express SoD constraints in terms of conflicting permissions that cannot be assigned to the same role. This naturally extends the hierarchical ordering of roles with mutually

exclusive roles, that is, roles that cannot be assigned to the same user.

In addition, our framework essentially bridges the BLP and RBAC worlds, while retaining the former's core security properties, namely the Simple Security Property and the ★-Property.

The notion of permission hierarchy offers certain additional advantages. Firstly, it gives rise to the concept of *direct permission inheritance* (see Section 5.3), whereby a role r automatically inherits all those permissions that are junior to the permissions explicitly assigned to it, in contrast to the indirect permission inheritance, as understood generally in RBAC, whereby r can inherit only those permissions assigned explicitly to roles junior to r. In our view, direct permission inheritance enables greater control over how permissions are assigned, implicitly or otherwise, to roles and, hence, to users. Secondly, defining SoD constraints directly in terms of conflicting permissions offers greater assurance than defining them indirectly through conflicting roles with no reference to the actual conflicting permissions [27].

As far as future work is concerned, we intend to pursue two main directions of research: (a) investigation of ways to extend and reinforce our framework; (b) investigation of automated methods and tools for policy validation and verification. More specifically, with respect to the first direction we intend to do the following:

(i) Investigate roles that span over different compartments (in the BLP sense). More specifically, we are interested in enriching our framework with roles that encompass permissions from different compartments, and studying the effect that such roles have on role hierarchies. Consideration of such roles is anticipated to increase the generality of our approach.
(ii) Dispense with pairwise-conflicting roles through the adoption of the more generalized t-out-of-m ($t, m \in \mathbb{N}$) SoD constraints [25], in which a user cannot be a member of t or more roles in a set consisting of m roles.
(iii) Incorporate in our framework administrative operations such as the ones performed for downgrading or upgrading a user's clearance.
(iv) Investigate the feasibility of extending our atomic permissions to take into account environmental and contextual attributes. More specifically, we are interested in exploring the effect of such attributes on permission and role hierarchies. Such attributes would enhance the generality of our framework by incorporating features from Attribute-Based Access Control.

With respect to the second direction, we intend to investigate methods and tools for verifying whether policies expressed through particular configurations of our framework capture specific security properties. Policy analysis and verification is a well-known problem of access control models [39]. Automated verification through model checking has been proposed as a solution to this problem, and several relevant approaches have been reported in the literature, for example [39–43]. Regarding our work, we intend to investigate an approach akin to the one proposed in [43], which has the capability to model-check an entire configuration of an access control framework rather than individual policies. Such an approach would enable us to verify whether a user is allowed to perform a certain operation on a particular object in a given configuration, i.e. for a given set of object security levels, and allowable permission-to-role and role-to-user assignments. This requires the expression of a particular configuration of our framework in the specification language of a model checker, i.e. in the form of a deterministic Finite State Machine. It also requires the expression of access control properties in the model checker's property language as temporal logic formulae (e.g. in CTL or LTL). The verification capabilities of the model checker may then be exploited in order to automatically verify that the specified configuration possesses the expressed properties.

A different, albeit related, avenue of research that we could potentially explore is the application of 'light-weight' approaches[29] to policy validation such as the one reported in [44, 45]. This entails the use of UML diagrams for representing the various classes of our framework (e.g. permissions, roles, users etc.), and their relationships (e.g. permission-to-role assignments, role-to-user assignments etc.). It also entails the use of some underlying formalism such as OCL[30] or RCL2000 [27] for specifying constraints (e.g. exclusion constraints such as the ones captured through the relation $PA^{\preccurlyeq}_{Excl}$ in Definition 5.4, or SoD constraints). Through the use of a tool such as RAE (RBAC Authorization Environment) [45], it is then possible to automatically generate system states (represented as UML object diagrams) and check them against the specified constraints. Additionally, we could also consider the provision of a facility analogous to the one offered by RAE, which allows the automatic generation of code implementing a particular configuration of our framework subject to any relevant constraints. Overall, we expect such a UML-based approach to fit in well with our object-oriented perspective of objects and operations outlined in Section 4.3.

Finally, we intend to investigate an approach akin to the one proposed in [46], which expresses BLP in terms of UML diagrams and then automatically translates these diagrams into the input language of a model checker through the use of appropriate algorithms such as, for example, the Hugo/RT algorithm [47]. The desirable properties (e.g. the ★-Property) against which the BLP model is to be verified are formulated in LTL. The verification capabilities of the model checker may then be exploited in order to automatically verify that the specified BLP configuration possesses the desired properties.

[29] 'Light-weight' in the sense that they avoid direct use of model checking tools which are cumbersome for non-experts. This may potentially render our framework usable by a wider audience.
[30] OCL is UML's constraint specification language.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Christos Ilioudis for his support and encouragement. They would also like to thank the reviewers for their constructive criticisms and valuable comments that helped improve this manuscript.

FUNDING

Veloudis' research was supported by the Research Committee of ATEI of Thessaloniki (grant number 2010-80161).

REFERENCES

[1] Bell, D. E. and LaPadula, L. J. (1976) Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report MTR-2997. MITRE Corporation, Bedford, MA, USA.

[2] Biba, K. J. (1977) Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR-76-372. MITRE Corporation, Bedford, MA, USA.

[3] Clark, D. D. and Wilson, D. R. (1987) A Comparison of Commercial and Military Computer Security Policies. *Proc. 1987 IEEE Symp. Security and Privacy*, Oakland, CA, USA, April 27–29, pp. 184–194. IEEE Computer Society, Los Alamitos, CA, USA.

[4] Sandhu, R. S., Coyne, E. J., Feinstein, H. L. and Youman, C. E. (1996) Role-based access control models. *IEEE Comput.*, **29**, 38–47.

[5] Ferraiolo, D. F., Sandhu, R. S., Gavrila, S., Kuhn, D. R. and Chandramouli, R. (2001) Proposed NIST standard for Role-Based Access Control. *ACM Trans. Inf. Syst. Secur.*, **4**, 224–274.

[6] Downs, D. D., Rub, J. R., Kung, K. C. and Jordan, C. S. (1985) Issues in Discretionary Access Control. *IEEE Symp. Security and Privacy*, Oakland, CA, USA, April 18–21, pp. 208–218. IEEE Computer Society, Los Alamitos, CA, USA.

[7] Gollmann, D. (2006) *Computer Security* (2nd edn). John Willey and Sons Ltd., Hoboken, NJ.

[8] Harrison, M. A., Ruzzo, W. L. and Ullman, J. D. (1976) Protection in operating systems. *Commun. ACM*, **19**, 461–471.

[9] Simon, R. and Zurko, M. E. (1997) Separation of Duty in Role-based Environments. *Proc. 10th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 10–12, pp. 183–194. IEEE Computer Society, Los Alamitos, CA, USA.

[10] Gavrila, S. I. and Barkley, J. F. (1998) Formal Specification for Role-Based Access Control User/Role and Role/Role Relationship Management. *Proc. 3rd ACM Workshop on Role-Based Access Control (RBAC'98)*, Fairfax, Virginia, October 22-23, pp. 81–90. ACM, New York, NY, USA.

[11] Jaeger, T. and Tidswell, J. E. (2001) Practical safety in flexible access control models. *ACM Trans. Inf. Syst. Secur.*, **4**, 158–190.

[12] Crampton, J. (2003) Specifying and Enforcing Constraints in Role-Based Access Control. *Proc. 8th ACM Symp. Access Control Models and Technologies*, Como, Italy, June 2–3, pp. 43–50. ACM, New York, NY, USA.

[13] Osborn, S., Sandhu, R. and Munawer, Q. (2000) Configuring Role-Based Access Control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.*, **3**, 85–106.

[14] Crampton, J. (2003) On Permissions, Inheritance and Role Hierarchies. *Proc. 10th ACM Conf. Computer and Communications Security*, Washington, DC, October 27–30, pp. 85–92. ACM, New York, NY, USA.

[15] Chen, L. (2011) Analyzing and developing role-based access control models. PhD Thesis, Royal Holloway, University of London.

[16] Ferraiolo, D. F., Cugini, J. A. and Kuhn, R. D. (1995) Role-Based Access Control (RBAC): Features and Motivations. *Proc. 11th Annual Computer Security Applications Conf.*, Louisiana, December 11–15, pp. 241–248. IEEE Computer Society, Los Alamitos, CA.

[17] Brewer, D. F. and Nash, M. J. (1989) The Chinese Wall Security Policy. *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, May 1–3, pp. 206–214. IEEE Computer Security, Los Alamitos, CA, USA.

[18] Sandhu, R., Ferraiolo, D. and Kuhn, R. (2000) The NIST Model for Role-Based Access Control: Towards a Unified Standard. *Proc. 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 26–28, pp. 47–63. ACM, New York, NY, USA.

[19] Barka, E. and Sandhu, R. (2000) Framework for Role-based Delegation Models. *Proc. 16th Annual Computer Security Applications Conf.*, Louisiana, December 11–15, pp. 168–176. IEEE Computer Society, Los Alamitos, CA, USA.

[20] Zhang, L., Ahn, G.-J. and Chu, B.-T. (2001) A Rule-based Framework for Role-based Delegation. *Proc. 6th ACM Symp. Access Control Models and Techonologies*, Chantili, Virginia, May 3–4, pp. 171–181. ACM, New York, NY, USA.

[21] Zhang, X., Oh, S. and Sandhu, R. (2003) PBDM: A Flexible Delegation Model in RBAC. *Proc. 8th ACM Symp. Access Control Models and Technologies*, Como, Italy, June 2–3, pp. 149–157. ACM, New York, NY, USA.

[22] Zhao, G. and Chadwick, D. W. (2008) On the Modeling of Bell-LaPadula Security Policies Using RBAC. *Proc. 2008 IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Rome, June 23–25, pp. 257–262. IEEE Computer Security, Los Alamitos, CA, USA.

[23] Habib, L., Jaume, M. and Morisset, C. (2008) A Formal Comparison of the Bell & LaPadula and RBAC Models. *Proc. 4th Int. Conf. Information Assurance and Security (IAS'8)*, Napoli, Italy, September 8–10, pp. 3–8. IEEE Computer Society, Los Alamitos, CA, USA.

[24] Kuijper, W. and Ermolaev, V. (2014) Sorting Out Role Based Access Control. *Proc. 19th ACM Symp. Access Control Models and Technologies (SACMAT'14)*, London, ON, Canada, June 25–27, pp. 63–74. ACM, New York, NY, USA.

[25] Li, N., Tripunitara, M. V. and Ziad, B. (2007) On mutually exclusive roles and separation of duty. *ACM Trans. Inf. Syst. Secur. (TISSEC)*, **10**, 1–36.

[26] Gligor, V. D., Gavrila, S. I. and Ferraiolo, D. F. (1998) On the Formal Definition of Separation-of-Duty Policies and their Composition. *Proc. IEEE Symp. Research in Security and Privacy*, Oakland, CA, May 4–6, pp. 172–183. IEEE Computer Society, Los Alamitos, CA, USA.

[27] Ahn, G.-J. and Sandhu, R. (2000) Role-based authorization constraints specification. *ACM Trans. Inf. Syst. Secur.*, **3**, 207–226.

[28] Joshi, J. B. D., Bertino, E., Latif, U. and Ghafoor, A. (2005) A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, **17**, 4–23.

[29] Roy, A., Sural, S. and Majumdar, A. (2012) Minimum User Requirement in Role Based Access Control with Separation of Duty Constraints. *12th Int. Conf. Intelligent Systems Design and Applications (ISDA)*, Kochi, India, November 27–29, pp. 386–391. IEEE Computer Society, Los Alamitos, CA, USA.

[30] Bertino, E., Ferrari, E. and Atluri, V. (1999) The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, **2**, 65–104.

[31] Crampton, J. (2005) A Reference Monitor for Workflow Systems with Constrained Task Execution. *Proc. 10th ACM Symp. Access Control Models and Technologies*, Stockholm, Sweden, June 1–3, pp. 38–47. ACM, New York, NY, USA.

[32] Li, N. and Wang, Q. (2008) Beyond separation of duty: An algebra for specifying high-level security policies. *J. ACM*, **55**, 1–46.

[33] Basin, D., Burri, S. J. and Karjoth, G. (2009) Dynamic Enforcement of Abstract Separation of Duty Constraints. *Proc. 14th European Conf. Research in Computer Security (ESORICS'09)*, Saint-Malo, France, September 21–23, pp. 250–267. Springer, Berlin, Heidelberg.

[34] Basin, D., Burri, S. J. and Karjoth, G. (2012) Dynamic enforcement of abstract separation of duty constraints. *ACM Trans. Inf. Syst. Secur.*, **15**, 1–30.

[35] Hoare, C. A. R. (1985) *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[36] Habib, M. A., Mahmood, N., Shahid, M., Aftab, M. U., Ahmad, U. and Faisal, C. N. (2014) Permission Based Implementation of Dynamic Separation of Duty (DSD) in Role based Access Control (RBAC). *8th Int. Conf. Signal Processing and Communication Systems (ICSPCS)*, Gold Coast, Australia, December 16–18, pp. 1–10. IEEE Computer Society, Los Alamitos, CA, USA.

[37] Lu, J. and Zhou, J. (2011) Specification and Enforcement of Separation-of-Duty Policies in Role-base Access Control. *Int. Conf. Computer Science and Network Technology (ICCSNT)*, Harbin, China, December 24–26, pp. 2135–2140. IEEE Computer Society, Los Alamitos, CA, USA.

[38] McDaniel, C. R. and Tardy, M. L. (2005) Role-Based Access Control for coalition partners in maritime domain awareness. Master's Thesis, Naval Postgraduate School, Monterey, CA.

[39] Jayaraman, K., Ganesh, V., Tripunitara, M., Rinard, M. and Chapin, S. (2011) Automatic Error Finding in Access-control Policies. *Proc. 18th ACM Conf. Computer and Communications Security (CCS'11)*, Chicago, IL, USA, June 11–15, pp. 163–174. ACM, New York, NY, USA.

[40] Zhang, N., Ryan, M. and Guelev, D. P. (2005) Evaluating Access Control Policies through Model Checking. *Proc. 8th Information Security Conference (ISC 2005)*, Singapore, September 20–23, pp. 446–460. Springer, Berlin, Heidelberg.

[41] Kikuchi, S., Tsuchiya, S., Adachi, M. and Katsuyama, T. (2007) Policy Verification and Validation Framework based on Model Checking Approach. *Proc. 4th Int. Conf. Autonomic Computing (ICAC'07)*, Jacksonville, FL, USA, October 17–21, pp. 1–9. IEEE Computer Society, Los Alamitos, CA, USA.

[42] Hu, H. and Ahn, G. (2008) Enabling Verification and Conformance Testing for Access Control Model. *Proc. 13th ACM Symp. Access Control Models and Technologies (SACMAT'08)*, Estes Park, CO, USA, June 11–13, pp. 195–204. ACM, New York, NY, USA.

[43] Hu, V. C., Kuhn, D. R., Xie, T. and Hwang, J. (2011) Model checking for verification of mandatory access control models and properties. *Int. J. Softw. Eng. Knowl. Eng.*, **21**, 103–127.

[44] Sohr, K., Drouineaud, M., Ahn, G.-J. and Gogolla, M. (2008) Analyzing and managing role-based access control policies. *IEEE Trans. Knowl. Data Eng.*, **20**, 924–939.

[45] Hu, H. and Ahn, G.-J. (2010) Constructing authorization systems using assurance management framework. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.*, **40**, 396–405.

[46] Cheng, L. and Zhang, Y. (2011) Model Checking Security Policy Model using Both UML Static and Dynamic Diagrams. *Proc. 4th Int. Conf. Security of Information and Networks (SIN'11)*, Sydney, Australia, November 14–19, pp. 159–166. ACM, New York, NY, USA.

[47] Balser, M., Baumler, S., Knapp, A., Reif, W. and Thums, A. (2004) Interactive Verification of UML State Machines. *Proc. 6th Int. Conf. Formal Engineering Methods (ICFEM 2004)*, Seattle, WA, USA, November 8–12, pp. 434–448. Springer, Berlin, Heidelberg.

APPENDIX

The formal proofs in this appendix are Fitch-style. The following notation is used. A horizontal line indicates a hypothesis. A vertical line indicates the scope of a hypothesis. The annotations x-I and x-E, where x is any logical quantifier or connective, denote the *introduction* and *elimination*, respectively, of the quantifier or connective. x-A, where x is any logical connective, stands for the *associative* property that the connective may possess. The abbreviations Dk, Tk and Ck (for integer k) indicate, respectively, *Definition* k, *Theorem* k and *Corollary* k.

A.1. Proof of Theorem 4.1

Reflexivity

Suppose arbitrary $p \in P$ and let $p = (o, op)$ for some $o \in O, op \in Op$.

1	$\ell^P(p) = \ell^P(p) \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op)$	(tautology)
2	$(\ell^P(p) = \ell^P(p) \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op)) \vee$	1, $\vee$-I
	$(\ell^P(p) < \ell^P(p) \wedge \mathcal{A}(op) = \{rd\} \wedge$	
	$\mathcal{A}(op) \subseteq \mathcal{A}(op)) \vee$	
	$(\ell^P(p) > \ell^P(p) \wedge \mathcal{A}(op) = \{ap\} \wedge$	
	$\mathcal{A}(op) \subseteq \mathcal{A}(op))$	
3	$p \preccurlyeq p$	2, D4.2
4	$p \preccurlyeq p$	1, 3, $\Rightarrow$-I

Antisymmetry

Suppose arbitrary $p, p' \in P$ and let $p = (o, op), p' = (o', op')$ for some $o, o' \in O, op, op' \in Op$. Also, let the abbreviations:

$$L_1 \widehat{=} \ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \quad \text{(A.1)}$$
$$R_1 \widehat{=} \ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op) \quad \text{(A.2)}$$
$$L_2 \widehat{=} \ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \quad \text{(A.3)}$$
$$R_2 \widehat{=} \ell^P(p') < \ell^P(p) \wedge \mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op) \quad \text{(A.4)}$$
$$L_3 \widehat{=} \ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \quad \text{(A.5)}$$
$$R_3 \widehat{=} \ell^P(p') > \ell^P(p) \wedge \mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op) \quad \text{(A.6)}$$

In the following proof the antisymmetry of the set inclusion relation ($\subseteq$) is used without explicit mention.

1	$p \preccurlyeq p' \wedge p' \preccurlyeq p$	
2	$p \preccurlyeq p'$	1, $\wedge$-E
3	$(\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op')) \vee$ $(\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op')) \vee$ $(\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'))$	2, D4.2
4	$p' \preccurlyeq p$	1, $\wedge$-E
5	$(\ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)) \vee$ $(\ell^P(p') < \ell^P(p) \wedge \mathcal{A}(op') = \{rd\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op)) \vee$ $(\ell^P(p') > \ell^P(p) \wedge \mathcal{A}(op') = \{ap\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op))$	4, D4.2
6	$(L_1 \vee L_2 \vee L_3) \wedge (R_1 \vee R_2 \vee R_3)$	3, 5, $\wedge$-I
7	$(L_1 \wedge (R_1 \vee R_2 \vee R_3)) \vee$ $(L_2 \wedge (R_1 \vee R_2 \vee R_3)) \vee$ $(L_3 \wedge (R_1 \vee R_2 \vee R_3))$	6, $\wedge$-A
8	$L_1 \wedge (R_1 \vee R_2 \vee R_3)$	
9	$(L_1 \wedge R_1) \vee (L_1 \wedge R_2) \vee (L_1 \wedge R_3)$	8, $\wedge$-A
10	$L_1 \wedge R_1$	
11	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.1), (A.2)
12	$\ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') = \mathcal{A}(op)$	11
13	$L_1 \wedge R_2$	
14	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') < \ell^P(p) \wedge \mathcal{A}(op') = \{rd\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.1), (A.4)
15	`false`	14
16	$L_1 \wedge R_3$	
17	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') > \ell^P(p) \wedge \mathcal{A}(op') = \{ap\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.1), (A.6)
18	`false`	17
19	$\ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') = \mathcal{A}(op)$	9, 10–18, $\vee$-E
20	$L_2 \wedge (R_1 \vee R_2 \vee R_3)$	
21	$(L_2 \wedge R_1) \vee (L_2 \wedge R_2) \vee (L_2 \wedge R_3)$	20, $\wedge$-A
22	$L_2 \wedge R_1$	
23	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') = \ell^P(p) \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.3), (A.2)
24	`false`	23
25	$L_2 \wedge R_2$	
26	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') < \ell^P(p) \wedge$ $\mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.3), (A.4)
27	`false`	26
28	$L_2 \wedge R_3$	
29	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') > \ell^P(p) \wedge$ $\mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.3), (A.6)
30	$\mathcal{A}(op) = \{rd\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	29, $\wedge$-E
31	$\mathcal{A}(op) = \{rd\} \wedge \mathcal{A}(op') = \{ap\} \wedge$ $\mathcal{A}(op) = \mathcal{A}(op')$	30
32	`false`	31

33	`false`	21, 22–32, ∨-E
34	$L_3 \wedge (R_1 \vee R_2 \vee R_3)$	
35	$(L_3 \wedge R_1) \vee (L_3 \wedge R_2) \vee (L_3 \wedge R_3)$	34, ∧-A
36	$L_3 \wedge R_1$	
37	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') = \ell^P(p) \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.5), (A.2)
38	`false`	37
39	$L_3 \wedge R_2$	
40	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') < \ell^P(p) \wedge$ $\mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.5), (A.4)
41	$\mathcal{A}(op) = \{ap\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	40, ∧-E
42	$\mathcal{A}(op) = \{ap\} \wedge \mathcal{A}(op') = \{rd\} \wedge$ $\mathcal{A}(op) = \mathcal{A}(op')$	41
43	`false`	42
44	$L_3 \wedge R_3$	
45	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') > \ell^P(p) \wedge$ $\mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op)$	(A.1), (A.6)
46	`false`	42
47	`false`	35, 36–46, ∨-E
48	$\ell^P(p') = \ell^P(p) \wedge \mathcal{A}(op') = \mathcal{A}(op)$	7, 8–47, ∨-E
49	$p \approx p'$	48, D4.3
50	$p \preccurlyeq p' \wedge p' \preccurlyeq p \Rightarrow p \approx p'$	1, 49, ⇒-I

Transitivity

Suppose arbitrary $p, p', p'' \in P$ and let $p = (o, op), p' = (o', op'), p'' = (o'', op'')$ for some $o, o', o'' \in O, op, op', op'' \in Op$. In the following proof, the transitivity of the relations $=$, $\subseteq$, and $>$ is used without explicit mention.

1	$p \preccurlyeq p' \wedge p' \preccurlyeq p''$	
2	$p \preccurlyeq p'$	1, ∧-E
3	$(\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op')) \vee$ $(\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op')) \vee$ $(\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'))$	2, D4.2
4	$p' \preccurlyeq p''$	1, ∧-E
5	$(\ell^P(p') = \ell^P(p'') \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')) \vee$ $(\ell^P(p') < \ell^P(p'') \wedge \mathcal{A}(op') = \{rd\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op'')) \vee$ $(\ell^P(p') > \ell^P(p'') \wedge \mathcal{A}(op') = \{ap\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op''))$	4, D4.2

Let now

$$R'_1 \widehat{=} \ell^P(p') = \ell^P(p'') \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'') \quad \text{(A.7)}$$

$$R'_2 \widehat{=} \ell^P(p') < \ell^P(p'') \wedge \mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'') \quad \text{(A.8)}$$

$$R'_3 \widehat{=} \ell^P(p') > \ell^P(p'') \wedge \mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'') \quad \text{(A.9)}$$

Continuing the proof:

6	$(L_1 \vee L_2 \vee L_3) \wedge (R'_1 \vee R'_2 \vee R'_3)$	3, 5, ∧-I
7	$(L_1 \wedge (R'_1 \vee R'_2 \vee R'_3)) \vee$ $(L_2 \wedge (R'_1 \vee R'_2 \vee R'_3)) \vee$ $(L_3 \wedge (R'_1 \vee R'_2 \vee R'_3))$	6, ∧-A
8	$L_1 \wedge (R'_1 \vee R'_2 \vee R'_3)$	
9	$(L_1 \wedge R'_1) \vee (L_1 \wedge R'_2) \vee (L_1 \wedge R'_3)$	8, ∧-A
10	$L_1 \wedge R'_1$	
11	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') = \ell^P(p'') \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.1), (A.7)
12	$\ell^P(p) = \ell^P(p'') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op'')$	11, ∧-E
13	$L_1 \wedge R'_2$	

14	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') < \ell^P(p'') \wedge \mathcal{A}(op') = \{rd\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.1), (A.8)
15	$\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	14, $\wedge$-E
16	$L_1 \wedge R'_3$	
17	$\ell^P(p) = \ell^P(p') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\ell^P(p') > \ell^P(p'') \wedge \mathcal{A}(op') = \{ap\} \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.1), (A.9)
18	$\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	17, $\wedge$-E
19	$(\ell^P(p) = \ell^P(p'') \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$ $(\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$ $(\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op''))$	9, 10–15, 18, $\vee$-I
20	$p \preccurlyeq p''$	19, D4.2
21	$L_2 \wedge (R'_1 \vee R'_2 \vee R'_3)$	
22	$(L_2 \wedge R'_1) \vee (L_2 \wedge R'_2) \vee (L_2 \wedge R'_3)$	21, $\wedge$-A
23	$L_2 \wedge R'_1$	
24	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') = \ell^P(p'') \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.3), (A.7)
25	$\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	24, $\wedge$-E
26	$L_2 \wedge R'_2$	
27	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') < \ell^P(p'') \wedge$ $\mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.3), (A.8)
28	$\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	27, $\wedge$-E
29	$L_2 \wedge R'_3$	
30	$\ell^P(p) < \ell^P(p') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') > \ell^P(p'') \wedge$ $\mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.3), (A.9)
31	$\mathcal{A}(op) = \{rd\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\mathcal{A}(op') = \{ap\}$	30, $\wedge$-E
32	`false`	31
33	$(\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$ $(\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$ `false`	22. 23–28, $\vee$-I
34	$\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	33, $\vee$-E
35	$L_3 \wedge (R'_1 \vee R'_2 \vee R'_3)$	
36	$(L_3 \wedge R'_1) \vee (L_3 \wedge R'_2) \vee (L_3 \wedge R'_3)$	35, $\wedge$-A
37	$L_3 \wedge R'_1$	
38	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') = \ell^P(p'') \wedge$ $\mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.5), (A.7)
39	$\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	38, $\wedge$-E
40	$L_3 \wedge R'_2$	
41	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') < \ell^P(p'') \wedge$ $\mathcal{A}(op') = \{rd\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.5), (A.8)
42	$\mathcal{A}(op) = \{ap\} \wedge \mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge$ $\mathcal{A}(op') = \{rd\}$	41, $\wedge$-E
43	`false`	42
44	$L_3 \wedge R'_3$	
45	$\ell^P(p) > \ell^P(p') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op') \wedge \ell^P(p') > \ell^P(p'') \wedge$ $\mathcal{A}(op') = \{ap\} \wedge \mathcal{A}(op') \subseteq \mathcal{A}(op'')$	(A.5), (A.9)
46	$\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	45, $\wedge$-E
47	$(\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$ `false` $\vee$ $(\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$ $\mathcal{A}(op) \subseteq \mathcal{A}(op''))$	36, 37–46 $\vee$-I
48	$\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$	46, $\vee$-E

	$\mathcal{A}(op) \subseteq \mathcal{A}(op'')$	
49	$p \preccurlyeq p'' \vee$	7,
	$(\ell^P(p) < \ell^P(p'') \wedge \mathcal{A}(op) = \{rd\} \wedge$	8–48,
	$\mathcal{A}(op) \subseteq \mathcal{A}(op'')) \vee$	$\vee$-I
	$(\ell^P(p) > \ell^P(p'') \wedge \mathcal{A}(op) = \{ap\} \wedge$	
	$\mathcal{A}(op) \subseteq \mathcal{A}(op''))$	
50	$p \preccurlyeq p''$	49, $\vee$-E, D4.2
51	$p \preccurlyeq p' \wedge p' \preccurlyeq p'' \Rightarrow p \preccurlyeq p''$	1, 50, $\Rightarrow$-I

Note that the latter two disjuncts of line 49 are subsumed within the first disjunct of the same line.

A.2. Proof of Corollary 5.1

Suppose that $PA(r)$ contains more than two elements. Then there are inevitably two or more permissions in $PA(r)$ all of which grant (at least) a `read` access, or grant (at least) an `append` access, and thus at least two permissions are mutually comparable (recall that we are assuming that all permissions pertain to a single compartment in the BLP sense). Assignment of any such pair of mutually comparable permissions to a role explicitly violates condition (27).

Suppose now that p, p' are distinct permissions. If $\#\mathcal{A}(op) > 1$, then $\mathcal{A}(op) = \{rd, ap\}$ and hence p is necessarily comparable with p' (recall again that we are assuming that all permissions pertain to a single compartment). It follows from (27) that p and p' cannot be both explicitly assigned to r. An analogous argument holds for the case in which $\#\mathcal{A}(op') > 1$. Next, suppose that $\mathcal{A}(op) = \mathcal{A}(op')$. Then p and p' are by definition comparable and, by virtue of (27), cannot be both assigned to r.

A.3. Role seniority—alternative formulation

THEOREM A.1. *The role seniority relation defined through predicate (30) is a partial order.*

Proof. Reflexivity
Suppose arbitrary $r \in R$ and let $p \in PA_{SoD}(r)$. The proof is trivial:

1	$p \preccurlyeq p$	T4.1
2	$\forall p \in PA(r) \bullet \exists p \in PA(r) \bullet p \preccurlyeq p$	1
3	$IP(PA(r))$	(14), (27)
4	$IP(PA(r)) \wedge$ $\forall p \in PA(r) \bullet \exists p \in PA(r) \bullet p \preccurlyeq p$	2, 3, $\wedge$-I
5	$PA(r) \preccurlyeq PA(r)$	4, (15)
6	$r \preccurlyeq r$	5, D5.2
7	$r \preccurlyeq r$	1, 6

Antisymmetry
Suppose arbitrary $r_1, r_2 \in R$. We assume here that if two roles entail permissions of equal seniority, i.e. if $PA(r_1) \approx PA(r_2)$, then the roles are of equal seniority too (denoted $r_1 \approx r_2$).

1	$r_1 \preccurlyeq r_2 \wedge r_2 \preccurlyeq r_1$	
2	$r_1 \preccurlyeq r_2$	1, $\wedge$-E
3	$PA(r_1) \preccurlyeq PA(r_2)$	2, D5.2
4	$\forall p_1 \in PA(r_1) \bullet \exists p_2 \in PA(r_2) \bullet$ $p_1 \preccurlyeq p_2$	3, (15), $\wedge$-E
5	$p_1 \in PA(r_1)$	arbitrary p_1
6	$p_1 \in PA(r_1) \Rightarrow$ $\exists p_2 \in PA(r_2) \bullet p_1 \preccurlyeq p_2$	4, $\forall$-E
7	$\exists p_2 \in PA(r_2) \bullet p_1 \preccurlyeq p_2$	5, 6, $\Rightarrow$-E
8	$\overline{p_2} \in PA(r_2) \wedge p_1 \preccurlyeq \overline{p_2}$	7, $\exists$-E
9	$\overline{p_2} \in PA(r_2)$	8, $\wedge$-E
10	$r_2 \preccurlyeq r_1$	1, $\wedge$-E
11	$PA(r_2) \preccurlyeq PA(r_1)$	10, D5.2
12	$\forall p_2 \in PA(r_2) \bullet$ $\exists p_1 \in PA(r_1) \bullet p_2 \preccurlyeq p_1$	11, (15), $\wedge$-E
13	$\overline{p_2} \in PA(r_2) \Rightarrow$ $\exists p_1 \in PA(r_1) \bullet \overline{p_2} \preccurlyeq p_1$	12, $\forall$-E
14	$\exists p_1 \in PA(r_1) \bullet \overline{p_2} \preccurlyeq p_1$	9, 13, $\Rightarrow$-E
15	$\overline{p_1} \in PA(r_1) \wedge \overline{p_2} \preccurlyeq \overline{p_1}$	14, $\exists$-E
16	$\overline{p_1} \in PA(r_1)$	15, $\wedge$-E
17	$\neg(p_1 \preccurlyeq \overline{p_1}) \vee p_1 = \overline{p_1}$	15, 16, (27)
18	$p_1 \preccurlyeq \overline{p_2}$	8, $\wedge$-E
19	$\overline{p_2} \preccurlyeq \overline{p_1}$	15, $\wedge$-E
20	$p_1 \preccurlyeq \overline{p_1}$	18, 19, T4.1
21	$p_1 = \overline{p_1}$	17, 20, $\vee$-E
22	$p_1 \approx \overline{p_2}$	18, 19, 21, T4.1
23	$\exists p_2 \in PA_{SoD}(r_2) \bullet p_1 \approx p_2$	22, $\exists$-I

24	$p_1 \in PA(r_1) \Rightarrow \exists p_2 \in PA(r_2) \bullet p_1 \approx p_2$	5, 23, $\Rightarrow$-I
25	$\forall p_1 \in PA(r_1) \bullet \exists p_2 \in PA(r_2) \bullet p_1 \approx p_2$	5, 24, $\forall$-I

An entirely symmetrical line of reasoning as the one provided through lines 2–25 yields:

26	$\forall p_2 \in PA(r_2) \bullet \exists p_1 \in PA(r_1) \bullet p_2 \approx p_1$	1
27	$(\forall p_1 \in PA(r_1) \bullet \exists p_2 \in PA(r_2) \bullet p_1 \approx p_2) \wedge (\forall p_2 \in PA(r_2) \bullet \exists p_1 \in PA(r_1) \bullet p_2 \approx p_1)$	25, 26, $\wedge$-I
28	$PA(r_1) \approx PA(r_2)$	27, (16)
29	$r_1 \approx r_2$	28
30	$r_1 \preccurlyeq r_2 \wedge r_2 \preccurlyeq r_1 \Rightarrow r_1 \approx r_2$	1, 29, $\Rightarrow$-I

Transitivity
Suppose arbitrary $r_1, r_2, r_3 \in R$.

1	$r_1 \preccurlyeq r_2 \wedge r_2 \preccurlyeq r_3$	
2	$r_1 \preccurlyeq r_2$	1, $\wedge$-E
3	$PA(r_1) \preccurlyeq PA(r_2)$	2, D5.2
4	$\forall p_1 \in PA(r_1) \bullet \exists p_2 \in PA(r_2) \bullet p_1 \preccurlyeq p_2$	3, (15), $\wedge$-E
5	$p_1 \in PA(r_1) \Rightarrow \exists p_2 \in PA(r_2) \bullet p_1 \preccurlyeq p_2$	4, $\forall$-E
6	$p_1 \in PA(r_1)$	arbitrary p_1
7	$\exists p_2 \in PA(r_2) \bullet p_1 \preccurlyeq p_2$	5, 6, $\Rightarrow$-E
8	$\overline{p_2} \in PA(r_2) \wedge p_1 \preccurlyeq \overline{p_2}$	7, $\exists$-E
9	$\overline{p_2} \in PA(r_2)$	8, $\wedge$-E
10	$r_2 \preccurlyeq r_3$	1, $\wedge$-E
11	$PA(r_2) \preccurlyeq PA(r_3)$	10, D5.2
12	$\forall p_2 \in PA(r_2) \bullet \exists p_3 \in PA(r_3) \bullet p_2 \preccurlyeq p_3$	11, (15), $\wedge$-E
13	$\overline{p_2} \in PA(r_2) \Rightarrow \exists p_3 \in PA(r_3) \bullet \overline{p_2} \preccurlyeq p_3$	12, $\forall$-E
14	$\exists p_3 \in PA(r_3) \bullet \overline{p_2} \preccurlyeq p_3$	9, 13, $\Rightarrow$-E
15	$\overline{p_3} \in PA(r_3) \wedge \overline{p_2} \preccurlyeq \overline{p_3}$	14, $\exists$-E
16	$p_1 \preccurlyeq \overline{p_2}$	8, $\wedge$-E
17	$\overline{p_2} \preccurlyeq \overline{p_3}$	15, $\wedge$-E
18	$p_1 \preccurlyeq \overline{p_3}$	16, 17, T4.1
19	$\exists p_3 \in PA(r_3) \bullet p_1 \preccurlyeq p_3$	18, $\exists$-I
20	$p_1 \in PA(r_1) \Rightarrow \exists p_3 \in PA(r_3) \bullet p_1 \preccurlyeq p_3$	6, 19, $\Rightarrow$-I
21	$\forall p_1 \in PA(r_1) \bullet \exists p_3 \in PA(r_3) \bullet p_1 \preccurlyeq p_3$	6, 20, $\forall$-I
22	$IP(PA(r_1)) \wedge IP(PA(r_3))$	(14), (27)
23	$IP(PA(r_1)) \wedge IP(PA(r_3)) \wedge \forall p_1 \in PA(r_1) \bullet \exists p_3 \in PA(r_3) \bullet p_1 \preccurlyeq p_3$	21, 22, $\wedge$-I
24	$PA(r_1) \preccurlyeq PA(r_3)$	23, (15)
25	$r_1 \preccurlyeq r_3$	24, D5.2
26	$r_1 \preccurlyeq r_2 \wedge r_2 \preccurlyeq r_3 \Rightarrow r_1 \preccurlyeq r_3$	1, 25, $\Rightarrow$-I

□

A.4. Proof of Corollary 5.4

Irreflexivity

Suppose arbitrary $r \in R$. We shall proceed with a proof by contradiction.

1	$r \in RConf(r)$	
2	$\exists p, p' \in PA^{\preccurlyeq}_{SoD}(r) \bullet p' \in PConf(p)$	1, D5.8
3	$\overline{p}, \overline{p'} \in PA^{\preccurlyeq}_{SoD}(r) \wedge \overline{p'} \in PConf(\overline{p})$	2, $\exists$-E
4	$PA^{\preccurlyeq}_{SoD}(r) \cap PConf(\overline{p}) \neq \emptyset$	3
5	$\overline{p} \in PA^{\preccurlyeq}_{SoD}(r)$	3, $\wedge$-E
6	$PA^{\preccurlyeq}_{SoD}(r) \cap PConf(\overline{p}) = \emptyset$	5, D5.5, D5.7
7	`false`	4, 6
8	$r \notin RConf(r)$	1, 7

Symmetry

Suppose arbitrary $r, r' \in R$.

1	$r \in RConf(r')$	
2	$\exists p \in PA^{\preccurlyeq}_{SoD}(r); p' \in PA^{\preccurlyeq}_{SoD}(r') \bullet$	1, D5.8

	$p' \in PConf(p)$	
3	$\overline{p} \in PA^{\preccurlyeq}_{SoD}(r) \wedge \overline{p'} \in PA^{\preccurlyeq}_{SoD}(r') \wedge$	2, $\exists$-E
	$\overline{p'} \in PConf(\overline{p})$	
4	$\overline{p'} \in PConf(\overline{p})$	3, $\wedge$-E
5	$\overline{p} \in PConf(\overline{p'})$	4, (39)
6	$\overline{p} \in PA^{\preccurlyeq}_{SoD}(r)$	3, $\wedge$-E
7	$\overline{p'} \in PA^{\preccurlyeq}_{SoD}(r')$	3, $\wedge$-E
8	$\exists p \in PA^{\preccurlyeq}_{SoD}(r); p' \in PA^{\preccurlyeq}_{SoD}(r') \bullet$	5, 6, 7, $\exists$-I
	$p \in PConf(p')$	
9	$r' \in RConf(r)$	8, D5.8
10	$r \in RConf(r') \Rightarrow r' \in RConf(r)$	1, 9, $\Rightarrow$-I

A.5. Proof of Theorem 7.1

Suppose arbitrary $u \in U; o, o' \in O; v, v' \in \mathbb{P}A \setminus \emptyset; op, op' \in Op$.

1	$(u, o, v) \in B \wedge rd \in v$	
2	$(u, o, v) \in B$	1, $\wedge$-E
3	$\exists s \in Session; r \in R; op \in Op \bullet$	2, (97)
	$u = user(s) \wedge r \in roles(s) \wedge \mathcal{A}(op) = v$	
	$\wedge (r, (o, op)) \in PA^{\preccurlyeq}_{SoD}$	
4	$u = user(\overline{s}) \wedge \overline{r} \in roles(\overline{s}) \wedge$	3, $\exists$-E
	$\mathcal{A}(\overline{op}) = v \wedge (\overline{r}, (o, \overline{op})) \in PA^{\preccurlyeq}_{SoD}$	
5	$u = user(\overline{s})$	4, $\wedge$-E
6	$roles(\overline{s}) \subseteq UA_{SoD}(u)$	5, (89)
7	$\overline{r} \in roles(\overline{s})$	6, $\wedge$-E
8	$\ell^R(\overline{r}) = f_C(u)$	7, (91)
9	$(\overline{r}, (o, \overline{op})) \in PA^{\preccurlyeq}_{SoD}$	4, $\wedge$-E
10	$(o, \overline{op}) \in PA^{\preccurlyeq}_{SoD}(\overline{r})$	9
11	$PA^{\preccurlyeq}_{Excl}(\overline{r}) \subseteq PA^{\preccurlyeq}(\overline{r})$	D5.4
12	$PA^{\preccurlyeq}_{SoD}(\overline{r}) \subseteq PA^{\preccurlyeq}_{Excl}(\overline{r})$	D5.5, D5.7
13	$PA^{\preccurlyeq}_{SoD}(\overline{r}) \subseteq PA^{\preccurlyeq}(\overline{r})$	11, 12
14	$(o, \overline{op}) \in PA^{\preccurlyeq}(\overline{r})$	9, 13
15	$(o, \overline{op}) \in PA(\overline{r}) \vee$	14, (32),
	$\exists p' \in PA(\overline{r}) \bullet (o, \overline{op}) \preccurlyeq p'$	$\forall$-E
16	$\quad (o, \overline{op}) \in PA(\overline{r})$	
17	$\quad \ell^R(\overline{r}) = \ell^P((o, \overline{op}))$	16, (26)
18	$\quad \ell^P((o, \overline{op})) \leq \ell^R(\overline{r})$	17, $\vee$-I
19	$\quad \exists p' \in PA(\overline{r}) \bullet (o, \overline{op}) \preccurlyeq p'$	
20	$\quad \overline{p'} \in PA(\overline{r}) \wedge (o, \overline{op}) \preccurlyeq \overline{p'}$	19, $\exists$-E
21	$\quad (o, \overline{op}) \preccurlyeq \overline{p'}$	20, $\wedge$-E
22	$\quad \overline{p'} \in PA(\overline{r})$	20, $\wedge$-E
23	$\quad \ell^R(\overline{r}) = \ell^P(\overline{p'})$	22, (26)
24	$\quad rd \in v$	1, $\wedge$-E
25	$\quad v = \{rd\} \vee v = \{rd, ap\}$	24
26	$\quad \mathcal{A}(\overline{op}) = v$	4, $\wedge$-E
27	$\quad \mathcal{A}(\overline{op}) = \{rd\} \vee \mathcal{A}(\overline{op}) = \{rd, ap\}$	25, 26
28	$\quad\quad \mathcal{A}(\overline{op}) = \{rd\}$	21, D4.2
29	$\quad\quad (\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$	
	$\quad\quad \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$	
	$\quad\quad (\ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$	
	$\quad\quad \mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$	
	$\quad\quad (\ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$	
	$\quad\quad \mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'}))$	
30	$\quad\quad\quad \ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$	
	$\quad\quad\quad \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
31	$\quad\quad\quad \ell^P((o, \overline{op})) = \ell^P(\overline{p'})$	30, $\wedge$-E
32	$\quad\quad\quad \ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$	
	$\quad\quad\quad \mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
33	$\quad\quad\quad \ell^P((o, \overline{op})) < \ell^P(\overline{p'})$	32, $\wedge$-E
34	$\quad\quad\quad \ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$	
	$\quad\quad\quad \mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
35	$\quad\quad\quad \mathcal{A}(\overline{op}) = \{ap\}$	34, $\wedge$-E
36	$\quad\quad\quad$ `false`	28, 35
37	$\quad \ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \vee$	29,
	$\quad \ell^P((o, \overline{op})) < \ell^P(\overline{p'})$	30–36,
		$\vee$-E
38	$\quad \ell^P((o, \overline{op})) \leq \ell^P(\overline{p'})$	37

Let us assume that $\overline{p'} = (\overline{o'}, \overline{op'})$ for some $\overline{o'} \in O$ and $\overline{op'} \in Op$. Continuing the proof:

39	$\quad\quad \mathcal{A}(\overline{op}) = \{rd, ap\}$	

40		$(\ell^P((o,\overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o,\overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o,\overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'}))$	21, D4.2
41	|	$\ell^P((o,\overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
42	|	$\ell^P((o,\overline{op})) = \ell^P(\overline{p'})$	41, $\wedge$-E
43	|	$\ell^P((o,\overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
44	|	$\mathcal{A}(\overline{op}) = \{rd\}$	43, $\wedge$-E
45	|	`false`	39, 44
46	|	$\ell^P((o,\overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
47	|	$\mathcal{A}(\overline{op}) = \{ap\}$	46, $\wedge$-E
48	|	`false`	39, 47
49		$\ell^P((o,\overline{op})) = \ell^P(\overline{p'})$	40, 41–48, $\vee$-E
50		$\ell^P((o,\overline{op})) = \ell^P(\overline{p'}) \vee$ $\ell^P((o,\overline{op})) < \ell^P(\overline{p'})$	49, $\vee$-I
51		$\ell^P((o,\overline{op})) \leq \ell^P(\overline{p'})$	50
52		$\ell^P((o,\overline{op})) \leq \ell^P(\overline{p'})$	27, 28–51, $\vee$-E
53		$\ell^P((o,\overline{op})) \leq \ell^R(\overline{r})$	23, 52
54		$\ell^P((o,\overline{op})) \leq \ell^R(\overline{r})$	15, 16–53, $\vee$-E
55		$\ell^P((o,\overline{op})) = f_O(o)$	D4.1
56		$f_O(o) \leq \ell^R(\overline{r})$	54, 55
57		$f_O(o) \leq f_C(u)$	8, 56
58		$f_O(o) \leq f_S(u)$	57, (2)
59		$(u,o,v) \in B \wedge rd \in v \Rightarrow f_O(o) \leq f_S(u)$	1, 58, $\Rightarrow$-I

A.6. Proof of Theorem 7.2

Proof of Predicate (98)
Suppose arbitrary $u \in U; o \in O; v \in \mathbb{P}A \setminus \emptyset$.

1		$(u,o,v) \in B \wedge ap \in v$	
2		$(u,o,v) \in B$	1, $\wedge$-E
3		$\exists s \in Session; r \in R; op \in Op \bullet$ $u = user(s) \wedge r \in roles(s) \wedge$ $(r,(o,op)) \in PA^{\preccurlyeq}_{SoD} \wedge$ $\mathcal{A}(op) = v$	2, (97)
4		$u = user(\overline{s}) \wedge \overline{r} \in roles(\overline{s})$ $\wedge(\overline{r},(o,\overline{op})) \in PA^{\preccurlyeq}_{SoD} \wedge$ $\mathcal{A}(\overline{op}) = v$	3, $\exists$-E
5		$u = user(\overline{s})$	4, $\wedge$-E
6		$roles(\overline{s}) \subseteq UA_{SoD}(u)$	5, (89)
7		$\overline{r} \in roles(\overline{s})$	4, $\wedge$-E
8		$\ell^R(\overline{r}) = f_C(u)$	7, (91)
9		$(\overline{r},(o,\overline{op})) \in PA^{\preccurlyeq}_{SoD}$	4, $\wedge$-E
10		$(o,\overline{op}) \in PA^{\preccurlyeq}_{SoD}(\overline{r})$	9
11		$PA^{\preccurlyeq}_{Excl}(\overline{r}) \subseteq PA^{\preccurlyeq}(\overline{r})$	D5.4
12		$PA^{\preccurlyeq}_{SoD}(\overline{r}) \subseteq PA^{\preccurlyeq}_{Excl}(\overline{r})$	D5.5, D5.7
13		$PA^{\preccurlyeq}_{SoD}(\overline{r}) \subseteq PA^{\preccurlyeq}(\overline{r})$	11, 12
14		$(o,\overline{op}) \in PA^{\preccurlyeq}(\overline{r})$	9, 13
15		$(o,\overline{op}) \in PA(\overline{r}) \vee$ $\exists p' \in PA(\overline{r}) \bullet (o,\overline{op}) \preccurlyeq p'$	14, (32), $\forall$-E
16	|	$(o,\overline{op}) \in PA(\overline{r})$	
17	|	$\ell^R(\overline{r}) = \ell^P((o,\overline{op}))$	16, (26)
18	|	$\ell^P((o,\overline{op})) \geq \ell^R(\overline{r})$	17, $\vee$-I
19	|	$\exists p' \in PA(\overline{r}) \bullet (o,\overline{op}) \preccurlyeq p'$	
20	|	$\overline{p'} \in PA(\overline{r}) \wedge (o,\overline{op}) \preccurlyeq \overline{p'}$	19, $\exists$-E
21	|	$(o,\overline{op}) \preccurlyeq \overline{p'}$	20, $\wedge$-E
22	|	$\overline{p'} \in PA(\overline{r})$	20, $\wedge$-E
23	|	$\ell^R(\overline{r}) = \ell^P(\overline{p'})$	22, (26)
24	|	$ap \in v$	1, $\wedge$-E
25	|	$v = \{ap\} \vee v = \{rd, ap\}$	24

26	$\mathcal{A}(\overline{op}) = v$	4, $\wedge$-E
27	$\mathcal{A}(\overline{op}) = \{ap\} \vee \mathcal{A}(\overline{op}) = \{rd, ap\}$	25, 26

Let $\overline{p'} = (\overline{o'}, \overline{op'})$ for some $o' \in O$ and $op' \in Op$. Continuing the proof:

28	$\mathcal{A}(\overline{op}) = \{ap\}$	
29	$(\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'}))$	21, D4.2
30	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
31	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'})$	30, $\wedge$-E
32	$\ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
33	$\mathcal{A}(\overline{op}) = \{rd\}$	32, $\wedge$-E
34	`false`	28, 33
35	$\ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
36	$\ell^P((o, \overline{op})) > \ell^P(\overline{p'})$	35, $\wedge$-E
37	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \vee$ $\ell^P((o, \overline{op})) > \ell^P(\overline{p'})$	30–36, $\vee$-E
38	$\ell^P((o, \overline{op})) \geq \ell^P(\overline{p'})$	37
39	$\mathcal{A}(\overline{op}) = \{rd, ap\}$	
40	$(\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})) \vee$ $(\ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'}))$	21, D4.2
41	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
42	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'})$	41, $\wedge$-E
43	$\ell^P((o, \overline{op})) < \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{rd\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
44	$\mathcal{A}(\overline{op}) = \{rd\}$	43, $\wedge$-E
45	`false`	39, 44
46	$\ell^P((o, \overline{op})) > \ell^P(\overline{p'}) \wedge$ $\mathcal{A}(\overline{op}) = \{ap\} \wedge \mathcal{A}(\overline{op}) \subseteq \mathcal{A}(\overline{op'})$	
47	$\mathcal{A}(\overline{op}) = \{ap\}$	46, $\wedge$-E
48	`false`	39, 47
49	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'})$	40–48, $\vee$-E
50	$\ell^P((o, \overline{op})) = \ell^P(\overline{p'}) \vee$ $\ell^P((o, \overline{op})) > \ell^P(\overline{p'})$	49, $\vee$-I
51	$\ell^P((o, \overline{op})) \geq \ell^P(\overline{p'})$	50
52	$\ell^P((o, \overline{op})) \geq \ell^P(\overline{p'})$	27–51, $\vee$-E
53	$\ell^P((o, \overline{op})) \geq \ell^R(\overline{r})$	23, 52
54	$\ell^P((o, \overline{op})) \geq \ell^R(\overline{r})$	16–53, $\vee$-E
55	$\ell^P((o, \overline{op})) = f_O(o)$	D4.1
56	$f_O(o) \geq \ell^R(\overline{r})$	54, 55
57	$f_O(o) \geq f_C(u)$	8, 56
58	$f_O(o) \geq f_S(u)$	57, (2)
59	$(u, o, v) \in B \wedge ap \in v \Rightarrow f_O(o) \geq f_S(u)$	1, 58, $\Rightarrow$-I

A.6.1. Proof of predicate (99)
Suppose arbitrary $u \in U; o, o' \in O; v, v' \in \mathbb{P}A \setminus \emptyset$.

1	$(u, o, v) \in B \wedge ap \in v \wedge$ $(u, o', v') \in B \wedge rd \in v'$	
2	$(u, o, v) \in B \wedge ap \in v$	1, $\wedge$-E
3	$(u, o, v) \in B$	2, $\wedge$-E
4	$\exists s \in Session; r \in R; op \in Op \bullet$ $u = user(s) \wedge r \in roles(s) \wedge$ $(r, (o, op)) \in PA^{\preccurlyeq}_{SoD} \wedge \mathcal{A}(op) = v$	3, (97)
5	$u = user(\overline{s}) \wedge \overline{r} \in roles(\overline{s}) \wedge$ $(\overline{r}, (o, \overline{op})) \in PA^{\preccurlyeq}_{SoD} \wedge \mathcal{A}(\overline{op}) = v$	4, $\exists$-E

A reasoning identical to that provided through lines 5-54 of the proof of predicate (98) yields:

6	$\ell^P((o,\overline{op})) \geq \ell^R(\bar{r})$	5
7	$(u,o',v') \in B \wedge \mathcal{A}(op') = \{rd\}$	1, $\wedge$-E
8	$(u,o',v') \in B$	7, $\wedge$-E
9	$\exists s \in Session; r \in R; op \in Op \bullet$ $u = user(s) \wedge r \in roles(s) \wedge$ $(r,(o',op)) \in PA^{\preccurlyeq}_{SoD} \wedge \mathcal{A}(op) = v$	8, (97)
10	$u = user(\bar{s}) \wedge \bar{\bar{r}} \in roles(\bar{s}) \wedge$ $(\bar{\bar{r}},(o',\overline{\overline{op}})) \in PA^{\preccurlyeq}_{SoD} \wedge \mathcal{A}(\overline{\overline{op}}) = v$	9, $\exists$-E

A reasoning identical to that provided through lines 5–54 of the proof of Theorem 7.1 yields:

11	$\ell^P((o',\overline{\overline{op}})) \leq \ell^R(\bar{\bar{r}})$	10
12	$u = user(\bar{s})$	5, $\wedge$-E
13	$\bar{r} \in roles(\bar{s})$	5, $\wedge$-E
14	$\ell^R(\bar{r}) = f_C(u)$	12, 13, (91)
15	$u = user(\bar{\bar{s}})$	10, $\wedge$-E
16	$\bar{\bar{r}} \in roles(\bar{\bar{s}})$	10, $\wedge$-E
17	$\ell^R(\bar{\bar{r}}) = f_C(u)$	15, 16, (91)
18	$\ell^R(\bar{\bar{r}}) = \ell^R(\bar{r})$	14, 17
19	$\ell^P((o',\overline{\overline{op}})) \leq \ell^P((o,\overline{op}))$	6, 11, 18
20	$f_O(o') \leq f_O(o)$	19, (10)
21	$((u,o,v) \in B \wedge ap \in v \wedge$ $(u,o',v') \in B \wedge rd \in v') \Rightarrow$ $f_O(o') \leq f_O(o)$	1, 20, $\Rightarrow$-I

A.7. Proof of Theorem 7.3

Suppose arbitrary $u \in U; o,o' \in O; v,v' \in \mathbb{P}A \setminus \emptyset; op,op' \in Op$. We shall proceed with a proof by contradiction.

1	$(u,o,v) \in B \wedge \mathcal{A}(op) = v \wedge \mathcal{A}(op') = v' \wedge$ $(u,o',v') \in B \wedge (o',op') \in PConf((o,op))$	
2	$(u,o,v) \in B$	1, $\wedge$-E
3	$\exists s \in Session; r \in R; op \in Op \bullet$ $u = user(s) \wedge r \in roles(s) \wedge \mathcal{A}(op) = v \wedge$ $(r,(o,op)) \in PA^{\preccurlyeq}_{SoD}$	2, (97)
4	$u = user(\bar{s}) \wedge \bar{r} \in roles(\bar{s}) \wedge \mathcal{A}(\overline{op}) = v \wedge$ $(\bar{r},(o,\overline{op})) \in PA^{\preccurlyeq}_{SoD}$	3, $\exists$-E
5	$\mathcal{A}(\overline{op}) = v$	4, $\wedge$-E
6	$\mathcal{A}(op) = v$	1, $\wedge$-E
7	$\mathcal{A}(op) = \mathcal{A}(\overline{op})$	5, 6
8	$(u,o',v') \in B$	1, $\wedge$-E
9	$\exists s \in Session; r \in R; op \in Op \bullet$ $u = user(s) \wedge r \in roles(s) \wedge \mathcal{A}(op) = v' \wedge (r,(o',op)) \in PA^{\preccurlyeq}_{SoD}$	5, (97)
10	$u = user(\bar{s}) \wedge \bar{\bar{r}} \in roles(\bar{s}) \wedge$ $\mathcal{A}(\overline{\overline{op}}) = v' \wedge (\bar{\bar{r}},(o',\overline{\overline{op}})) \in PA^{\preccurlyeq}_{SoD}$	9, $\exists$-E
11	$\mathcal{A}(\overline{\overline{op}}) = v'$	10, $\wedge$-E
12	$\mathcal{A}(op') = v'$	1, $\wedge$-E
13	$\mathcal{A}(op') = \mathcal{A}(\overline{\overline{op}})$	11, 12
14	$(o',op') \in PConf((o,op))$	1, $\wedge$-E
15	$(o',\overline{\overline{op}}) \in PConf((o,\overline{op}))$	14

From line 7, $\overline{op}$ and op have the same effect on o. Similarly, from line 13, $\overline{\overline{op}}$ and op' have the same effect on o'. It follows that, if (o,op) and (o',op') are conflicting (line 14), then so are $(o,\overline{op})$ and $(o',\overline{\overline{op}})$. Continuing the proof:

16	$(\bar{r},(o,\overline{op})) \in PA^{\preccurlyeq}_{SoD}$	4, $\wedge$-E
17	$(o,\overline{op}) \in PA^{\preccurlyeq}_{SoD}(\bar{r})$	16
18	$(\bar{\bar{r}},(o',\overline{\overline{op}})) \in PA^{\preccurlyeq}_{SoD}$	10, $\wedge$-E
19	$(o',\overline{\overline{op}}) \in PA^{\preccurlyeq}_{SoD}(\bar{\bar{r}})$	18
20	$(o,\overline{op}) \in PA^{\preccurlyeq}_{SoD}(\bar{r}) \wedge$ $(o',\overline{\overline{op}}) \in PA^{\preccurlyeq}_{SoD}(\bar{\bar{r}}) \wedge$ $(o',\overline{\overline{op}}) \in PConf((o,\overline{op}))$	15, 17, 19, $\wedge$-I
21	$\exists p \in PA^{\preccurlyeq}_{SoD}(\bar{r}); p' \in PA^{\preccurlyeq}_{SoD}(\bar{\bar{r}}) \bullet$ $p' \in PConf(p)$	20, $\exists$-I
22	$\bar{\bar{r}} \in RConf(\bar{r})$	21, D5.8
23	$u = user(\bar{s}) \wedge \bar{r} \in roles(\bar{s})$	4, $\wedge$-E
24	$u = user(\bar{\bar{s}}) \wedge \bar{\bar{r}} \in roles(\bar{\bar{s}})$	10, $\wedge$-E
25	$\bar{r},\bar{\bar{r}} \in \bigcup_{s \in user^{-1}(u)} roles(s)$	23, 24
26	$\bar{\bar{r}} \notin RConf(\bar{r})$	25, C6.2
27	`false`	22, 26

28 $\neg((u, o, v) \in B \wedge \mathcal{A}(op) = v \wedge \mathcal{A}(op') = v' \wedge (u, o', v') \in B \wedge (o', op') \in PConf((o, op)))$ 1, 27

29 $(u, o, v) \in B \wedge \mathcal{A}(op) = v \wedge \mathcal{A}(op') = v' \wedge (u, o', v') \in B \Rightarrow (o', op') \notin PConf((o, op))$ 28