Data flow security in Role-Based Access Control

Luigi Logrippo

Université du Québec en Outaouais,
Dépt. d'informatique et d'ingénierie
CP 1250, Succ. Hull
Gatineau, Québec, Canada J8X 3X7

University of Ottawa
SEECS: Electrical Eng & Computer Sc.
75 Laurier Ave. E
Ottawa, Ontario, Canada K1N 6N5

luigi@uqo.ca

Abstract. We show how data security concepts such as data flow, secrecy (or confidentiality) and integrity can be defined for RBAC, Role-Based Access Control. In contrast to the prevailing literature that uses a lattice model to express such concepts, we demonstrate the use of a partial order model that is more general. This is done by using the concepts of "partial order of equivalence classes" and of "security labels" that can be associated with RBAC subjects and objects and determine their mutual data flows, as well as their secrecy and integrity properties. Our model allows to reason on RBAC configurations with different assignments of roles to subjects. On the converse, we demonstrate a method for obtaining RBAC configurations from data security requirements or security label assignments. These results are supported by a proof showing that three methods for defining data flow: by access control matrices, by labels and by roles, are equivalent and mutually convertible. We show how RBAC state changes, or "reconfigurations" can be defined in this framework, and what are the effects of elementary reconfigurations on data flow, secrecy and integrity of data.

Keywords: RBAC, Role-based access control, data flow control; data security; data secrecy; data confidentiality; data integrity; multi-level access control; mandatory access control; security labeling; design for security.

1. Introduction

Role-Based Access Control (RBAC) is a very well-established access control method. In its many variations and adaptations, it is used in many organizations and systems. With extensions, it is being considered for use in the Cloud and in the Internet of things (IoT) [2,6,11,26,32,35,36,37,44,46,47].

RBAC can be used to protect resources of different types. In this paper, we concentrate on the use of RBAC for protecting data objects with respect to reading and writing operations and resulting data flows. Given a network of entities, which can be data objects (data sets, databases, files) and users represented by subjects, each with a set of role-based permissions, and with the hypothesis of transitivity (that a subject that reads some data can pass them on), where can data end up, what are the levels of secrecy and integrity of subjects and objects? How do these change if the permissions change? These questions are part of the general questions of data security and privacy. Landwehr [25] and Kozyri et al. [23], among others, note the relationship between privacy violations and improper data flow. However, the literature on the specific subject of this paper is limited to essentially three papers [34,17,36], as we will see in Section 6.

Research on RBAC data flow security has for long followed the theory that secure data networks must be defined in terms of lattice structures, a theory contrasted by the fact that RBAC can define non-lattice networks [34]. We have shown in our previous papers [27,28] that this established theory can be generalized to a theory that is valid for all networks,

whether they are lattice-structured or not, given the fact that for any network of communicating entities there exists a partial order of equivalence classes of entities, to which security labels can be assigned.

We have also shown that a data flow security theory based on partial orders has the following advantages with respect to the established theory based on lattices:

- Only actually used entities and security labels need to be present in partial orders; further, by excluding certain labels, it is possible to exclude the possibility of certain subjects or objects being able to access certain data combinations, corresponding to situations of conflict such as those addressed by familiar 'Brewer-Nash' or 'Chinese wall' mechanisms [9,38].
- The theory is closed under reconfigurations, in the sense that changes in the permissions will lead from partial orders to partial orders (see Sect. 5).
- Well-known, efficient algorithms exist to make the theory applicable [42].
- Implementation methods exist for this theory, namely one using SDN, Software Defined Networking, which is described in [43].

Using these results, we show in this paper that:

a) It is possible to efficiently analyze the data flows defined by RBAC configurations, to see what paths data can take, from the entities that initially can have them, to all the entities that potentially can get them, directly or indirectly.
b) Given the results of this analysis, it is possible to determine the levels of secrecy (or confidentiality) and integrity of the RBAC subjects and objects.
c) Given a specified data flow in a network of entities, it is possible to efficiently generate an RBAC configuration that defines it. The specification can be in terms of data flow requirements or labels.
d) The effects of RBAC reconfigurations for secrecy and integrity can be evaluated.

In our previous work [42] we have mentioned that our partial order data flow theory is applicable to RBAC. This paper explains this claim, by explicitly using RBAC definitions and examples.

To this end, we proceed as follows:

After restating the essence of our previous work in Sect.2, in Sect. 3 we define arbitrary bipartite networks, consisting of subjects and objects, with a Channel relation, which represents permissions to read or write. Then we define RBAC networks, which are networks as just described, but defined in terms of RBAC subjects, objects, roles and permissions, in a given RBAC *configuration*. We then show that any arbitrary bipartite network can be defined as an RBAC network. We will see that such networks define partial orders of equivalence classes, enabling the definition of secrecy and integrity levels, as well as security labeling. Several examples follow in Sect. 4, including one where we show how it is possible to construct RBAC networks starting from data flow requirements. In Sect. 5 we make an analysis of RBAC reconfigurations, to see how changes in roles and permissions can affect changes in secrecy and integrity of subjects and objects. In Sect. 6 we review the literature, comparing it with our results. Section 7 concludes the paper and underlines contributions.

2. Background

This section is a quick recapitulation of definitions and results that have already been published [28,30,42] or submitted and available [29]. We start with some standard order-theoretical concepts that will be used throughout the paper. A binary relation in a set of *entities* is [8,19]:

- A *preorder* (also called *quasi-order*), if it is reflexive and transitive.
- A *partial order* if it is reflexive, transitive and antisymmetric. For two entities *x, y* in a partial order one of the following is true: *x dominates y* ($y \sqsubseteq x$) or *x is dominated by y* ($x \sqsubseteq y$), or they are *incomparable*. $x \sqsubseteq y$ and $y \sqsubseteq x$ iff $x=y$. We say that *x strictly dominates y* and we write $y \sqsubsetneq x$ if $y \sqsubseteq x$ and $x \neq y$. A finite partial order has maximal and minimal elements: an element is maximal (minimal) if no element strictly dominates it (is strictly dominated by it).
- A *total order* if it is a partial order where any two entities are comparable.
- A *lattice* if it is a partial order and for any two entities *x,y* there exist unique *z,w* such that *z,* the *join (w,* the *meet)* dominates (is dominated by) both *x* and *y* and is dominated by (dominates) all the other entities that have the same property.

The concept of lattice is not used in our theory but is mentioned since the established theory uses it, following [14,39].

Definition 1. A *data network* (or simply a *network* henceforth) is a set of entities with a binary relation *Channel*. Each entity *x* has a *Name(x),* which uniquely identifies it in the network, it is a character string with capital initials. An *access control matrix* is a Boolean matrix representation of a *Channel* relation.

Letters *x, y, z,* possibly with primes or subscripts, will be used as variables for entities. *Channel* models a permission to execute unidirectional data transfers, such as reading or writing, receiving or sending.

Definition 2. The binary relation *CanFlow* (written *CF*) is the reflexive, transitive closure of the relation *Channel*.

Note that *CF* is a preorder.

Definition 3. Entities *x* and *y* are *equivalent* if *CF(x,y)* and *CF(y,x)*. An equivalence class of entities including *x,y,*…. is denoted *[x,y,…]*.

Definition 4. Two networks are *equivalent* if they have the same set of entities with the same names and, for all entities *x* and *y*, *CF(x,y)* in one iff *CF(x,y)* in the other.

Hence, equivalent networks may have different *Channel* relations.

According to order theory [8], the preorder *CF* defines a *partial order of equivalence classes* on the set of entities, such that *CF(x,y) iff* $[x] \sqsubseteq [y]$.

Definition 5. Let *Names* be the set of all names of entities in a network. We associate with each equivalence class *[x]* a set, the *label* of *[x]* or *Lab([x]),* which is a subset of *Names.* For each *[x]*, let $Ownlabel([x]) = \{Name(y) \mid y \in [x]\}$. For each *[x]*, let $Lab([x]) = \cup\{Ownlabel([y]) \mid [y] \sqsubseteq [x]\}$.

Definition 6. For an entity *x*, *Lab(x)=Lab([x]).*

Note the following equivalent definition:

Definition 6'.For an entity *x*, *Lab(x)={Name(y)|CF(y,x)}.*

Labels can be efficiently calculated starting from the bottom of the partial order of equivalence classes. According to Def. 5, an equivalence class at the bottom gets a label that is the set of the names of entities in the class. An equivalence class that dominates others gets a label that is the union of the names of entities in the class with the labels of the classes it dominates, see the examples given later in the paper. Labels implement a simple concept of *provenance*, in the sense that each entity's label includes the names of all entities whose data it can contain, with the given *CF* relation.

Theorem 1. For any network, *CF(x,y) iff [x]⊑[y] iff Lab(x)⊆Lab(y).*

Proof. The first part has been given earlier as an order-theoretical result. The second part can be easily checked by Def. 5.

Theorem 2. For any network, given any of:
a) a *CF* relation,
b) a partial order of equivalence classes of entities,
c) an assignment of labels to entities in the set,
the other two, satisfying Theorem 1, can be calculated with linear time or polynomial time algorithms.

Proof. To go from a) to b), note that equivalence classes constitute *strongly connected components* in the graph of the *CF* relation, and so we can use standard linear time algorithms to find the corresponding partial order of components [42]. To go from b) to c), given the partial order, calculate the labels using Def. 5 and moving up from the minimal elements of the partial order found in the previous step. This algorithm is also linear-time. To go from c) to a), the *CF* relation can be calculated from the set of labels using Theorem 1: *CF(x,y) iff Lab(x)⊆Lab(y).* We need an algorithm for checking for set inclusion. Such algorithms are known to have the same complexity as sorting algorithms: first sort the two labels and then compare their elements one by one.

We will not elaborate on these concepts and proofs, since they are introduced in [28] and more formally proved in [29]. *Lab(x)⊆Lab(y)* is the label dominance relation in established data security theory. Theorem 1 says that, consistently with established theory, data flow in our networks according to the dominance relation. Established theory limits this principle to dominance in lattice-formed networks, but the principle can be generalized to any network [28,29]. Theorem 2 states the 'efficiency' claim of our method, as defined in [1]. We will not insist on this claim, since a previous paper [42] has examined the subject in detail, both theoretically and by simulation. In this paper, we will see examples of applications of these concepts.

The following definitions can apply just as well to entities as to equivalence classes of entities.

Definition 7. We say that an equivalence class *[x]* is *more secret* than an equivalence class *[y]* if *[y]⊑[x]*; in this case, *Lab(y) ⊊ Lab(x)* and we also say that *x* is more secret than *y*. We say that an equivalence class *[x]* has *more integrity* than an equivalence class *[y]* if *[x] ⊑[y]*; in this case, *Lab(x) ⊊ Lab(y)* and we also say that *x* has more integrity than *y*. If *[x]* is a maximal (minimal) element in a partial order, we say that *[x]* or *x* is of *maximum secrecy and minimum integrity (maximum integrity and minimum secrecy).*

In the case of maximum secrecy, there are no outgoing flows from elements of *[x],* in the of maximum integrity there are no incoming flows to them. A justification of the second definition in terms of the literature is given in our Literature Review (Sect. 6).

Note that our concept of label is in agreement with the traditional one of labels such as *TopSecret*, *Secret*, … *,Public*, etc. These are names for equivalence classes of entities, that can be used instead of, or together with, labels as in Def. 5 in order to obtain shorter labels. See [28,29] and the example in Sect. 4.4.

3. Application of the theory to RBAC

3.1 Bipartite networks and RBAC configurations

The definitions of RBAC are well-known and some of the most cited references are [15,16] We follow the formulation in [16], with the following limitations:

- The objects considered *(OBS)* are data-carrying entities such as files or databases;
- the subjects *(SUBS)* are entities that can read or write objects; they can also carry data obtained from objects by the effect of reading operations or passed to them by their *users*;
- the operations *(OPS)* considered are of reading and writing objects;
- the permissions *(PRMS)* are for subjects to read from objects *(CanRead or CR)* and subjects to write to objects *(CanWrite* or *CW)*; these are *Channel* relations, see below;
- we do not consider explicitly *users* in our formulation because users can generate data flows only when they activate subjects (also called *sessions*) with subsets of their roles; therefore the assignment of subjects to users is not considered in our configurations;
- we concentrate on $RBAC_0$, although some considerations on inheritance and constraints will be presented later.

We start with a generic definition:

Definition 8. A *bipartite data network* (or simply *bipartite network* henceforth) is a network of named entities that are partitioned into two subsets *SUBS* and *OBS*. The relation *Channel* is a subset of $OBS \times SUBS \cup SUBS \times OBS$. As all networks, bipartite networks can be defined by access control matrices.

By this definition, the *CF* relation is defined for bipartite networks and the theory of Sect. 2 applies. In particular, equivalence classes and labels are defined for subjects and objects in such networks, according to the definitions of Sect. 2.

We will use the letter *s* (resp. *o*) with primes or subscripts for variables denoting members of *SUBS* (resp. *OBS*). The names of entities are arbitrary strings of characters with capital initials. In much of this paper, names for subjects will be *S1, S2, …* and for objects *O1, O2, ….* Also roles, see below, will have names that will often be written *R1, R2, ….*Variables for roles will be noted by the letter *r* with primes and subscripts.

Property 1. In bipartite networks:

1) *CF(s,s')* iff there is *o* such that *CF(s,o)* and *CF(o,s')*
2) *CF(o,o')* iff there is *s* such that *CF(o,s)* and *CF(s,o')*

Proof. By the fact that there cannot be channels between subjects or objects.

RBAC *systems* are sets of RBAC definitions that can evolve over time by administrative or user action that take them from configuration to configuration. In a RBAC *configuration*, all RBAC definitions are fixed. Changes in the definitions will be called *reconfigurations*. RBAC *networks* are defined for RBAC configurations. The RBAC network for a RBAC configuration is a bipartite network that has the same sets *SUBS* and *OBS* as the RBAC configuration, and a *Channel* relation defined according to RBAC definitions in the configuration, as described below. Thus a RBAC configuration defines a RBAC network and an RBAC network defines an RBAC configuration by Construction 1 below.

Definition 9. The RBAC set *OPS* is defined as $OPS=\{CR,CW\}$.

Definition 10. In a bipartite network, we define the two relations *CR* and *CW* on $SUBS \times OBS$ as follows [24]:

a) *CR(s,o)* iff *Channel(o,s)* and
b) *CW(s,o)* iff *Channel(s,o).*

The RBAC set *PRMS* is the set of such relations.

In RBAC [16], the set of permissions for subjects according to their roles is defined through several expressions. There were reasons for how these definitions were formulated, but we simplify them here by saying that in each configuration, there is an assignment of roles to subjects and an assignment of permissions to roles, which leads to an assignment of permissions to subjects.

Definition 11. We use then three functions, which we consider to be defined for each RBAC configuration (let $\mathbb{P}$ denote the set of all subsets of a set):

- $SR:SUBS \rightarrow \mathbb{P}(ROLES)$ assigns sets of activated roles to subjects;
- $PA:ROLES \rightarrow \mathbb{P}(PRMS)$ assigns sets of permissions to roles;
- $SP:SUBS \rightarrow \mathbb{P}(PRMS)$, composition of *SR* and *PA*, assigns sets of permissions to subjects according to their activated roles.

Definition 12. An *RBAC network R* for an RBAC configuration is a bipartite network that has the same sets *SUBS* and *OBS* as the RBAC configuration and where:

1) *Channel(o,s)* or *CR(s,o)* is true in *R* iff $<CR,o> \in SP(s)$ is *true* in the RBAC configuration;
2) *Channel(s,o)* or *CW(s,o)* is true iff $<CW,o> \in SP(s)$ is *true* in the RBAC configuration;
3) *Channel* and the relations *CR, CW* are *false* for all other pairs of entities in *R.*

3.2 Bipartite networks, bipartite partial orders, and RBAC networks

The following theorem shows that any bipartite network or access control matrix can be defined as an equivalent RBAC network corresponding to an RBAC configuration, thus RBAC is in this sense a 'complete' formalism. This result has already been proved with other definitions, see literature review. We provide our own proof, and in doing so we present Construction 1, which will be used later.

Theorem 3. For any bipartite network *N*, there is an equivalent RBAC network *R*.

Proof. We provide a construction of roles and permissions that, given a bipartite network, leads to the desired *CF* relation in the RBAC network *R* for the configuration. Let *N* be the bipartite network. The equivalent RBAC network *R* will have the same sets *SUBS* and *OBS* as

N, with the same names. Construction 1 is used to determine what the roles and permissions are.

Construction 1. From the *Channel* relation of the bipartite network, calculate the *CF* relation, the equivalence classes and then the labels for the subjects and objects in *N* (Defs. 1 to 6). For each label *l*, define a set of permissions *Pr(l)* as follows:

1) $<CR,o> \in Pr(l)$ for all and only objects *o* such that $Lab(o) \subseteq l$ in *N*
2) $<CW,o> \in Pr(l)$ for all and only objects *o* such that $l \subseteq Lab(o)$ in *N*

Further, we define a role *Rl(l)* such that $PA(Rl(l))=Pr(l)$ and for every subject *s* such that $Lab(s)=l$, let $SR(s)=\{Rl(l)\}$.

In other words, labels identify positions in the partial order. A subject gets a role that permits it to read all and only the objects it dominates, and write all and only the objects by which it is dominated. The permissions and role calculations need to be done only for the labels that apply to subjects.

We should check that *CF(x,y)* in *N* iff *CF(x,y)* in *R*.

To see *CF(x,y)* in *N* implies *CF(x,y)* in *R*, suppose *CF(x,y)* in *N*. Then by Theorem 1 $Lab(x) \subseteq Lab(y)$ in *N*. Let us consider all the cases using *s, s', o, o'* for *x* and *y:*

a) *CF(s,o)* in *N*. Then $Lab(s) \subseteq Lab(o)$. Let $Lab(s)=l$. By Construction 1, $<CW,o> \in Pr(l)$ and $SR(s)=\{Rl(l)\}$ and $<CW,o> \in SP(s)$ is true in the RBAC configuration and so by Def. 12.2) *Channel(s,o)* is true in *R*, hence *CF(s,o)* is true in *R*.
b) *CF(o,s)* in *N*. Similarly, $Lab(o) \subseteq Lab(s)$ and $<CR,o>$ is a permission in the role of *s* and so *Channel(o,s)* is true in *R*.
c) *CF(s,s')* in *N*. Then by Property 1 there must be an object *o* such that *CF(s,o)* and *CF(o,s')* and the reasoning reduces to cases a) and b). There must be channels from *s* to *o* and from *o* to *s'* in *N* and *R*, so *CF(s,s')* in *R*.
d) *CF(o,o')* in *N*. By a similar reasoning (Property 1), *CF(o,o')* in *R*.

To see that *CF(x,y)* in *R* implies *CF(x,y)* in *N*, note that the *CF* relation in *R* is the transitive closure of the *Channel* relation in *R*. The latter is defined in terms of the *CR* and *CW* permissions in the RBAC configuration. By Construction 1, *CR* (or *CW*) permissions are in a role for a subject *s* on an object *o* in *R* iff $Lab(o) \subseteq Lab(s)$ in *N* (or $Lab(s) \subseteq Lab(o)$ in *N*), which, by Theorem 1, implies that *CF(x,y)* in *N*. Hence no additional flows can exist in *R* with respect to *N*. This completes the proof of Theorem 3.

Definition 13. We say that a partial order of equivalence classes on a set of subjects and objects (the entities), is *bipartite* if the following are true:

a) For any two *s* and *s'* such that $Lab(s) \subseteq Lab(s')$ there is an *o* such that $Lab(s) \subseteq Lab(o) \subseteq Lab(s')$.
b) For any two *o* and *o'* such that $Lab(o) \subseteq Lab(o')$ there is an *s* such that $Lab(o) \subseteq Lab(s) \subseteq Lab(o')$.

Theorem 4. For every bipartite network *N* there is a bipartite partial order *P* such that for entities *x* and *y*, *CF(x,y)* in *N* iff $Lab(x) \subseteq Lab(y)$ in *P* and vice-versa.

Proof. The label generation method of Section 2 gives us a *P* for each *N*. For the converse, given a *P*, an *N* can be constructed as follows. Suppose that $Lab(x) \subseteq Lab(y)$, then considering all cases where *x* and *y* can be subjects or objects, we have:

a) $Lab(s) \subseteq Lab(o)$ in *P*. Then we define *Channel(s,o)* (or *CW(s,o)*) in *N*.

b) $Lab(o) \subseteq Lab(s)$ in P. Then we define $Channel(o,s)$ (or $CR(s,o)$) in N.
c) $Lab(s) \subseteq Lab(s')$ in P. Then by Def. 11.a) there is an o such that $Lab(s) \subseteq Lab(o) \subseteq Lab(s')$. We define $Channel(s,o)$ and $Channel(o,s')$ (or $CW(s,o)$ and $CR(s',o)$) in N.
d) $Lab(o) \subseteq Lab(o')$ in P. Then by Def. 11.b) there is an s such that $Lab(o) \subseteq Lab(s) \subseteq Lab(o')$. We define $Channel(o,s)$ and $Channel(s,o')$ (or $CR(s,o)$ and $CW(s,o')$) in N.

Corollary 1. For a set of subjects and objects (the entities), given any of:
a) an access control matrix,
b) a bipartite network,
c) a CF relation,
d) a mapping Lab defined for the entities,
e) a bipartite partial order of equivalence classes,
f) an RBAC configuration for the entities,
the other five, all defining equivalent networks, can be calculated with linear time or polynomial-time algorithms.

Proof. a) and b) are equivalent representations, one in relational and the other in Boolean form. A transitive closure algorithm, of cubic complexity [1] takes us from b) to c). From c) Lab is calculated as described in Section 2, and this gives d). We can go from d) to e) by using the construction of Theorem 4. This construction gives us also a bipartite network, which takes us to f) by using Construction 1. This requires the use of a subset-checking algorithm for labels, which, as mentioned above, has the same complexity as sorting. Once we have an RBAC configuration, we have the function SP which tells which channels exist between subjects and objects (Def. 12) and we can then construct the access control matrix a). The complexity of this last algorithm is $/SUBS/ \times /OBS/$. Of course, other methods can be found to go directly between any two of a) to f).

We take a mapping Lab, together with the relation of Theorem 1, to define a *bipartite MLS system (Multilayer Access Control System*, also called *MAC* or *Mandatory Access Control system*). As a consequence, we have an efficient construction to go from any bipartite MLS system to an equivalent RBAC configuration and vice-versa.

It should be noted that our construction of roles is only a formal device and is not necessarily one that would be useful in an organizational context, where roles are assigned to subjects according to their functions in the organization, see Ferraiolo [16] and the abundant literature on role mining.

3.3 Secrecy (or confidentiality) and integrity in RBAC

Def. 7 applies to RBAC entities. We consider that it is possible to define secrecy and integrity for both subjects and objects, on the intuition that the fact that a subject can write is equivalent for secrecy to the fact that an object can be read, and similarly for integrity. We do not attempt in this paper to compare secrecy and integrity of entities, apart from telling which entities have maximal ones. Different criteria could be used to define absolute levels of secrecy or integrity, e.g. by the number of outgoing or incoming edges for an entity, by the number of entities that strictly dominate or are dominated by an entity, by the shortest path from an entity to ones of maximal secrecy or integrity, and there may be other possibilities. More precise definitions may depend on the needs of specific application domains.

4. Examples

4.1 Graphical notation, terminology and first example

We write $s(r_1,...,r_n)$ if subject s has, in the current configuration, roles $r_1...r_n$. We combine this notation with the label notation to write $s(r_1,...,r_n):\{y_1,...,y_n\}$ to refer to both the roles and the label of s. We have seen how the label of a subject in an RBAC network, derived from a configuration, can be computed from its permissions, thus its roles.

The graphical representation for the examples (*configuration graphs*) is as follows (in examples we will have constants for subjects, objects, and roles, with capital initials).

- As usual in related work, the function *PA* for an RBAC configuration is represented by an access control matrix which shows the *CR* or *CW* permissions for each role on each object, abridged to simple *R* and *W*. We call this *role-permission table*.
- Subjects are represented as ovals with the names of the subjects, their assigned roles and their computed labels.
- Objects are represented as rectangles with their computed labels.
- Directed arrows represent the relation *Channel* or *CF* between subjects and objects, usually in transitively reduced form and without reflexive edges. The direction of an arrow is the direction of the permitted data transfer or flow, and bidirectional arrows signify the presence of channels or flows in both directions. We do not use different arrows for the *Channel* and the *CF* relations derived by transitivity since we are interested in *CF* only, as it could be implemented by different channel combinations.
- Configuration graphs will be usually arranged in an upward direction, to show the data flow from the highest integrity to the highest secrecy level.
- For RBAC configurations, we may also give their *partial order graphs*. Such graphs consist of double-sided rectangles containing each a label and the set of equivalent entities that have that label; the arrows represent the inclusion partial order in the set of labels, and so also the data flow between equivalence classes.

Informally, we say that subjects *can know* data and objects *can store* them, where data are files or databases. Subjects can know data from their users (RBAC function *SU*) or read them from objects; they can also write data that they know on objects. Objects can store data of their own and data that they get by the effect of subjects writing on them. We are interested here in potential data flows, and so only in the possibility of reading or writing, rather than in actually occurring reading or writing operations.

We will use the following intuitions:

- *CF(s,o)* implies that any data that *s* can know, *o* can store;
- *CF(o,s)* implies that any data that *o* can store, *s* can know;
- *CF(s,s')* implies that any data that *s* can know, *s'* can know also;
- *CF(o,o')* implies that any data that *o* can store, *o'* can store also.

A first example follows to introduce the notation and some basic ideas of our method.

Fig. 1.a) shows a role-permission table. Fig. 1.b) shows, as a bipartite graph, a network with three subjects, *S1* with roles *R1* and *R2*, *S2* with role *R3*, and *S3* with role *R1.* The read

and write permissions of the subjects on the objects are represented by arrows (this representation will not be used in the rest of the paper, since it is easily obtained).

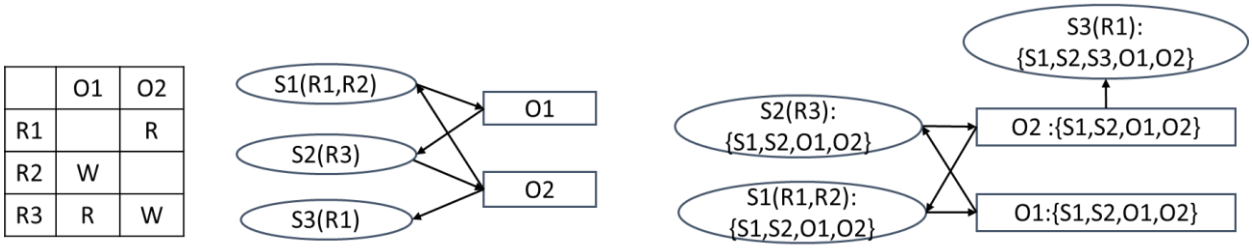

Figure 1. a) A role-permission table, b) its corresponding bipartite graph, and c) its corresponding labeled graph

Fig. 1.c) shows a representation of the same configuration but oriented from maximum integrity towards maximum secrecy, with the labels calculated according to Defs. 5 and 6. There are two equivalence classes of entities in the configuration, i.e. $[S1,S2,O1,O2] \sqsubseteq [S3]$. Also $Lab([S1,S2,O1,O2]) = \{S1,S2,O1,O2\}$ and $Lab([S3]) = \{S1,S2,S3,O1,O2\}$. Fig. 2.a) shows at the same time the entities, their labels derived from their roles, and their bipartite partial order. We call this representation a *partial order graph*. We can now see that the configuration described so far is equivalent to others, one of which is shown in Fig. 2.b) for the role-permission table of Fig. 2.c), where only two roles are used. The inheritance $R1>R2$ can be defined.

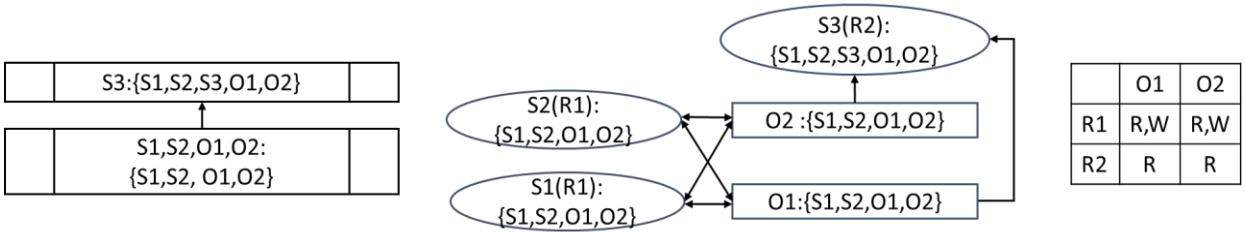

Figure 2. a) The partial order graph for the network of Fig.1; b) an equivalent network; c) its role-permission table

So a partial order graph is a representation of a number of equivalent RBAC networks and configurations and can be a basis for a reorganization of the set of entities and roles. This is important in view of possible implementation constraints, by which some role or permission assignments could be preferable to others.

Fig. 1.a) shows a *reduced* data flow network, in the sense that the removal of any permissions would change the labels and the data flow. In reduced networks, data may have to go through intermediate entities in order to get to entities that can know or store them. Fig. 2.b) shows instead the *maximal* data flow network in its equivalence class, since all possible channels are present, as in Construction 1. In maximal networks, such as the one of Fig. 2.b), direct permissions are given to subjects to transfer data to wherever they can end up and from wherever they can come. In implementations, which permissions are given depends on the organizational structure and needs. We will usually show reduced graphs, in order to reduce arrow clutter for readability. Note that, while a maximal data flow network is unique in its equivalence class, in general reduced networks are not.

Examples like this, where we have a totally ordered label set, can be considered to be implementations of simple Bell-La Padula systems. Subject *S3* could be considered to be at the *Secret* level since its data cannot be known anywhere else, where all the remaining entities could be considered to be at a lower level, which may be called *Public*, since their data are known everywhere. For integrity the converse is true: the entities in the lower-level equivalence class, not getting data from other entities, have collectively the highest integrity, while *S3*, that gets data from the former class, has the lowest integrity.

From this analysis is also possible to get representations showing the data flows among subjects and objects (thus eventually among users), such as those in Fig. 3. We shall not elaborate on these representations in this paper, but they could be useful for administrators.

Figure 3. a) Data flow among subjects; b) data flow among objects for the example of Fig.1

4.2 Other examples of role assignment analysis

Given sets of subjects and roles, we can analyze the data flows arising by assigning different combinations of the roles to the subjects. Each combination is a different RBAC configuration. Questions that can be answered by such analysis for a given configuration include:

- How can data flow between subjects (and so their users) and objects?
- What are the most secret entities and those that have the highest integrity?
- Are specified data separation constraints implemented?
- Are there parts of the network that don't appear to be useful for subject-to-subject or for user-to-user data flows?
- Are there configurations that are equivalent to the current one?

Of course, in order to do these analyses on real systems, appropriate software must be developed, and certain difficulties have to be taken care of, among others in practical RBAC systems permissions may be conditional, which is one important aspect we don't consider.

To establish a relation with related research, we use an example from a previous paper by Radhika et al. on a closely related topic [36], which in turn was inspired by examples in a paper by Chakraborty et al. on RBAC to ABAC policy mining [10]. This example is extremely small, but the advantage is that all the facts claimed will be easy to see. Three objects and four roles are proposed, see Table 1. When subjects are assigned to roles, we get the *SP* function, which will be used for the analysis.

Table 1. Sample role-permission table

	O1	O2	O3
R1	R		W
R2		W	
R3			R
R4	R		R

We will present a few configurations, to illustrate the analysis that can be carried out on different subject and role combinations, and also the very different results that can be obtained. But immediately we can see that

1) Object *O1*, being able to be read only, will always be of highest integrity in this configuration; it can be interpreted as a database containing constant values at the current configuration.
2) Object *O2*, being able to be written only, will always be of highest secrecy.
3) Since roles *R3* and *R4* have only reading permissions, subjects having only these roles will have the highest secrecy; on the other hand, since role *R2* has only writing permissions, subjects having only this role will have the highest integrity.

A question of principle can be expressed about this role-permission table, since all possible assignments of roles to subjects will lead to flows starting from *O1*, which then must have constant contents, and ending in *O2* (not necessarily in the same flow), which cannot be read by any subject, thus by any user. Questions of this type can be answered by saying that the system might allow other configurations, with different flows. For example, *O1* at this configuration might contain data written there in a previous configuration, and *O2* might be the destination of data to be read in future configurations. Such considerations will be of interest for a systems designer or administrator.

Let us see some examples of assignments. The first assignment considers a configuration with four subjects, each having one of the four roles. The labeled global data flow is given in Fig. 4.a), and its partial order graph is given in Fig. 4.b).

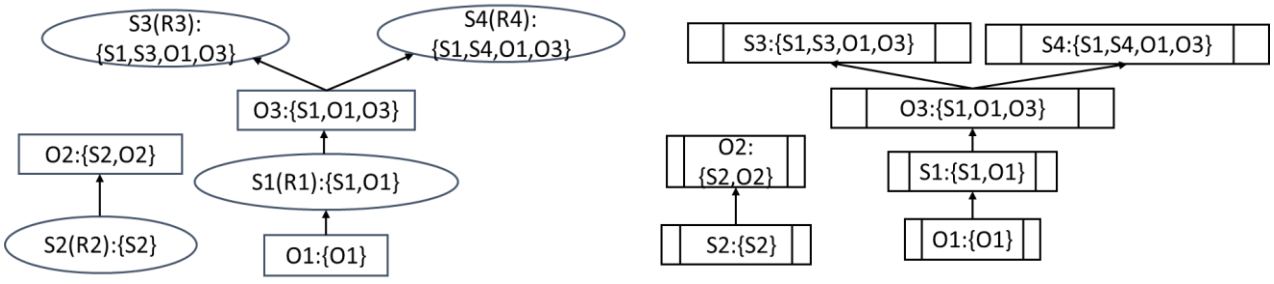

Figure 4. a) A network with one role per subject; b) Its partial order

As mentioned, the permission of *S4* to read from *O1* is not shown since *S4* can get *O1*'s data indirectly by transitivity. All equivalence classes are singletons. The most secret entities are *O2, S3* and *S4*, and those that have the highest integrity are *S2* and *O1*. We see that *S2* can write data on *O2* (imagine a database of statistics which is being compiled for use in later

configurations). The right-hand flow could be interpreted as *S1* processing data coming from a user (not shown), using constant data in *O1* and writing results in *O3*, results that can be consumed by users associated with *S3* and *S4*. This role assignment implements a situation of conflict or separation of data (possibly a *Chinese Wall* [9,38]) between the data of *O2* and those of *O1,O3*, since no entity can have labels containing *{O1,O2}* or *{O2,O3}*. This causes the flow to be partitioned in two.

Note also that Fig. 4.b) is not a lattice and could be made into a lattice only by adding extraneous entities and permissions, inconsistent with what appear to be the requirements that motivated the design of this configuration. The same will be true for Fig. 7.b) and Fig. 8.

Next, we consider the opposite case, where all roles are assigned to a single subject *S1*, see Fig. 3. *S1* and *O3* are equivalent, however here they are in a configuration where they are obliged to read and write only on objects that are not accessible to any other subject: *O1* which has maximum integrity and minimum secrecy, and *O2* which is the opposite. *S1* can get data from a user, and calculate results using constants in *O1*. Some of these results can be stored in *O3*, and then remain accessible to *S1*, and others in *O2*, for possible future access.

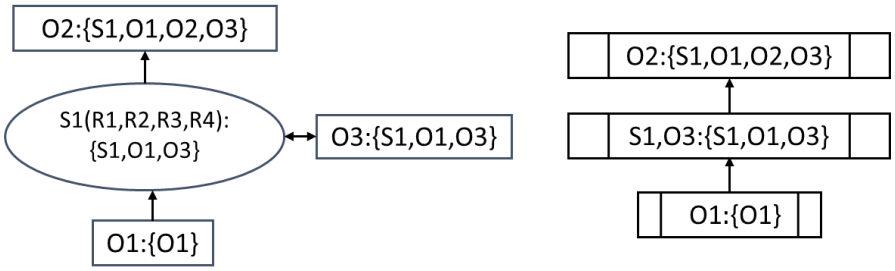

Figure 5. a) A network with only one subject that has all the roles; b) its partial order

Next, we consider a configuration with two subjects *S1* and *S2*, where *S1* has roles *R2* and *R4*, and *S2* has roles *R1* and *R3*. This network is represented in Fig. 6.a), while Fig. 6.b) is its partial order graph.

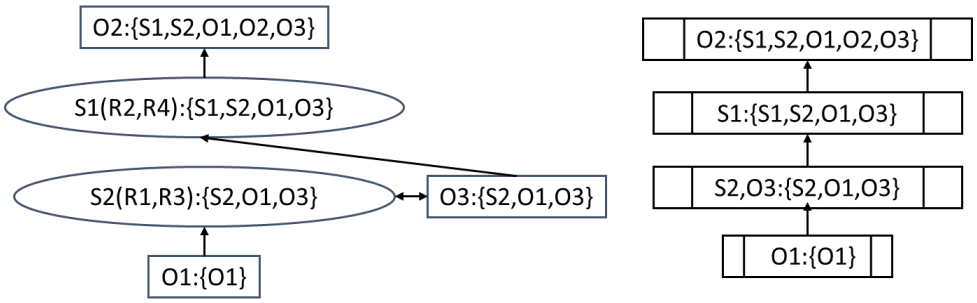

Figure 6. a) Another assignment of roles and b) its partial order

We have the non-singleton equivalence class *{S2,O3}*. Again *O1* has the highest integrity and least secrecy, while *O2* is the opposite. In this case, *S2* could be processing data brought in from its user, using data in *O1*, and storing the results in *O3*. These data are available to *S1* which can make them available to its user while storing them in *O2* for later use.

Finally, we consider a case where there are two subjects, *S1* with roles *R2, R4* and *S2* with role *R3*. Role *R1* is not used.

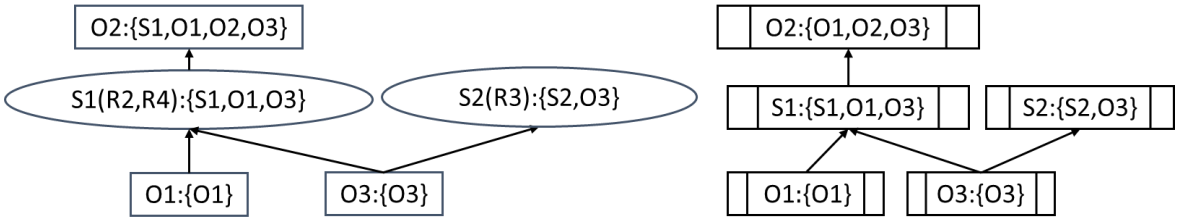

Figure 7. a) Another assignment of roles and b) its partial order

In this case *O1* and *O3* are of lowest secrecy but highest integrity, both are constant databases. *S2* cannot know *O1*'s data and the subjects cannot know each other's data, describing a situation of conflict. *CF(O1,O3)* is false because *R1* is not used. This example could be interpreted as two users acting through their subjects to share database *O3*, *S1* using its own database *O1* and storing its results in database *O2* for future use.

Although we are leaving users implicit in this paper, for interpreting the previous examples we have brought in the notion of user. In fact, different associations of users to subjects (RBAC function *SU*) might be required for different utilizations of the networks. This type of study is interesting in view of specific applications. In Sect. 4.3 we will see an example where we will design a network according to data flow requirements among users.

Labels can be extended to users. In each configuration, for a user *u, Lab(u)= ∪{Lab(s) | SU(s)=u},* this defines a partial order of users based on the labels of their subjects. If it is desired to construct labels that explicitly represent data flow among users, then such labels can be constructed for users including user names, by replacing each *Name(s)* with *Name(u)* where *SU(s)=u*. We leave this to further research.

This analysis can lead to imposing 'separation of duties' (SoD) constraints. In the example of Fig. 5, if *SU(S1)=U*, the network shown is impossible if *U* has the (static or dynamic) constraint *({R1,R2,R3,R4},4)*. Separation of duties can also be specified, in a different manner and with different results, by excluding certain combinations of categories in labels. For example, several of the networks seen above are impossible if labels containing *{O1,O2,O3}* are forbidden.

4.3 Example of deriving roles from data flow requirements

We show here an example of application of Theorem 3, by which, given any network, it is possible to construct an equivalent RBAC configuration.

The bipartite network of Fig. 8 was generated at random, but it could be justified in terms of the following description. In a company, there are normally many teams and many data flows, and the network of Fig. 8 can be thought as showing one of them, which we call the Project. Six users and three databases are involved in the Project. We assume that each user can open only one session, and so we give the subjects the names of their users. Zak is the manager of the Project and Ali is its accountant. Moh, Kai and Jul work in collaboration, forming an equivalence class of subjects that will be called the MainTeam. Ben has been asked to work on his own, without any communication except reporting to the manager. The four databases are used as shown in the diagram, they are used for communication, and they can also contain their own data for storage and consultation. The members of the MainTeam

have different permissions on the databases DB A and DB B but each of them can get all their contents, directly or indirectly. Fig. 8 could be expressed as an access control matrix.

In terms of security requirements, this network satisfies the following:

- The MainTeam can share among themselves all data, including those of DB A and DB B.
- The MainTeam, including its databases, cannot know the data known or stored in outside databases or by outside Project members. It constitutes an equivalence class of maximum integrity.
- Zak and Ali cannot know each other's data. Each has maximum secrecy.
- Ben can only know its own data, and so is of maximum integrity.
- Only Zak can know the contents of all databases.
- Ali can know the data of the MainTeam and DB C, it cannot know the data of Ben, DB D or Zak.
- DB D can only store the data of the MainTeam and Ben, including its own.
- DB C can only store the data of the Main Team, including its own.

We have expressed the requirements visible in Fig. 8, or Fig. 8 could be derived from the requirements, possibly specified in a more stylized form, and we leave this to further research.

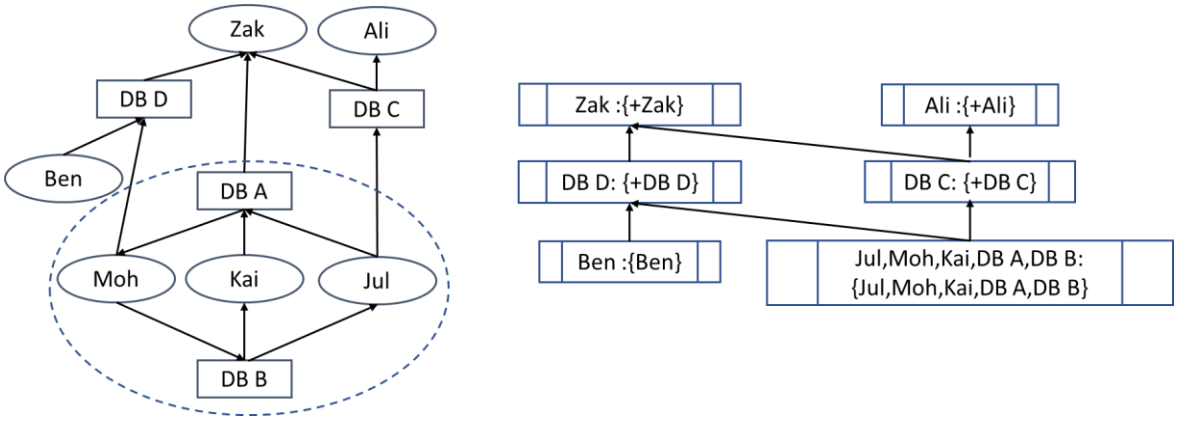

Figure 8. a) Data flow in the Project; b) Labels in the Project

To give labels to the entities of this network, according to Defs. 5 and 6 and Theorem 4, we start by giving all members of the MainTeam their common label, which is *{Moh, Kai, Julie, DBA, DB B}.* To shorten the other labels, we use the familiar notation +, meaning that each equivalence class of entities has a label that includes its own and the ones of the entities from which it can read. Fig. 8.b) shows the bipartite partial order of the labels of the entities in the Project Table 2 shows the *CR* and *CW* permissions of each subject on each object. According to Theorem 3 and Construction 1, roles for the project can be generated as shown in Table 3. The obtained RBAC network is equivalent to the initially given network.

Note that the permissions of Tables 2 and 3 give subjects direct access to databases to which they only have indirect access in Fig. 8.a). This is consistent with our view that, from

a security perspective, direct or indirect access are equivalent. Surely, the permissions can be modified according to other considerations.

Table 2. Permissions for the Project

	DB A		DB B		DB C		DB D	
	CR	CW	CR	CW	CR	CW	CR	CW
Zak	V		V		V		V	
Ali	V		V		V			
Ben								V
Moh	V	V	V	V		V		V
Kai	V	V	V	V		V		V
Jul	V	V	V	V		V		V

Roles (SUBJECTS) \ OBS	DB A	DB B	DB C	DB D
R1 (Zak)	R	R	R	R
R2 (Ali)	R	R	R	
R3 (Ben)				W
R4 (Moh,Kai,Jul)	R,W	R,W	W	W

Two inheritance relations can be defined, namely: *R2<R1* and *R3<R4*. However we reiterate that these roles and inheritances are mechanically determined according to the permissions required and may not be appropriate in terms of the structure of the organization and its evolution needs. In practice, these require further analysis.

Following is a summary of the method:

1) The data flow requirements are specified in a language (yet to be defined).
2) The specification is translated into a bipartite graph (method to be devised).
3) The methods described in Section 2 are used to identify the bipartite partial order of equivalence classes and calculate the labels for the entities in the bipartite graph.
4) The reading and writing permissions are calculated from the labels and the roles are assigned to subjects (Construction 1).

Note that, in order for this example to be realistic, we must assume that there are other data flows in the Project: one such flow would be necessary to allow Zak to relay instructions and feedback to his team, another would allow Zak and Abby to communicate, etc.

There are two different mechanisms to allow such additional data flows:

- Sequentially, by users changing roles for subjects or creating different sessions
- Concurrently, by allowing users to have different Usernames, each with different roles defined on different data types. In the example above, one could define a datatype for InstructionToTeam, another for RequestToAccountant, etc. In practice, each such datatype will have its own data format to be sent over specific channels. The different Usernames for a user will have to keep their data separate. Communication among them will have to abide by specific protocols, possibly including anonymization and other practices that are not the subject of this paper.

The second mechanism is not usually considered in RBAC literature, but there is nothing against using it in practice.

4.4 Example of deriving roles from security labels

In organizations, data flows can be defined implicitly by directly defining security labels and assigning them to subjects. As a variation of the above example, we now give an example of this.

The typical examples of security labels found in practice show totally ordered sets of labels, such as *Public ⊑ Protected ⊑ Confidential ⊑ Secret*. Our theory [28,29] allows security

labels to form more complex partial orders. Fig. 9 shows a partial order of labels that can also be written in the following way:

$ProtectedA \sqsubseteq ConfidentialA \sqsubseteq SecretA,$

$ProtectedB \sqsubseteq ConfidentialB \sqsubseteq SecretB,$

$ProtectedB \sqsubseteq ConfidentialA,$

$ConfidentialB \sqsubseteq SecretA.$

This is a renaming of the partial order of Fig. 8.b) but could also be interpreted as a labeling system used in an organization with two offices, *A* and *B*, with a requirement that some employees (or databases) of office *A* should be able to know (or store) some of office *B*'s data, as shown. Since this is a bipartite partial order, according to Corollary 1, it can be translated into an RBAC network. The RBAC role-permission table for this example is still the one of Table 3.

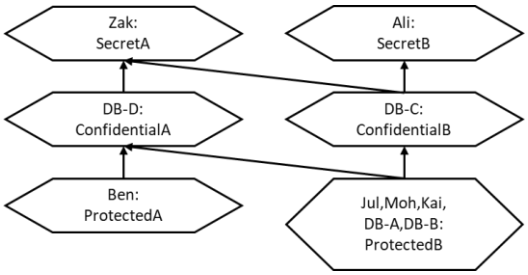

Figure 9. Example of security labels

5. Reconfigurations

When any of the sets or functions in an RBAC configuration change, we have a *reconfiguration*. Reconfigurations can occur by user action, such as activation or deactivation of sessions or roles, by administrative action or by effect of policies. In some environments (e.g. in the Cloud or in the IoT), reconfigurations can also occur as effect of failures, e.g. when a subject, object or communication line fails. We will take failure as a case of subject, object or permission removal. Reconfigurations of RBAC networks yield other RBAC networks with their own relations, partial orders of equivalence classes and sets of labels.

We are interested in reconfigurations that change the *CR, CW* relations, hence possibly the *CF* relation. Logrippo [28] gives a brief discussion of 'transformations' in generic networks. In RBAC we have additions or removals of permissions to subjects, which can be caused by addition or removal of permissions to roles, addition or removal of roles to subjects, or changes in inheritance relations. Creation and removal of entities will be treated by assuming the existence of a sufficient number of subjects with no roles, and objects for which no permissions exist. These will have labels containing only their own names. They will be connected to the rest of the network when roles or permissions are created for them or towards them and disconnected when there are no more such roles or permissions (they may then be purged of all the data they have come to know or to store, see later).

Describing in general terms the effects of reconfigurations on the *CF* relation, and thus on labels, secrecy and integrity, is difficult for several reasons: the transitive nature of the *CF* relation; the fact that role assignment or inheritance (or their removal) can give (remove)

several reading and writing permissions at once; as can the fact that a change of permissions for a role has effect on all subjects that have the same role, or have senior roles if inheritance is present. Of course, reconfigurations can lead from a configuration to an equivalent one (i.e. may not change the *CF* relation) but in general this can only be known after the relation is recalculated for the whole network.

We assume that entities maintain their names in different configurations and we use subscripts to denote their relations at various configurations. So we write $CR_i(x,y)$ to refer to the *CR* relation existing at configuration *i*, and similarly for CW_i, CF_i and Lab_i. In the rest of this section, we will use indexes *i* and *k*, where *k* is a configuration that follows *i*. Calculating the new labels is easy in two cases of adding permissions, by application of Defs. 5 and 6:

Property 2. If the only difference between configuration *i* and configuration *k* is that for some *s* and *o*:

1) $CR_i(s,o)$ is *false*, but $CR_k(s,o)$ is *true*, then $Lab_k(x) = Lab_i(x) \cup Lab_i(o)$ for all *x* such that $CF_i(s,x)$.
2) $CW_i(s,o)$ is *false*, but $CW_k(s,o)$ is *true*, then $Lab_k(x) = Lab_i(x) \cup Lab_i(s)$ for all *x* such that $CF_i(o,x)$.

If, however, read or write permissions are removed, then the solution is less simple, since it is not possible to remove the name of an entity *x* from the label of an entity *y* without considering all possible flows from *x* to *y*. To see this, consider the following example: two subjects *S1* and *S2* and two objects *O1* and *O2*, with *CR(S1,O1), CR(S2,O1), CW(S1,O2), CW(S2,O2).* Then $Name(O1) \in Lab(O2)$ and this will remain true if *CW(S2,O2)* is removed.

A difficulty is the fact that roles may include both reading and writing permissions. To simplify this aspect, we follow Sandhu's proposal [41] to separate reading and writing roles. Using Tab. 1 as an example, reading and writing roles can be separated by splitting Role *R1*, as seen in Tab. 4. Hence in a network using Tab. 4, assignment (removal) of role *R1* of Tab.1 can be accomplished by assigning (removing) roles $R1_R$ and $R1_W$. In the rest of this section, we assume that each role is either a writing or a reading role. Although such separation is not done in current practice, it can be used in order to better understand the effects of adding or removing roles.

Recall that a subject that has only reading roles is a subject of highest secrecy, while a subject that has only writing roles is a subject of highest integrity. In the middle are subjects with both reading and writing roles.

Table 4. Separation of reading and writing roles

	O1	O2	O3
$R1_R$	R		
$R1_W$			W
$R2_W$		W	
$R3_R$			R
$R4_R$	R		R

Adding or removing reading or writing roles may have effects on the secrecy and integrity of entities.

Definition 14:

1) The *secrecy of x is increased* (resp. *decreased*) with respect to *y* from configuration *i* to configuration *k* if $CF_i(x,y)$ but *not* $CF_k(x,y)$ (resp. $CF_k(x,y)$ but *not* $CF_i(x,y)$).
2) The *integrity of x is increased* (resp. *decreased*) with respect to *y* from configuration *i* to configuration *k* if $CF_i(y,x)$ but *not* $CF_k(y,x)$ (resp. $CF_k(y,x)$ but *not* $CF_i(y,x)$).

Or equivalently, since $CF(x,y)$ iff $Lab(x) \subseteq Lab(y)$:

1) The *secrecy of x is increased* (resp. *decreased*) with respect to *y* from configuration *i* to configuration *k* if $Lab_i(x) \subseteq Lab_i(y)$ is *true* but $Lab_k(x) \subseteq Lab_k(y)$ is *false* (resp. $Lab_k(x) \subseteq Lab_k(y)$ is *true* but $Lab_i(x) \subseteq Lab_i(y)$ is *false*).
2) The *integrity of x is increased* (resp. *decreased*) with respect to *y* from configuration *i* to configuration *k* if $Lab_i(y) \subseteq Lab_i(x)$ is *true* but $Lab_k(y) \subseteq Lab_k(x)$ is *false* (resp. if $Lab_k(y) \subseteq Lab_k(x)$ is *true* but $Lab_i(y) \subseteq Lab_i(x)$ is *false)*.

Note that the secrecy of *x* is increased with respect to *y* iff the integrity of *y* is increased with respect to *x*. Also the secrecy of *x* is decreased with respect to *y* iff the integrity of *y* is decreased concerning *x*. Increase or loss can be with respect to several entities.

Theorem 5. For a subject *s :*

1) Adding a reading role to *s* does not decrease its secrecy or increase its integrity with respect to any entity.
2) Removing a reading role from *s* does not increase its secrecy or decrease its integrity with respect to any entity.
3) Adding a writing role to *s* does not increase its secrecy or decrease its integrity with respect to any entity.
4) Removing a writing role from *s* does not decrease its secrecy or increase its integrity with respect to any entity.

Proof. There are eight properties here, but the proofs are similar.

1) Adding a reading role to *s* will not reduce its label (Prop.2.1). For secrecy, this means that it will not be the case that $Lab_i(s) \subseteq Lab_i(y)$ is *false* but $Lab_k(s) \subseteq Lab_k(y)$ is *true* for any *y*. In other words, *s* will not flow to any additional entities. Hence the secrecy of *s* will not decrease with respect to any entity. For integrity, all the entities that could flow to *s* earlier still could after the addition, so the integrity of *s* will not increase.

2) Removing a reading role from *s* may result in its label to be reduced. *s* will still be able to flow to the entities to which it flew before (secrecy not increased) but no additional entities will become able to flow to *s* (integrity not decreased).

3) Adding a writing role to *s* will not change the label of *s* but may augment other labels. *s* will still be able to flow to all entities to which it flew earlier (secrecy of *s* not increased); no other labels may become included in the label of *s* (integrity of *s* not decreased).

4) Removing a writing role from *s* will not change the label of *s* but may reduce other labels. The secrecy of *s* will not decrease but its integrity will not increase.

Note that, of course, whenever it is said that a property is not increased (or not decreased), this means that it may well decrease (or increase).

Very similar properties can be derived to describe what can happen to the labels and the secrecy and integrity of objects if subjects get or lose roles that allow them to read from or write to them.

The properties of Theorem 5 are weak because, when a role is added, it is not known if its permissions were already present because of another previously assigned role (similarly for removals). Stronger properties could be proved by adding assumptions, but this would complicate the statements without significant added insight. More useful is the fact that stronger properties can be derived if the labels are recomputed after each reconfiguration and compared with the ones of the previous configuration, see Property 2 above for two easy cases. The application of Def. 14 will then make it possible to determine the changes in secrecy and integrity that have occurred. We leave this matter for future research.

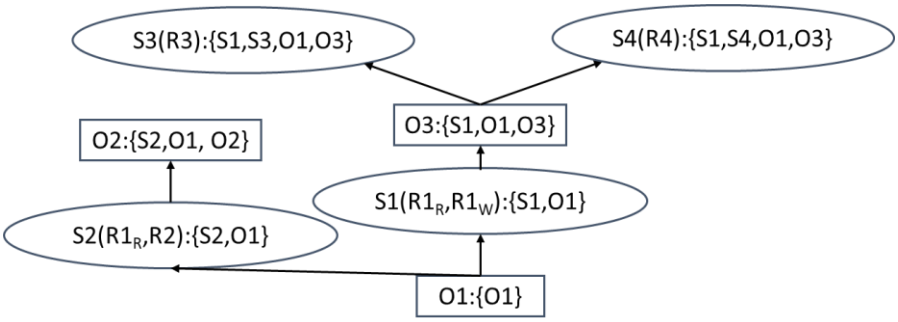

Figure 10. A reconfiguration of Fig. 4

As a first example, consider the network of Fig. 10, which is a reconfiguration of the network of Fig. 4 where role *R1* has been split into $R1_R$ and $R1_W$ (Table 4) as well $R1_R$ has been added to *S2*. So we are in case 1) of Theorem 5, also in Case 1) of Property 2. *O1* has been added to the labels of *S2* and *O2*. The secrecy of *S2* or *O2* has not been decreased with respect to any entity. However the integrity of *S2* and *O2* has been decreased with respect to *O1* and so the label of *O1* is now included in the label of *S2* and *O2*. Correspondingly, the secrecy of *O1* has been decreased with respect to *S2* and *O2*. Perhaps a warning to the administrator or to the users involved should be issued.

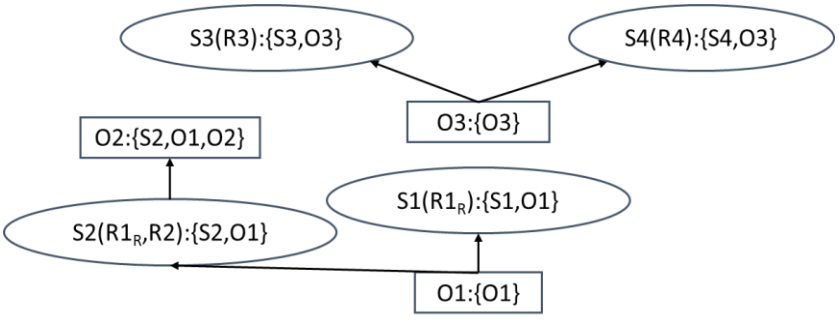

Figure 11. A reconfiguration of Fig. 10

Further, consider the network of Fig. 11, where the network of Fig. 10 has been reconfigured by removing $R1_W$ from *S1*. We are in case 4 of Theorem 5. We see that the secrecy of *S1* and *O1* has increased with respect to *O3, S3, S4,* so *S1* is now of maximum secrecy. We also

see that the integrity of *O3, S3* and *S4* has increased with respect to *S1* and *O1*, however it might not have been so if there were other flows between *O1, S1* and *O3*.

Increases or decreases in secrecy or integrity are important in practice because of possible data flows through reconfigurations [28] These flows occur if an entity can keep the data that it has known or stored through reconfigurations, in other words if we assume that these data will be available for reading or writing in the next configuration. In our formalism, there is the implicit assumption that entities keep the data for which the flow from the originating entity still exist, but purge the data for which the flow from the originating entity has been lost. In practice, policies should be established for the following events:

- The label of an entity *x* loses *Name(y)* for some *y*. The integrity of *x* is increased with respect to *y* and the secrecy of *y* is increased with respect to *x*. *x* should be required to purge all data previously acquired from *y*. Administrative warnings may be in order. Also this may be prevented from happening by policies, e.g. policies specifying that *x* should always know the data of *y* (constraints on the sets of labels were discussed in [28]).
- The label of an entity *x* acquires *Name(y).*The secrecy of *y* is decreased with respect to *x* and the integrity of *x* is reduced with respect to *y*. Administrative warnings may be in order. Also this may be prevented from happening by policies, e.g. policies specifying that the data of *y* should never be known to *x*.

More complicated labeling methods could also be envisaged, for example to remember that an entity is keeping data acquired until a specified reconfiguration, but we leave this for further research.

Purging and warning policies are well-known in corporate practice and have their place in the IoT. For example, when an employee moves from one office to another, which is not higher in the data flow order, normally she not only loses access to the data sources she had access to in the previous office, but she also has to dispose of (purge) all related documents she might have acquired from those sources. This matter acquires a different aspect if the data themselves can be transformed. For example, in a hospital setting, data can be moved from one entity to another of lower classification after losing identifying information, they then become different data with different secrecy requirements, see Myers and Liskov [31]. A new data flow is started with the anonymized data and the entity where the anonymization is done can be thought as split into two entities, one that is at the end of a flow, and another that is at the beginning of a new flow [29]. *Obsolescence* is an automatic data transformation by which data lose secrecy value in time. This has been studied in other contexts and we leave it for future research in our context.

Theorem 5 is also relevant for the introduction or reconfiguration of role hierarchies. It is interesting to see that the secrecy of a subject *s* may decrease if *s* becomes senior of a role that has writing permissions that its initial role did not have. Data purging from *s* may be required.

6. Literature review and comparisons

The theory developed here was influenced by two early papers, the work of Bell and La Padula [4] and its refinement in the work of Sandhu [39]. Although neither of these papers is on RBAC, they introduce several basic concepts that are still valid, well beyond the application

areas for which they were originally conceived. These concepts are: the partial ordering of data security levels and its relation with data flows, labels, secrecy and integrity; with the idea that data flow policies can be enforced by constraining the sets of available labels. Bell and La Padula [4] use a partial order model (see below) while Sandhu [39] uses a lattice model and cites the fact that partial orders can be embedded into lattices [5]. It has been shown in our papers [28,29], as well as in this paper, that this embedding may force the introduction of unnecessary entities, and that a simpler, more general theory can be developed by using directly partial orders instead of lattices.

Basing data flow theory on partial order theory generalizes established notions of secrecy and integrity. There can be little doubt that the top entities in a partial order, not having outgoing channels, should be considered the most secret. Coming to integrity, several notions of integrity have been presented in the literature, and ours is consistent with Sandhu's *information flow integrity* [40], and derives from Biba [7]. It also conforms with Bell-La Padula's definition [4], where it is noted that an entity with no incoming flows "cannot be sabotaged". Sandhu shows in [39] that both secrecy and integrity levels can be expressed in a single lattice partial order and we follow his idea, after extension to general partial orders. Since our labels determine both secrecy and integrity of entities, together with the permitted data flows, it is justified to call them *security labels*.

The short paper by Osborn [34] was the first on the problem of data flow analysis for RBAC. We share its basic goals, and we agree on its examples, but we generalize its approach in several ways: i) the paper addresses data flows among objects, as determined by data flows among roles, and not data flows among subjects and objects as we do; ii) the paper assumes that to generate security labels it is necessary to generate a lattice flow, which we show not to be the case; iii) as a consequence, we can in our paper introduce concepts of secrecy and integrity, which that paper does not use; iv) the paper uses its own algorithms, based on node and edge creation; these algorithms are not analyzed for complexity.

Gofman et al. [17] elaborate on the results of the previous paper and present improved algorithms, which are specific to this problem and not the application of general-purpose algorithms as in our case. They provide a complexity analysis of the algorithms they propose, which is polynomial, just as ours [42], however with some exponents to the fifth power while our algorithms are of cubic complexity, due to the use of the transitive closure algorithm together with the linear algorithm that constructs the partial ordering. A comparison of their algorithm with ours would require much detailed work, since our approach differs from theirs for the same reasons i) to iii) mentioned for Osborn's paper. It would require aligning the two algorithms to take in consideration these differences, and then executing them on the same data and computer. The paper continues with an "incremental analysis algorithm" which proposes to "incrementally update the information flow graph in response to changes to the RBAC policy". We have presented our view on this subject in Sect. 5. Essentially, instead of ad-hoc algorithms to take care of various types of reconfigurations, which these authors propose, we propose to recalculate all the labels at each reconfiguration, which is an efficient process. In any case, comparing algorithm efficiency is outside of the scope of this paper.

Then, this research subject has remained dormant until the paper by Radhika et al. [36], which is the most recent reference. This paper essentially "describes how a lattice model can be captured using an RBAC configuration" and it also "helps in creating information-flow

secure RBAC policies". Our aims are similar but not being bound to the lattice model we can be more general. While they claim that standard RBAC does not provide information flow control, we claim that any RBAC configuration with reading and writing permissions controls the flow of data in ways that can be determined by using efficient algorithms. We use generic algorithms, while they use their own algorithms. They derive labels for objects and roles, while we find it more useful to derive labels for subjects and objects. Their method does not produce the detailed flow analyses, involving combinations of subjects and roles, that we have demonstrated in Sect. 4.

None of these three papers positions itself in the framework of a general theory of data security in networks. In particular, none of them justifies the use of the lattice model, except than by citation of previous work. However, as mentioned, Osborn notes in [34] that the Can-Flow graph for RBAC configurations "may or may not be a lattice".

Coming to other related papers, Kuhn [22] presents a method for implementing RBAC using multi-level mechanisms. He develops a set of definitions for assigning permissions to category sets, and he evaluates how many roles can be supported with different numbers of categories, ignoring security levels. His concern is implementation, whereas our concern has been to show that several methods of defining data networks are essentially equivalent and can be mutually translated in principle.

Barkley [3] shows how an RBAC access control *policy* can be created for any given access policy expressed in terms of an access control list. See for this our Corollary 1.

Osborn et al. [33] show that it is possible to configure RBAC to enforce both Lattice-based and Discretionary access control policies (LBAC and DAC). They present constructions to satisfy in RBAC the various properties of LBAC, such as Simple security property, Liberal *-property, Strict *-property, and variations (a previous paper by Sandhu [41] is based on similar ideas). Concerning DAC, they show that it is possible to represent in RBAC not only access control lists, but also reconfigurations due to creation and destruction of objects, ownership changes, granting and revocation of permissions. Their goals are more general than ours.

Zhao and Chadwick [48] present an alternative view of the same topic. They concentrate on the Bell-La Padula model [4], and they show how it can be implemented in RBAC. Faithful to BLP, they consider executing, reading, appending, and writing permissions, as well as executing roles, reading roles, appending roles and writing roles. They define four mappings, one from subjects to executing roles, another from security levels to reading roles, another from security levels to appending roles, and a final one from security levels to writing roles. They also take advantage of role hierarchy allowing reading permissions to increase and writing permissions to decrease according to the positions in the security hierarchy.

Habib et al. [18] present the BLP model as a lattice-based model and get to the conclusion that "the BLP model is more restrictive than the RBAC model".

All these authors define BLP, as well as LBAC and MLS, as lattice-based, while in fact the BLP report [4] bases its model on the theory of partial orders without any mention of lattices (most clearly, see Fig. A1, p. 72 in [4], which is not a lattice). Thus BLP, just as the bipartite MLS defined in this paper, can specify any bipartite network and so any RBAC network (Section 3.2).

The contribution of our paper is theoretical, i.e. to show, by a straightforward proof, that all the methods mentioned for defining data flows are, in their essence, mutually equivalent

and mutually reducible. Our proof uses a construction that has no claims of practicality or efficiency, and so if realistic constructions are required, other methods should be used. We do not propose here a direct comparison with the BLP model since it is complex and would require a separate study.

Koch et al. [21] present a graph-based formalism for RBAC. However their graphs represent structural relationships such as ownership of sessions, role assignments, activation of roles, etc., which are different concerns than data flow and security.

The problem of analyzing the effects of reconfigurations (the *safety analysis problem* for RBAC) has been studied in several papers. See Jha et al. [20] which contains a literature review. Examples of properties studied in this literature are whether certain classes of users could possibly gain or be refused access to resources by effect of sequences of reconfigurations. A related topic is comparing the expressive power of access control models for their ability to preserve security properties across state transitions [45]. These issues are outside of the scope of this paper, which is concerned only with the presentation of our partial-order model and the immediate consequences of basic reconfigurations.

Our approach does not take into account role hierarchies. However for any RBAC network defined with the use of role hierarchies, there is an equivalent one defined without using them, and role hierarchies can always be 'flattened', see Chen and Crampton [12]. Crampton [13] pointed out the awkwardness of implementing multi-level systems using RBAC because of the inheritance rules in RBAC, and proposed a new inheritance model to address this problem. An earlier paper by Sandhu [41] addressed similar concerns.

7. Conclusions

We have reviewed the long-standing problem of data flow analysis for RBAC systems, using the concept of partial order of equivalence classes of entities, so far not exploited in this research area. The contributions of this paper with respect to the literature on the same topic are:

1) Previous studies were limited by the view that secure data flows must be lattice-structured, however RBAC can define data flows that are not lattice-structured. We have shown that RBAC can define any bipartite network, and that data security concepts such as secrecy and integrity can be defined for any network, thus for any RBAC network (Sect. 3).
2) Previous studies considered data flow relations between roles and objects; we have considered instead flow relations between subjects and objects, where subjects can have combinations of roles (Sect. 4).
3) We have shown the equivalence and mutual transformability between apparently different data security models such as access control matrices (or arbitrary bipartite networks), multi-level networks, and RBAC configurations. Roles correspond to equivalence classes of entities and labels (Sect. 3 with examples in Sect. 4).
4) We have shown the expressive capabilities of our partial order diagrams, where single vertices represent equivalence classes of entities. From them, information can be obtained concerning data flows, secrecy and integrity; roles and role assignments can be derived from them (Sect. 4).

5) We have shown, but only by example, how roles can be derived from data flow requirements (Sect. 4.3). In the example, we have shown how 'data separation' or 'conflict' requirements on data flows can be implemented in labels and then translated into role configurations. A generic method for generating RBAC configurations that implement specified data security requirements is a subject for further research.
6) We have shown that the reconfiguration of any network will yield another network that is a partial order of equivalence classes (while clearly the reconfiguration of a lattice is not necessarily a lattice), and so all these networks can be defined in RBAC; we have examined the data flow consequences of adding or removing roles (Sect. 6).
7) We have considered also the problem of data flows between RBAC configurations, as permissions in a system change, analyzing the possibility that subjects and objects keep their data, and pass them on in the next configuration. We have seen the effects of certain reconfigurations on secrecy and integrity. Concepts of administrative warnings and data purging were related to such reconfigurations (Sect. 6).

Reference [43] presents a method for implementing partial order data security models such as the one used in this paper in Software defined networks (SDN).

Future research could deal with the application of these concepts for the design of organizational or IoT systems with various data security requirements. Requirement specification languages could be defined, for implementation in terms of bipartite networks and roles.

This theory is of course limited to RBAC systems that constrain data flows. RBAC is a very general security model that has many applications, well beyond data security. Also, RBAC must adapt itself to the needs of the organizations where it is used, which opens other considerations that are outside of the scope of this paper.

Acknowledgment. This research was funded in part by a grant of the Natural Sciences and Engineering Research Council of Canada. We are grateful to Omer Landry Nguena Timo for his comments that have led to several improvements.

Compliance with ethical standards. This research was funded exclusively by a Discovery Grant of the Natural Sciences and Engineering Research Council of Canada (NSERC) awarded to the author for long-term research. The author declares that he has no conflicts of interest.

Ethical approval: This article does not contain any studies with human participants or animals. This research has not involved any data repositories or Artificial Intelligence methods.

References

1. A.V. Aho, J.E. Hopcroft, J.D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.
2. M. Alramadhan, K. Sha. An overview of access control mechanisms for Internet of things. Proc. 2017 26th International Conference on Computer Communication and Networks (ICCCN 2017) IEEE, 1-6.
3. J. Barkley. Comparing simple role-based access control models and access control lists. Proc. 2nd ACM workshop on Role-based access control. (1997) ACM 127-132.
4. D.E. Bell, L.J. La Padula. Secure computer systems: unified exposition and Multics interpretation. MTR-2997, Mitre Corp., Bedford, Mass., 1976.

5. K. Bertet, M. Morvan, L. Nourine. Lazy completion of a partial order to the smallest lattice. Intern. KRUSE Symposium: Knowledge Retrieval, Use and Storage for Efficiency (1997), 72–81.
6. E.Bertin, D.Hussein, C.Sengul, V. Frey. Access control in the Internet of Things: a survey of existing approaches and open research questions. Annals of telecomm. 74(7) (2019), 375-388.
7. K.J. Biba. Integrity considerations for secure computer systems. TR-3153. Mitre Corp. Bedford, Mass.(1977).
8. G. Birkhoff. *Lattice Theory,* American Mathematical Society, 1967.
9. M. Bishop. *Computer security, Art and science*. 2nd edition. Addison-Wesley, 2019.
10. S .Chakraborty, R. Sandhu, R. Krishnan. On the feasibility of attribute-based access control policy mining. 2019 IEEE 20th Internat. Conf. on Information Reuse and Integration for Data Science (IRI 2019) IEEE, 245-252.
11. H.C. Chen. Collaboration IoT-Based RBAC with Trust Evaluation Algorithm Model for Massive IoT Integrated Application. Mobile Netw Appl 24, (2019) 839–852.
12. L. Chen, J. Crampton. Risk-aware role-based access control. 7th International Workshop on Security and Trust Management, (STM 2011), Revised Selected Papers 7 2012, 140-156. Springer.
13. J. Crampton. On permissions, inheritance and role hierarchies. In: Proc. 10th ACM Conf. on Computer and communications security (CCS 2003). ACM Press, 85-92.
14. D.E. Denning. A lattice model of secure information flow. Comm. ACM 19(5) (1976), 236-243.
15. D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli. Proposed NIST standard for Role-Based Access Control. ACM Trans. on Information and System Security, 4(3) (2001) 224–274.
16. D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli. *Role-based access control.* 2nd Ed. Artech House, 2007.
17. M.I. Gofman, R. Luo, J. He, Y. Zhang, P. Yang, S. Stoller. Incremental Information Flow Analysis of Role Based Access Control. Security and Management (2009), 397-403.
18. L. Habib, M. Jaume, C. Morisset. A formal comparison of the Bell & Lapadula and Rbac models. The Fourth International Conference on Information Assurance and Security (2008) IEEE,3-8.
19. E. Harzheim. *Ordered sets.* Springer, 2005.
20. S. Jha, N. Li, M. Tripunitara, Q. Wang, W. Winsborough. Towards formal verification of role-based access control policies. IEEE Trans. on Dependable and Secure Computing. 2008 11;5(4) (2008):242-55.
21. M. Koch, L.V. Mancini, F. Parisi-Presicce. A graph-based formalism for RBAC. ACM Trans. on Information and System Security (TISSEC). 5(3) (2002) 332-65.
22. D. R. Kuhn. Role Based Access control on MLS Systems without Kernel changes. Proc. 3rd ACM Workshop on Role-Based Access Control (1998), 25-32.
23. E.Kozyri, S Chong, A.C. Myers. Expressing information flow properties. Foundations and Trends in Privacy and Security., 18;3(1), (2022) 1-102.
24. B.W. Lampson. Protection. ACM SIGOPS Operating Systems Review, 8(1) (1974) 18-24.
25. C. E. Landwehr. Privacy research directions. Comm. ACM 59(2) (2016) 29-31.

26. H. Li, S. Wang, X.Tian, W. Wei, C. Sun. A survey of extended role-based access control in cloud computing. Proc. 4th Intern. Conf`. on Computer Engineering and Networks (2015) 821-831.
27. L. Logrippo. Multi-level access control, directed graphs and partial orders in flow control for data secrecy and privacy. Proc. 10th Intern. Symp. on Foundations and Practice of Security (FPS 2017), LNCS Vol. 10723, 111-123.
28. L. Logrippo. Multi-level models for data security in networks and in the Internet of things. Journal of Information Security and Applications, Vol. 58 (2021), 102778.
29. L. Logrippo The order-theoretical foundation for data flow security. Submitted for publication. Available in: https://www.site.uottawa.ca/~luigi/papers/23_Order-Theory.pdf (accessed Jul. 2024),
30. L. Logrippo, A. Stambouli. Configuring data flows in the Internet of Things for security and privacy requirements. Proc. 11th International Symp. on Foundations and Practice of Security (FPS 2018). Springer LNCS Vol. 11358, 115-130.
31. A. C. Myers, B. Liskov. Protecting privacy using the decentralized label model. ACM Trans. on Software Eng. and Methodology, 9(4), 2000, 410-442.
32. A. Ouaddah, H. Mousannif, A. Abou Elkalam, A.A. Ouahman. Access control in the Internet of Things: Big challenges and new opportunities. Computer Networks 112. (2017) 237-62.
33. S. Osborn, R. Sandhu, Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM Trans. on Information and System Security (TISSEC). 3(2) (2000) 85-106.
34. S. Osborn, Information flow analysis of an RBAC system. Proc. 7th ACM Symp. on Access control models and technologies (SACMAT 2002), 163-68.
35. J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, B. Fang A survey on access control in the age of Internet of things. IEEE Internet of Things Journal. 7(6)(2020)4682-96.
36. B.S. Radhika, N.V. Narendra Kumar, R.K. Shyamasundar. Towards unifying RBAC with information flow control. Proc. 26th ACM Symposium on Access Control Models and Technologies (2021), ACM, 45-54.
37. S. Ravidas, A.Lekidis, F.Paci, N.Zannone. Access control in Internet-of-Things: A survey. Journal of Network and Computer Applications, 144 (2019), 79-101.
38. R.S. Sandhu. Lattice-based enforcement of Chinese Walls. Computers & Security 11(8)(1992), 753-763.
39. R.S. Sandhu. Lattice-based access control models. IEEE Computer 26(11)(1993), 9–19.
40. R.S. Sandhu. On five definitions of data integrity. Database Security VII: Status and Prospects, North-Holland (1994), 257-267.
41. R.S. Sandhu. Role hierarchies and constraints for lattice-based access control. 4th ESORICS Symp., Springer (1996), 65-79.
42. A. Stambouli, L. Logrippo. Data flow analysis from capability lists, with application to RBAC. Information Processing Letters, 141(2019), 30-40.
43. A. Stambouli, L. Logrippo. Implementation of a partial order data security model for the Internet of Things (IoT) using Software defined networking (SDN). To appear in the Journal of Cybersecurity and Privacy (MDPI) (2024).

44. A.Thakare, E. Lee, A. Kumar, V.B. Nikam, Y.G. Kim. PARBAC: Priority-attribute-based RBAC model for Azure IoT cloud. IEEE Internet of Things Journal. 7(4) (2020) 2890-900.
45. M. Tripunitara, N. Li. A theory for comparing the expressive power of access control models. Journ. Computer Security, 15 (2007), 231-272.
46. Y.A. Younis, K. Kifayat, M. Merabti. An access control model for cloud computing. Journal of Information Security and Applications. 19(1) (2014) 45-60.
47. G. Zhang, J. Tian. An extended Role based access control model for the Internet of Things. Intern. Conf. on information, networking and automation (ICINA 2010) IEEE, .Vol. 1, 319-323,
48. G. Zhao, D.W. Chadwick. On the Modeling of Bell-LaPadula Security Policies using RBAC. 17th IEEE International Workshops on Enabling Technologies (WETICE 2008) 257-262.