Multi-level models, directed graphs and partial orders in flow control for data secrecy and privacy

Luigi Logrippo

Université du Québec en Outaouais
and University of Ottawa
Ottawa-Gatineau, Canada
luigi@uqo.ca

Abstract. We present the view that the method of multi-level access control, often considered confined in the theory of mandatory access control, instead is central in access control methods, in the sense that it is necessary for data secrecy (i.e. confidentiality) and privacy. This is consequence of a result in directed graph theory showing that there is a multi-level structure in any data flow graph. Then, given the data flow graph of any access control system, it is possible to determine which multi-level access control system it implements. On the other hand, given any desired data flow graph, it is possible to assign subjects and objects to its different levels and thus implement a multi-level access control system for secrecy and privacy. As a consequence, we propose that the well-established lattice model of secure information flow be replaced by a model based on partial orders of components.

Keywords: Security, Secrecy, Confidentiality, Privacy, Access control, Flow control, Mandatory access control, Multi-level security models, Multi-layer security models, Partial orders.

1 Introduction

We present the view that multi-level (ML) access control methods, in the sense that will be defined here, have fundamental importance for access control, data secrecy and data privacy; in fact, any access control system that intends to provide secrecy and privacy must implement such methods. By using a result in directed graph theory, we show that the dataflow graphs representing data flow networks are partial orders of maximal strongly connected components. By generating the data flow graphs of access control systems, one can see what ML systems they implement. By appropriately assigning data to the strongly connected components of dataflow graphs, one can implement ML data security and secrecy methods.

Note that our use of the term *data privacy* in this paper refers to accessibility of private data only. Other research, such as in *privacy by design*, has much wider motivations and requirements [4] and is usually concerned with making it impossible to identify personal information in data sets.Note also that the term *confidentiality* is often considered to be a synonym of *secrecy*.

In Section 2, we present some established concepts on ML systems. In Section 3, we present the mentioned result of directed graph theory. In Section 4 we show that it is possible to find the ML model implicit in any access control system that can be represented by a dataflow graph. In Section 5 we show how, given a desired dataflow graph, it is possible to populate it with subjects and objects thus realizing the dataflow in a concrete system. In Section 6 we make a synthesis of our results, with recommandations.

2 Data flow control and Multi-level access control methods

The study of data flows in access control networks was addressed, directly or indirectly, in many papers in the early years of research on access control methods. Such research was based on the following main ideas:

- Distinction between *secure* or *legal* and *insecure* or *illegal* flows.
- *State-based:* following the famous Bell-La Padula model (BLP) proposed in 1973 [2,3], it was usually assumed that models for secure information flow could be proved secure by reasoning in terms of state transitions, caused by reading and writing operations.
- *Lattice-based*: following an equally famous 1976 paper by Denning [5], it was usually accepted that secure data flows could be guaranteed by lattice-structured models. Entities should considered to be placed in the nodes of a lattice and data should flow along the order relations of the lattice structure. So, much research was concentrated on how to ensure such lattice structuring in information systems [17,8].

Note that there is tension between the state-based concept and the lattice-based concept, the second being relational. This tension leads to complex proof methods [15]. In this paper, we use relational concepts only.

This research was important because it showed how to conceive models that implement both access control and flow control, with a single mechanism. These became known as the *Mandatory access control models (MAC)* [16]. However MAC models seemed to be too restrictive for enterprise applications. Their realm of application is often considered limited to the military or to operating systems. Subsequently, research moved on to flexible models capable of implementing in practice the access control needs of organizations, leading to the Role based access control model (RBAC) [7] and to the Attribute based access control model (ABAC) [10]. Of these, many variants exist but they are mostly conceived for access control and flow control requires further attention.

ML access control methods have been defined and used in the literature and practice in several different ways [18,16]. One of the best-known early proposal for such systems was the mentioned BLP access control model, whose goal is to ensure that in an organization data can move only upwards, from the less secret to the more secret levels. Many variants and generalizations of this concept have been proposed and described. In this work, we generalize the idea of BLP by showing that the entities in data flows are related by partial order relationships represented by directed graphs and data exchanges occur along these relationships.

Further, we react to the limiting view of ML system by demonstrating the opposite view that the fact of being ML is an intrinsic property of any data flow; *secrecy must implemented according to this ML structure, failing which the system will not implement secrecy*. That is, any data security system that is not designed according to the necessary ML structure of its data flow doesn't implement secrecy. This holds for systems specified in Role based access control (RBAC) or Attribute based access control (ABAC) or other models. We underline that data secrecy is a prerequisite to data privacy, which will be not mentioned much in the rest of this paper but will be always implied.

We review briefly here some well-known concepts that lead to our conclusions, before presenting in the next section the graph-thoretical foundation for them.

In any data secrecy system, the following principles are generally accepted:

1) there are at least two types of data: the data to be protected (let us call them *secret*) and the rest (let us call them *public*).
2) there are at least two types of subjects: those that should be able to read secret data, and the others.

This creates a *two-level hierarchy* of data and subjects. The extension to a ML hierarchy of *n-levels* is straightforward, and leads to the following well-known principles.:

3) no read up: subjects at a given level of the hierarchy should be able to read at their own or lower levels only
4) no write down: subjects at a given level of the hierarchy should be able to write at their own or higher levels only.
5) data bases containing high secrecy data can also contain low secrecy data, but not vice-versa.

Further, the theory of non-interference [15] is also based on the existence of at least two levels of data security.

Finally, in many organizations data are routinely classified according to sensitivity levels and personnel are classified according to clearance, with policies defining what clearance is necessary to read or write which data, given their sensitivity levels. Therefore, ML principles relate closely to practical needs.

3 Data flow digraphs as partial orders of components

We use data flow graphs for abstract, relational views of data flows in systems. Data flow graphs are directed graphs, or *digraphs*. In our first presentation of the theory, nodes in our data flow digraphs are *entities* that will represent in a unified way the usual subjects and objects of access control systems. Edges between two entities represent the fact that data can flow between the two entities, e.g. if entity *A* is a subject and entity *B* is an object, then an edge from *A* to *B* means that *A* can write on *B*, while an edge from *B* to *A* means that *A* can read from *B*. This simple view enables us to present synthetically some results that can be adapted to several interpretations and contexts. We also take a pessimistic assumption, common in security theory, by which, if any data at all can flow from *A* to *B*, then any other data of *A* can also flow to B. This leads to assuming the *transitivity* of the dataflow relationship, i.e. if data can flow from *A* to *B* and also from *B* to *C*, then it can flow from *A* to *C*. Finally, it is reasonable to assume reflexivity of data flows.

In Fig. 1, taken from [1] we represent an arbitrary network of entities, where the arrows can be interpreted to denote the data flows that exist in a system. This digraph does not represent a partial order (thus of course not a lattice) because of the presence of symmetric relationships.

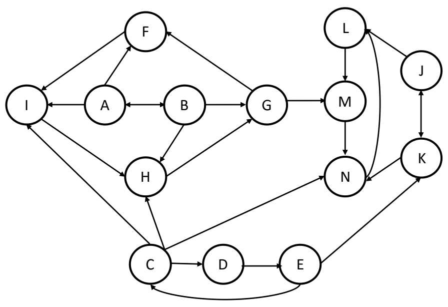

Figure 1. A digraph showing allowed data flows in an organization [1]

We see that entities *A* and *B* can send or receive data from each other. We conclude that *A* and *B* can share all data they have or, the data that one can originate or receive the other can also receive, so they can be considered to be one entity for access control purposes. We will speak of a *strongly connected component {A,B}*, which is also *maximal* because it is not part of another component that would make with it a larger strongly connected component. Henceforth, for conciseness we will use the term *component* to denote a *maximal strongly connected component*. By the same reasoning, entities *F,G,H,I* can receive data from each others, and so they should be considered a component also. Proceeding in this way for the whole digraph, we detect the components *{C,D,E}*, *{L,M,N}* and *{J,K}*, and we can conclude that we have identified all components of the digraph of Fig. 1. Since we have assumed transitivity, we know that all the edges in a component can be thought of as bidirectional, and there is an implied bidirectional edge between *F* and *H*.

Using this information, we can derive the *component digraph* of Fig. 1, shown in Fig. 2. We note that this digraph preserves all the essential information of Fig.1, except for the fact that components have been condensed into one entity. All symmetric relationships have been encapsulated. Basic results of digraph theory [1,9] inform us that

1. this construction is always possible and will always lead to an acyclic digraph, which represents a *partial order* because of the transitivity we have assumed.
2. the component digraph has the same connectedness as the original one; this means that there is a directed path from *X* to *Y* in the original digraph iff there is such a path in the component digraph.

Of course, there can be singleton components consisting of only one node.

We will now get into considerations more typical of discussion of flow control and access control systems.

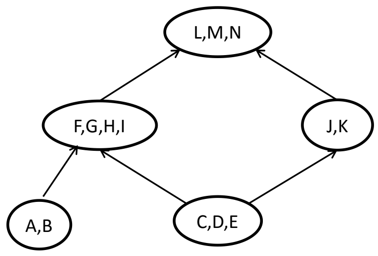

Figure 2. The component digraph of the digraph of Fig. 1

For access control systems and flow control systems this result is very useful because the digraph of Fig. 2 shows more concisely the essential information in Fig. 1, among others that there are some entities that receive no data from other nodes in the digraph, and others that can receive all data. We can also assume that each entity can have some data of its own, which can be shared with other entities according to the available data flow relationships. In the digraph of Fig. 3 we have shown more explicitly how data can circulate in the original digraph. For example, we have shown that all data in entities *A* and *B* and *C,D,E* can be sent to entities *F,G,H,I* so at the higher level we can possibly find all data in all these entities. The nodes in the partial order of Fig. 3 can be thought of as security levels in a ML model.

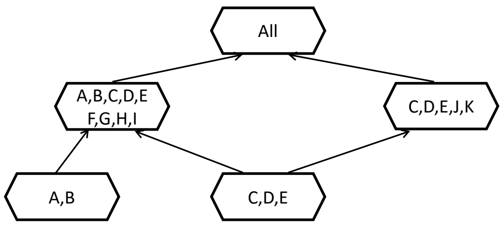

Figure 3. The data flow digraph of the digraphs of Fig. 1 and 2.

We can use this information in several ways. For example, if entities in a node of Fig. 2 represent databases, we know that they can contain the same data and thus could be merged. If they represent subjects, then they can have the same *role* in an RBAC system. Also, the condensed digraph shows us how to reorganize the original digraph, see Fig.4a), where the original digraph is shown as a *partial order of components*, where each component can again be thought of as a security level in a ML system. In Fig. 4b) one further transformation has been done, simply only one edge between any two components has been selected. This could be useful in practice if it is desired to place some protection mechanisms in the edges that run between components.Note that there is some amount of arbitrariness in Fig. 4b), for example instead, or in addition to, the edge *<A,I>* we could have had any edge from any of *{A,B}* to any of *{F,G,H,I}*. But the transitive closures of the digraphs of Figs. 1 and 3, and of all possible digraphs obtained in this way, are the same, they all represent the same data flows.

By looking at these figures, one can see clearly how data circulates in data flow networks specified by digraphs. The symmetric relationships are relegated inside levels. Each component represents a set of entities where there can be complete data sharing, without any secrecy. But then data can also move to the next component up in the partial order, if there is one. Data cannot move down in the partial order, and this implements secrecy. This is the way data circulates in ML networks, and so this allows us to arrive at the possibly surprising conclusion that *every data flow digraph is a ML network of components*.

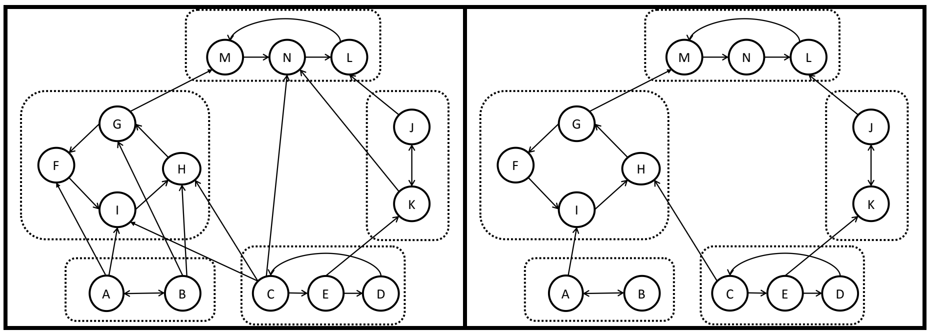

Figure 4a) and 4b). The digraph of Fig. 1 reorganized and then simplified

Subjects and objects can be associated with the nodes of Fig.3 just as they were assumed to be in Fig.1. Access control matrices will have to be constructed or roles with permission lists, or other policies. If the digraph must be implemented as a distributed network, then routing lists will have to be constructed. Encryption mechanisms can also be used to establish different data flows. Depending on the method used, the reduced number of edges in Fig. 4b) might make the task easier. The access control system for these digraphs can be constructed in the following way:

1) data flow is permitted between any two elements of a component
2) data flow is permitted between two elements of different components according to the partial order relationship, as represented by the paths in the digraph, for example data can flow, directly or indirectly, from *B* to *M*.

This is the same thing that should be done to construct the access control system for the digraph of Fig. 1, however this construction has made it possible to see clearly the underlying partial order logic.

There are efficient algorithms to obtain component digraphs. The best known is perhaps Tarjan's algorithm for finding the components of a graph. The time complexity of Tarjan's algorithm is linear on the number of edges plus the number of nodes [20].

It is interesting to observe that similar methods have a long history of being used for data flow analysis in programs, where one of the main concerns is to identify the main components in the data flows [13].

4 Finding levels in existing systems

Consider a network with five subjects *S1* to *S5* and five objects *O1* to *O5*. Diagrams like this can be obtained for access control systems specified by means of access control matrices, RBAC permissions [14], etc. By using the notations *CR* for *can read*, and *CW* for *can write* [11,12], the permissions for the subjects in this network are as in Table 1:

Table 1: Read-Write relationships for our example

CR(S1,O1)	CW(S1,O2)
CR(S2,O2)	CW(S2,O3)
CR(S3,O4)	CW(S3,O3)
CR(S4,O3)	CW(S4,O3)
CR(S5,O2)	CW(S4,O4)
CR(S5,O5)	CW(S5,O5)

Fig. 1 gives a digraph representation of this network, using ovals for subjects and rectangles for objects.

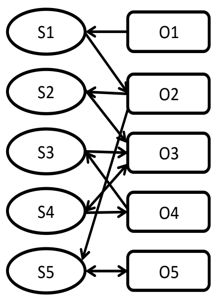

Fig. 5. An access control network

This an arbitrarily constructed network, and not one constructed to prove our conclusions. By using the principles we have presented, the digraph of Fig. 5 can be shown as in Fig. 6a).

In Fig. 6 we see clearly the partial order of components implicit in Fig. 5. We assume that all data are in the objects or databases *O1-O5* and we show where the data of each of these database can start and end in the network's dataflow. On the right-hand side we show similar information in a more schematic form. Using the terminology of [11,12], from Fig.6a) it is clear that databases *O3* and *O4 can store* the same data, thus possibly they can be merged. Subjects *S3* and *S4* also *can know* the same data, and so it is possible to give them the same role.Thus this view has implications for role engineering. But we will leave such considerations to future work.

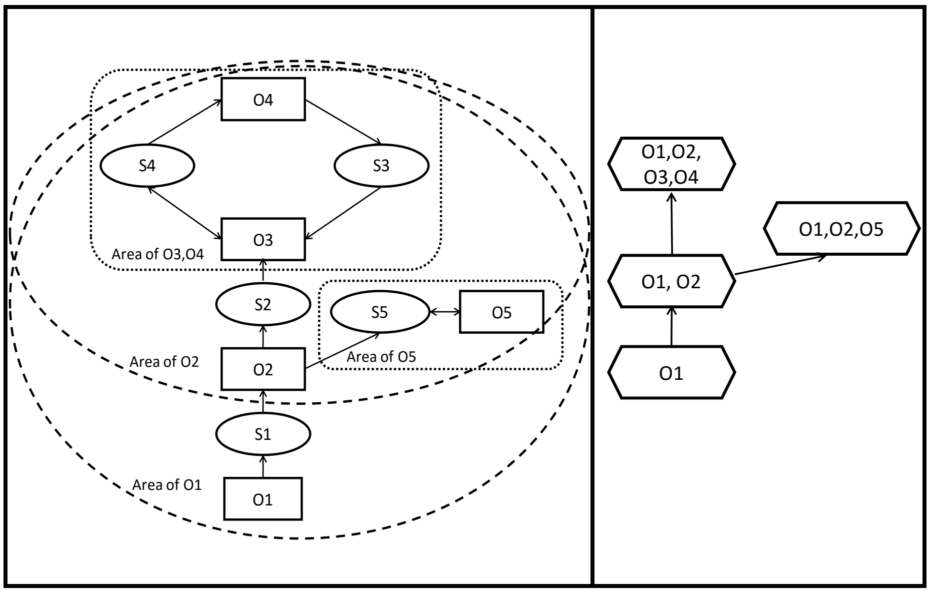

Fig.6.a) Components and b) data flows for the example of Fig.5

Most important from the secrecy point of view, we see where in the network the data in the various objects can be known or stored. The most secret data are the ones in objects *O3*, *O4* and *O5*, which can be known or stored in most internal (or topmost) areas only, and the least secret are the data in *O1* which can go anywhere.This may not be intended by the designers of the system of Table 1 or Fig. 5 but is a necessary consequence of the structure of the system.

This analysis can be done in practice on systems of moderate size. Reference [19] presents, analytically and by simulation, the fact that efficient algorithms exist to do it.

5 Designing multi-level systems

The previous discussion has not been helpful from the design point of view. In the example of the previous section, we have made some observations about the secrecy status of some data, but this was an observation on a randomly generated network of entities, it was not the result of design decisions. The initial representation of the system of Fig. 5 did not show clearly that the data in *O3, O4* or *O5* have the least visibility, thus are the most secret.

Once again, we will proceed by example. We wish to design an access control network for the following application. We have two banks in conflict of interest, *Bank1* and *Bank 2*. *Bank1* has only one category of data, called *B1*. However *Bank 2* has public data labelled *B2P* that can be available to anyone, and secret data *B2S* that should be available only to its own employees. There is also a *Company 1* that collaborates with *Bank 1* and shares all its data *C1* with *Bank 2*. However *Bank 2* does not want its secret data *B2S* to be known to *Company 1*. A possible dataflow diagram for this system is shown in Fig 7.

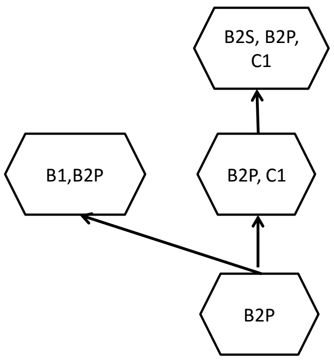

Figure 7. Data flow in a hypothetical network

We now populate this dataflow with subjects and objects, or databases and employees.This can be done in many ways. We will use a very simple structure with one database for each possible data contents and one employee for each database. We use the following notation: *Bob:{B1,B2P}* means that employee *Bob* has clearance only to read the data of the types indicated, and similarly *Bk1:{B1,B2P}* means that database *Bk1* can store only data of the types indicated. The populated diagram is shown in Fig. 8.

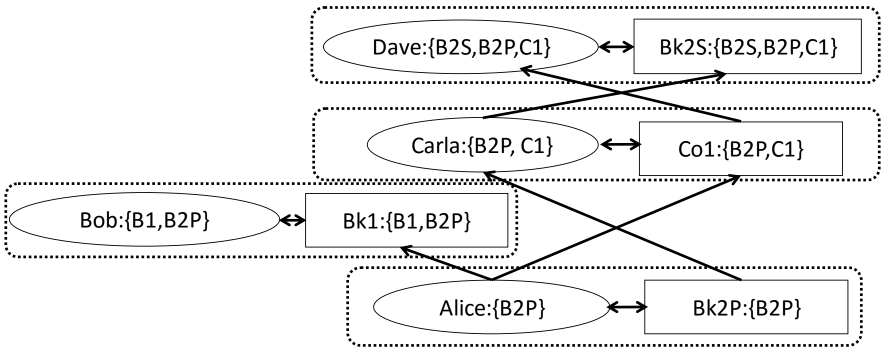

Figure 8. A network of entities for the data flow of Fig. 7

Many structural details are not specified in Fig. 8. We can imagine that: *Bob* is an employee of *Bank 1*; *Alice* is an employee of *Bank 2* in charge of making available public information for *Bank 2* from a database that she administers for this purpose; *Carla* is an employee of *Company 1* and *Dave* is an employee of *Bank 2*. Arrows that can be deduced by transitivity are not shown, i.e. we can imagine that *Alice* is also authorized to write directly on *Bk2S*. This simplification can be considered to be inadequate from the security point of view, since in Fig. 8 the transfer of data from *Alice* to *Bk2S* depends on decisions by *Carla*. However our dataflow diagrams show only the *possibility* of data transfers.

From this example, it should be clear that the classical BLP model can be obtained, in its essential aspects. as a special case of our construction. If we wish a BLP system with three levels: *Public, Confidential* and *Secret*, then the necessary labels are: *{Public},{Public, Confidential }, {Public, Confidential, Secret}*.

6 Synthesis and conclusions

In conclusion, we believe that, using a basic result of digraph theory, we have established intuitively the following facts:

- Access control systems and dataflow systems that can be described as directed graphs define partial orders of components
- No data secrecy or data privacy is possible within a single component
- Data available in one component will also be available in the following components in the partial order
- In order to have data secrecy or privacy, a system must have at least two components
- Data secrecy or privacy can then be defined in terms of data being available only in some components
- Data should be distributed among the components according to their level or secrecy, with the most secret data in the top components of the partial order.

While the *sufficiency* of these facts as principles for the design of data security systems has been understood for a long time, their *necessity* has been overlooked (except for the acceptance in theory of the lattice model, to be further discussed below). Any system that intends to protect data security or data privacy in this sense must implement an appropriate partial order; this is done by construction in strict BLP systems and similar ones, but must also be implemented in systems using other access control methods, such as RBAC or ABAC. Partial orders can be implemented by using appropriate policies, access control matrices, encryption or, in truly distributed systems, by using appropriate data forwarding policies.

Some difficulties present themselves, of course.

A common objection against ML methods is that the constraint of allowing data flow in one direction only is impractical. However we have shown that all directed graphs describe unidirectional flows in their partial orders, and that this is necessary for secrecy. Note however that an essential assumption in our discussion has been that when a flow is allowed between two entities, all data can move from one to the other by reading and writing operations. This view can be refined by distinguishing among types of data, limiting the operations to specific types of data and constructing different data flow digraphs for different types of data. For example, in an organization we could have tables showing salaries with names, and tables showing salary statistics without names. The allowed data flow for one type of table will normally be different from the one of the other. Different data flow diagrams, which means different partial orders, can then coexist for different types of data. In the process called *sanitization* sensitive data can be transformed into less sensitive ones and *declassified*, with different data flow requirements. Typically, salary tables with names could arrive at an office at the top of one partial order, and this office could produce statistics that should be available for everyone, placing itself at the bottom of another partial order. The office must be certified to produce such sanitization and declassification. The model ABAC can define attributes and policies for data items at different levels of granularity, and thus can be used to define different dataflows for different types of data.

Another major difficulty is the fact that many modern access control systems do not define fixed data flows. These can change by administrative changes or environmental changes, leading to changes in the values of Boolean conditions. Corresponding data-flow graphs can be very complex in practice, with edges labelled by conditions. Changes must be conceived in a way that they do not modify crucial partial order relationships.How to achieve this appears to be an interesting research topic.

Established theory considers lattices as the basic structuring model for data security [5,8,17], however it seems that this view must be corrected. Lattices are restrictive, in the sense that they require the presence of joins and meets; often, in order to accommodate this need, unnecessary subjects and object are introduced. They are also restrictive in the sense that they forbid symmetric relationships, which in the digraph model are encapsulated in components. While partial orders can always be extended to lattices, this extension requires the introduction of entities to complete the structure, entities that may be unnecessary, difficult to justify, or unwanted.

Finally, we propose that the ML model as outlined here be seen as the obligatory design pattern [6] for systems intended to enforce strict data secrecy and privacy.

We have remained on an intuitive level, to avoid tying our discussion to a specific formalism. We are continuing work towards a suitable formalism to reason about secrecy properties [12], for which a first version was presented in [11].

Acknowledgment. This work was partially supported by a grant of the Natural Sciences and Engineering Research Council of Canada. The author is indebted to Sofiene Boulares and Abdel Stambouli for many useful discussions, and to Guy-Vincent Jourdan for useful comments on the draft copy.

References

1. J. Bang-Jensen, G.Z. Gutin. *Digraphs. Theory, algorithms and applications*, Springer, 2010.
2. D. E. Bell, L. J. La Padula. Secure computer systems: unified exposition and Multics interpretation. TR MTR-2997 Rev.1, Mitre Corporation, 1976.
3. D. E. Bell. Looking back at the Bell-La Padula model. 21st Ann. IEEE Comput. Security Appl. Conf., 2005 (on line, no page numbers).
4. A. Cavoukian. Privacy by design. The 7 foundational principles. White Paper, Information and Privacy Commissioner of Ontario, Canada, 2009.
5. D.E. Denning. A lattice model of secure information flow. Comm. ACM 19(5), 1976, 236-243.
6. E. Fernandez-Buglioni. *Security patterns in practice.* Wiley, 2013.
7. D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli. *Role-based access control.* 2nd Ed. Artech House, 2007.
8. S.N. Foley. Aggregation and separation as noninterference properties. Journal of Computer Security 1 (2), 1992, 159-188.

9. F.Harary, R.Z.Norman, D.Cartwright. *Structural models. An introduction to the theory of directed graphs.* Wiley, 1966.
10. V.C. Hu, D.R. Kuhn, D.F. Ferraiolo. Attribute-based access control. Computer, 48(2), 2015, 85-88.
11. L Logrippo. Logical Method for Reasoning about Access Control and Data Flow Control Models. Proc. of the 7th International Symposium on Foundations and Practice of Security, 2014. FPS 2014, LNCS 8930 (2015), 205–220.
12. L.Logrippo. A first-order logic formalism for access control and flow control,with application to multi-level access control. In preparation.
13. F. Nielson, H.R. Nielson, C. Hankin. *Principles of program analysis.* Springer, 2004.
14. S.L. Osborn. Information flow analysis of an RBAC system, Proc. of the seventh ACM symposium on Access control models and technologies, (SACMAT 2002), 163-168.
15. J. Rushby. Noninterference, transitivity, and channel-control security policies. TR CSL-92-02. Computer Science Lab., SRI International, Menlo Park, CA, 1992.
16. P. Samarati, S.De Capitani di Vimercati. Access control : policies, models and mechanisms. FOSAD 2000: Foundations of Security Analysis and Design. Springer, 137-196.
17. R. Sandhu. Lattice-based access control models. Computer 26(11), 1993, 9-19.
18. R. Smith. Multilevel Security. In H. Bidgoli, editor, Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management, Vol. 3, Ch. 205. Wiley, 2005.
19. A. Stambouli, L. Logrippo. Data flow analysis with access control matrices. In preparation.
20. R. E. Tarjan. Depth-first search and linear graph algorithms, SIAM Journal on Computing, 1 (2), 1972, 146–160.