Multi-level access control, 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, is in fact necessary for data secrecy (i.e. confidentiality) and privacy. This is consequence of a result in directed graph theory showing that there is a partial order of components in any data flow graph. Then, given the data flow graph of any access control system, it is in principle 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 data 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. Applications to Internet of Things and Cloud contexts are briefly mentioned.

Keywords: Security, Secrecy, Confidentiality, Privacy, Access control, Flow control, Mandatory access control, Multi-level security, Multi-layer security, Internet of Things, Cloud.

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 data flow 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 data flow graphs, one can implement ML data security and secrecy.

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

Extended version of paper appeared in: Foundations and Practice of Security. FPS 2017. LNCS 10723 (2018), 111-123. Springer, DOI https://doi.org/10.1007/978-3-319-75650-9_8.

considered to be a synonym of *secrecy*. We subscribe to the view by which *"the fundamental nature of a privacy violation is an improper information flow"* [15] and we propose new analysis and design methods to enable only proper flows. Data secrecy is a prerequisite to data privacy, and the latter will be implied in the rest of this paper.

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 in principle to find the ML model implicit in any access control system that can be represented by a data flow graph. In Section 5 we show how, given a desired data flow graph, it is possible to populate it with subjects and objects thus realizing secrecy-preserving data flows in concrete systems. In Section 6 we make a synthesis of our results, with recommendations.

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 [14]. 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 [6], it was usually accepted that secure data flows could be guaranteed by imposing a lattice-structure on the data flow. Entities should be placed in the nodes of a lattice and data should flow along the order relations of the lattice structure. So, much research was directed to ensuring such lattice structuring in information systems [24,20,10].

This research introduced models that implement both access control and flow control, with a single mechanism. These became known as the Mandatory access control models (MAC) [23], and are usually considered to include the ML methods. 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, and even there, with some relaxations. 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) [9] and to the Attribute based access control model (ABAC) [12]. 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 different ways [25,23]. One of the best-known early proposal for such methods was the 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.

In this work, we react to the limiting view of ML system by demonstrating the opposite view that being ML is an intrinsic property of any data flow; so *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 intrinsic ML structure of its data flow cannot implement secrecy. This holds for systems specified in RBAC or ABAC or other models. We will see that this view implies a significant correction to the view of ML structures as lattices.

We review briefly here other 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*); they are usually segregated to different databases.
2) there are at least two types of subjects: those that should be able to know secret data, and the others.

This creates a *two-level hierarchy* of data and subjects. The extension to hierarchies 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) databases containing high secrecy data can also contain low secrecy data, but not vice-versa.

Further, the theory of non-interference [22] is also based on the existence of at least two levels of data secrecy.

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 methods are necessary for data secrecy, and also relate closely to practical needs.

The combination of state-based concepts and relational concepts (as in the lattice model) leads to complex proofs. In this paper, as in [16,17], we use relational concepts only, while acknowledging that state-based concepts can be more expressive for modeling attacks [12].

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 represented here as 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 data flow 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*. The transitivity of data flows is a property that cannot be postulated in general [22], but, since it is based on the mentioned pessimistic assumption, cannot lead to systems that are under-protected. Finally, it is reasonable to assume the *reflexivity* of data flows.

In Fig.1a), taken from [1] we represent an arbitrary digraph, where the arrows can be interpreted to denote possible data flows among entities in a system, perhaps in an Internet of Things context.This digraph does not represent a partial order (thus of course not a lattice) because of the presence of symmetric relationships; however it is easy to see that it defines a data flow where all data can end up in entities L,M,N.

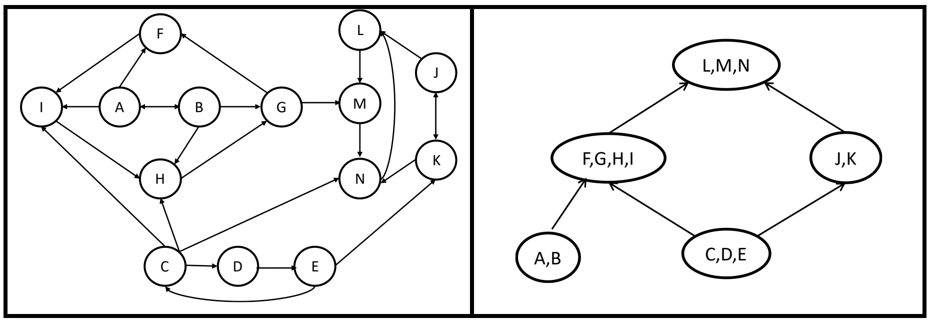


Figure 1a) and 1b). A digraph showing allowed data flows in a network and its component digraph [1]

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 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 to form a component also. Proceeding in this way for the whole digraph, we detect the components *{C,D,E}*, *{L,M,N}* and *{J,K}*. Since we have assumed transitivity, all the edges in a component can be thought of as bidirectional, and there is an implied bidirectional edge between *F* and *H*. Of course, there can be singleton components consisting of only one node.

Using this information, we can derive the *component digraph* of Fig. 1a), shown in Fig. 1b). We note that this second digraph preserves all the essential information of the first, except for the fact that components have been condensed into one node: symmetric relationships, which are equivalence relationships, have been encapsulated. Elementary results of digraph theory [1,11] 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 reflexivity and transitivity we have assumed;
2. the component digraph has the same connectedness as the original one, in the sense that there is a directed path from *X* to *Y* in the original digraph iff there is such a path between components containing *X* and *Y* in the component digraph.

This leads us to conclude that *any data flow digraph can be understood as a partial order of components*.

For access control systems and flow control systems this result is very useful because the digraph of Fig.1b) shows more concisely the essential information in Fig. 1a. We can also assume that each entity can have some data of its own (we say that these data *originate* in the entity), which can be shared with other entities according to the data flow relationships. The digraph of Fig. 2 shows concisely how data can flow in the original digraph. A comparison between Fig. 1b) and Fig. 2 shows that the greater entities in the partial order can have available more data, also that data originating higher in the partial order can be available to fewer entities.

The nodes in the partial order of Fig. 2 can be thought of as *security levels* in a ML model.

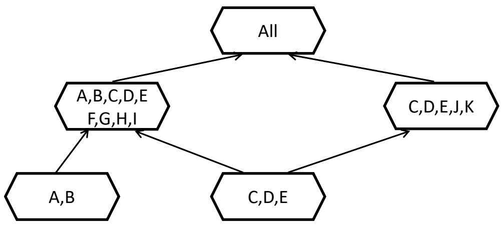
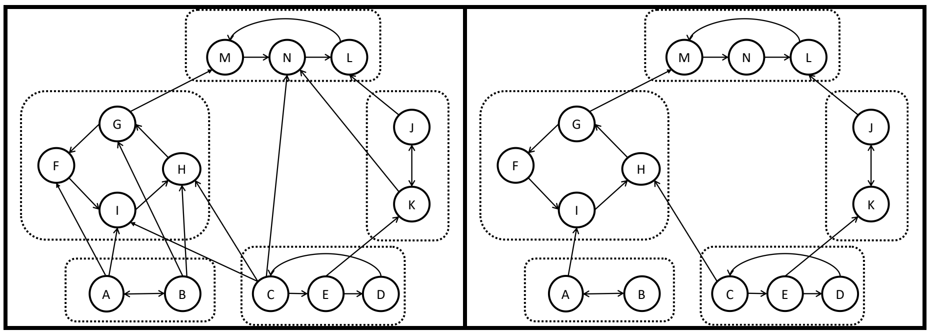


Figure 2. The data flow digraph of the digraphs of Fig. 1.

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 perhaps be merged; if they represent subjects, then they can have the same *role* in an RBAC system; if they represent roles, they can perhaps be merged. Merging decisions however should be conditional to administrator's approval because there may be reasons not to implement them. Also, the condensed digraph of Fig. 1b) shows us how to reorganize the original digraph, see Fig.3a), where the original digraph is shown more explicitly as a partial order of components, where each component can again be thought of as a security level in a ML system. In Fig. 3b) one further transformation has been done: only one edge between any two components has been selected, also relationships between components are implied when they can be derived by transitivity. This could be useful in practice if it is desired to place protection mechanisms in the edges that run between components.Note that there is some amount of arbitrariness in Fig. 3b), 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. 1a) and 3, and of all possible digraphs similarly obtained, are the same, they all represent the same data flows.

Each component in these figures 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 flow in ML networks, and so *we define ML networks as partial orders of components*, leading to the conclusion that *any data flow digraph can be understood as a ML network*.



Figures 3a) and 3b). The digraph of Fig. 1 reorganized and then simplified

Generic entities or subjects and objects can be associated with the nodes of Fig.3 just as they were assumed to be in Fig.1. The access control systems 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 relationships represented by the paths in the original or derived digraphs, for example data can flow, directly or indirectly, from *B* to *N*.

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. 3b) might make the task easier. These are the same things that should be done to construct the access control system for the digraph of Fig. 1a), however our construction has made it possible to see clearly the underlying partial order logic.

There are efficient algorithms to obtain component digraphs. For example, the time complexity of the well-known algorithm reported in [27] is linear on the number of edges plus the number of nodes.

It is interesting to observe that similar methods have a 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 [19].

4 Finding levels in existing access control systems

Table 1 gives the permissions for a network with five subjects *S1* to *S5* and five objects *O1* to *O5*, using the notations *CR* for *can read*, and *CW* for *can write* [16,17].This an arbitrarily constructed network, and not one constructed to prove our conclusions. Diagrams like this can be obtained for access control systems specified by means of access control matrices, RBAC permissions [21], etc.

Table 1: Read-Write relationships for the network of Fig. 4

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

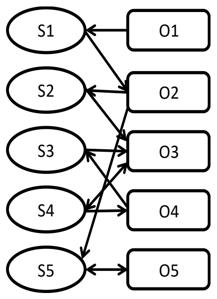


Fig. 4. An access control network

For uniformity and to justify transitivity, we can think that all edges represent a single transitive relationship *can flow* [21] rather than two distinct relationships as presented in Table 1.It remains that, in conformity with the concepts of access control, this is a bipartite digraph of two different types of entities: *subjects* and *objects*. Earlier we said that data can be available to entities; henceforth we will also say that subjects *can know* data and objects *can store* them.

By using the principles we have presented, the digraph of Fig. 4 can be shown as in Fig. 5a). In Fig. 5b) we see clearly the partial order of components implicit in Fig. 4. Using the terminology of [16,17], from Fig. 5a) 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.

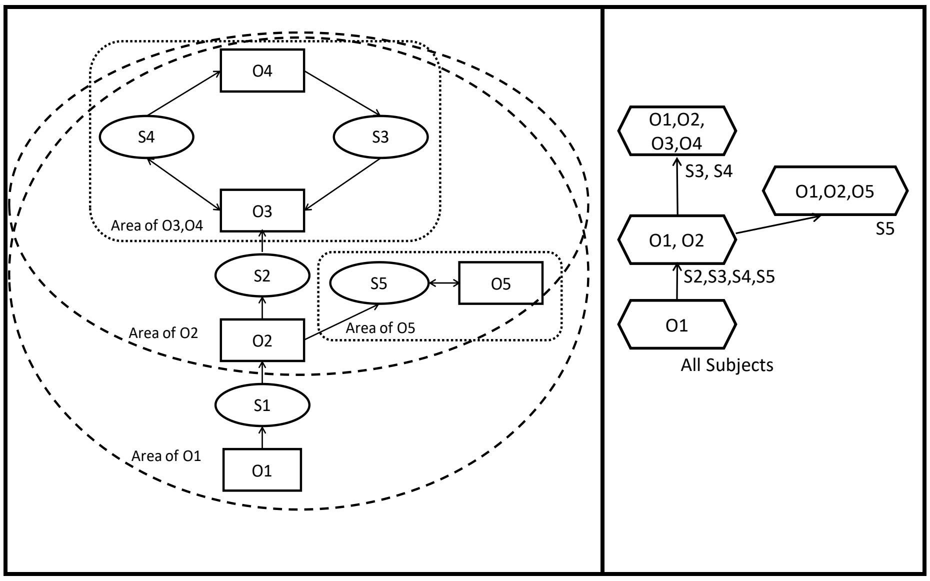


Fig.5.a) and b). Components and data flow for the example of Fig.4

Assuming that all data are in the objects or databases *O1-O5*, we show where the data of each of these databases can possibly be available in the network, by using the terminology *'area of'*. We see that the data of *O1* can be available anywhere in the network, while the most secret data are the ones originating in objects *O3*, *O4* and *O5*, which can be available in the most internal (or topmost) areas only.This may not be intended by the designers of the system of Table 1 or Fig. 4 but is a necessary consequence of the structure of the data flow.

The concept of area can be used to define a *formal notion of secrecy*: the data in an object are a secret of the subjects and objects in the area of the object.

Fig. 5b) is also useful to answer *inference* questions [7]: for example, assuming that the combined knowledge of data in *O1* and *O5* can lead to further knowledge, who in the system of Fig. 4 can achieve such knowledge? Clearly, it is only *S5*. Further, this inferred information can be available only to *S5* and *O5*.

Because of the efficient algoritms we have mentioned, this analysis can be done in practice on systems of moderate size. Reference [26] presents this fact, analytically and by simulation.

5 Constructing 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. 4 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, possibly in a Cloud context. We have two banks in conflict of interest, *Bank1* and *Bank 2*. *Bank1* has only one category of data, called *B1*, which it wishes to keep private. However *Bank 2* has public data labelled *B2P* that can be available to any entity, and secret data *B2S* that should be available only to its own entities. There is also a *Company 1* that collaborates with *Bank 2* and so shares all its data *C1* with *Bank 2*. However *Bank 2* does not want its secret data *B2S* to be known to *Company 1*, nor to *Bank 1* of course. Note that here we have added another type of entity, which we can call *organization*, and which will turn out to be a set of subjects and objects. Note also that we have expressed both *need to know* and *conflict* requirements. A Boolean analysis of these requirements leads to the data flow diagram shown in Fig 6.

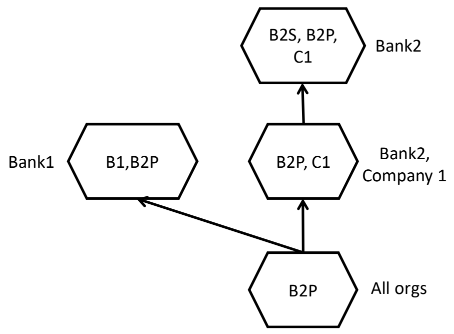


Figure 6. Data flow in a hypothetical network

We now populate this data flow with subjects and objects, or employees and databases.This can be done in different 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. Taking *Bob* as an employee of *Bank 1*, in charge of the bank's database; *Alice* as an employee of *Bank 2* in charge of making available public data for *Bank 2* from a database that she administers for this purpose; *Carla* as an employee of *Company 1* and *Dave* as an employee of *Bank 2*, the populated diagram is shown in Fig. 7.

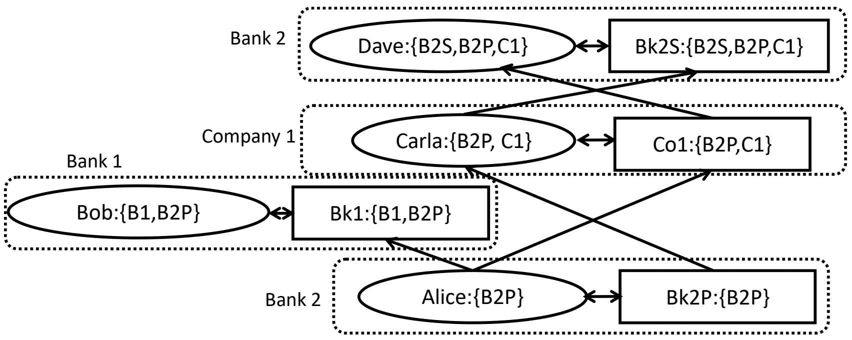


Figure 7. A network of entities and organizations for the data flow of Fig. 6

Arrows that can be inferred by transitivity are not shown in Fig. 7, 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. 7 the transfer of data from *Alice* to *Bk2S* depends on decisions by *Carla*. We have mentioned that transitivity cannot be given for granted in data flow systems. However our data flow diagrams show only the *possibility* of data flows, based on our pessimistic hypothesis.

Many other realizations of the original requirements are of course feasible, e.g. *Bank 1* may wish to keep separate *B1* and *B2P* data.

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}*.

This mechanism of constructing data flows by using label sets is powerful. We have seen above how it can be used to express conflicts. It can be used to express other types of constraints, but this is left to future papers.

6 Synthesis and conclusions

In conclusion, using a basic result of digraph theory, we have established intuitively the following facts for access control and data flow systems that can be described as transitive, directed graphs:

- They define partial orders of components.
- These directed graphs and partial orders can be obtained efficiently from access control policies in some practical cases.
- No data secrecy is possible within a single component, since in each component, all entities can have available the same data.
- Data available in a component can also be available in the greater components in the partial order; data originating in a component cannot be available in lower components.
- As we move up in the partial order, the amount of data that can be available there will monotonically increase; also the number of entities that can have available data originating there will monotonically decrease.

- In order to have data secrecy, a system must have at least two components.
- Data secrecy can then be defined in terms of data being available only in *some* components.
- For data secrecy, data *must* be distributed among the components according to the desired levels of secrecy, with the most secret data originating in the top components of the partial order (from where they cannot move down). This will allow, *all* and *only*, *legal* or *secure* data flows.

While the *sufficiency* of some 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 secrecy in this sense must implement appropriate partial orders of components; 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. Implementation can be done by using appropriate role assignments [20], policies, access control matrices, encryption or, in truly distributed systems, by using data forwarding policies.

As shown, these principles can be used not only for data protection within an organization, but also for networks of organizations (Section 4), in the Internet of Things and in Cloud environments where data of different ownerships coexist.

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 multi-level, unidirectional flows in their partial orders, and that this is necessary for secrecy. But this was based on the pessimistic assumption 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, with different partial orders, 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. Allowed data flows will normally be different for the two types of data. The two concepts of *data declassification* and *trusted subjects* contribute towards solutions [23,18]. 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 (a trusted subject) could produce statistics available for everyone, thus writing the data at the bottom of another partial order. Or, a director general could receive secret information but, being trusted, can also place itself at a lower level to distribute directives. Different data flow relationships for different types of data can be specified with any access control method if one supposes that different types of data are put in different objects. Access control systems with data labeling offer more flexibility [5,18].

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. Graphs that describe such flows can be complex, with edges labelled by conditions. Changes must be conceived in a way that they do not modify essential partial order relationships. How to achieve this appears to be an interesting research topic.

Established theory considers lattices as the basic structuring model for secure data flows [6,10,24], however it seems that this view must be corrected. Lattices are restrictive, in the sense that they require the presence of joins and meets. Partial orders can be extended to lattices but in order to do so, unnecessary entities may have to be introduced. For example, to extend the partial order of Fig.6 into a lattice, it is necessary to add a node containing both B1 and B2S, contradicting the requirements without any advantage; such a node must be excluded from the solution of Fig. 7. Further, to extend the partial orders of Fig. 1b) (or Fig. 5b) to lattices it is necessary to add superfluous empty components that do not correspond to any entities. Lattices are also restrictive in the sense that they forbid symmetric relationships, which in our model are encapsulated in components. In [6] it is assumed that equivalent nodes can be merged, but in practice this may not be possible. In contrast, partial orders of components always exist in data flows that can be represented as digraphs, without any extensions.

Therefore, the ML model as outlined here should be seen as the obligatory design pattern [8] for systems intended to enforce strict data secrecy.

Finally, it should be mentioned that data flow theory has many aspects, by which our definitions can be considered to be very simplified. Still, this simplified view leads to results of practical significance for the analysis and synthesis of secrecy systems, as shown by our examples.

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 [17], for which a first version was presented in [16].

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 Abdelouadoud 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, p. 17 and Fig. 1.12.
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. E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati. A fine-grained access control system for XML documents. ACM Trans. Inf. Syst. Secur. 5, 2 (2002), 169-202.
6. D.E. Denning. A lattice model of secure information flow. Comm. ACM 19(5), 1976, 236-243.

7. C. Farkas, S. Jajodia. The inference problem: a survey. SIGKDD Explor. Newsl. 4, 2 (2002), 6-11.
8. E. Fernandez-Buglioni. *Security patterns in practice.* Wiley, 2013.
9. D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli. *Role-based access control.* 2nd Ed. Artech House, 2007.
10. S.N. Foley. Aggregation and separation as noninterference properties. Journ. Computer Security 1 (2), 1992, 159-188.
11. F.Harary, R.Z.Norman, D.Cartwright. *Structural models. An introduction to the theory of directed graphs.* Wiley, 1966, Chapter 3.
12. V.C. Hu, D.R. Kuhn, D.F. Ferraiolo. Attribute-based access control. Computer, 48(2), 2015, 85-88.
13. M. Jaume, V. Viet Triem Tong, L. Mé. Flow based interpretation of access control: Detection of illegal information flows. Proc. 7th International Conference, ICISS 2011, LNCS 7093 (2011), 231-245.
14. C. E. Landwehr. Formal models for computer security. ACM Computer Surveys 13(3) (1981) 247-278.
15. C. E. Landwehr. Privacy research directions. Comm. ACM 59(2) (2016) 29-31.
16. L Logrippo. Logical Method for Reasoning about Access Control and Data Flow Control Models. Proc. of the 7th Intern. Symp. on Foundations and Practice of Security (FPS 2014), LNCS 8930 (2015), 205–220.
17. L.Logrippo. A first-order logic formalism for access control and flow control,with application to multi-level access control. In preparation.
18. A.C. Myers, B. Liskov. Protecting Privacy Using the Decentralized Label Model. ACM Trans. Softw. Eng. Methodol. 9(4) (2000), 410-442.
19. F. Nielson, H.R. Nielson, C. Hankin. *Principles of program analysis.* Springer, 2004.
20. S.L. Osborn, R. Sandhu, Q Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. ACM Trans. Inf. Syst. Secur. 3(2) (2000), 85-106.
21. S.L. Osborn. Information flow analysis of an RBAC system, Proc. of the 7th ACM symposium on Access control models and technologies, (SACMAT 2002), 163-168.
22. J. Rushby. Noninterference, transitivity, and channel-control security policies. TR CSL-92-02. Computer Science Lab., SRI International, Menlo Park, CA, 1992.
23. P. Samarati, S.De Capitani di Vimercati. Access control : policies, models and mechanisms. Foundations of Security Analysis and Design (FOSAD 2000). Springer, 137-196.
24. R. Sandhu. Lattice-based access control models. Computer 26(11), 1993, 9-19.
25. R. Smith. Multilevel Security. In H. Bidgoli, editor, Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management, Vol. 3, Ch. 205. Wiley, 2005.
26. A. Stambouli, L. Logrippo. Data flow analysis from capability lists,with application to RBAC. Information Processing Letters 141 (2019) 30–40. Modified version: http://www.site.uottawa.ca/~luigi/papers/18_IPL.pdf
27. R. E. Tarjan. Depth-first search and linear graph algorithms, SIAM Journal on Computing, 1(2) (1972), 146–160.