Configuring data flows in the Internet of Things
for security and privacy requirements

Luigi Logrippo

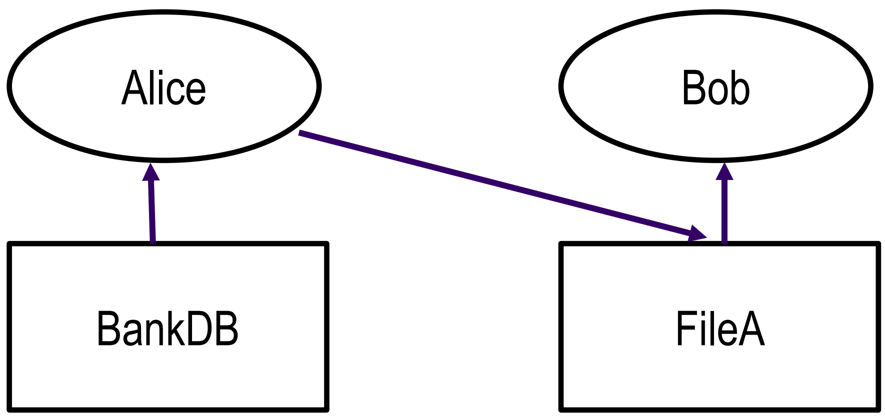
Abdelouadoud Stambouli

Université du Québec en Outaouais

Foundation and Practice of Computing, November 2018

Data flow control vs access control

- Access control:
  - Controls access of subjects to objects
- Data flow control:
  - Controls where data can end up in a network



  - Bob can know the data in BankDB although it has no direct access to it (see Trojan horse etc.)

MAC

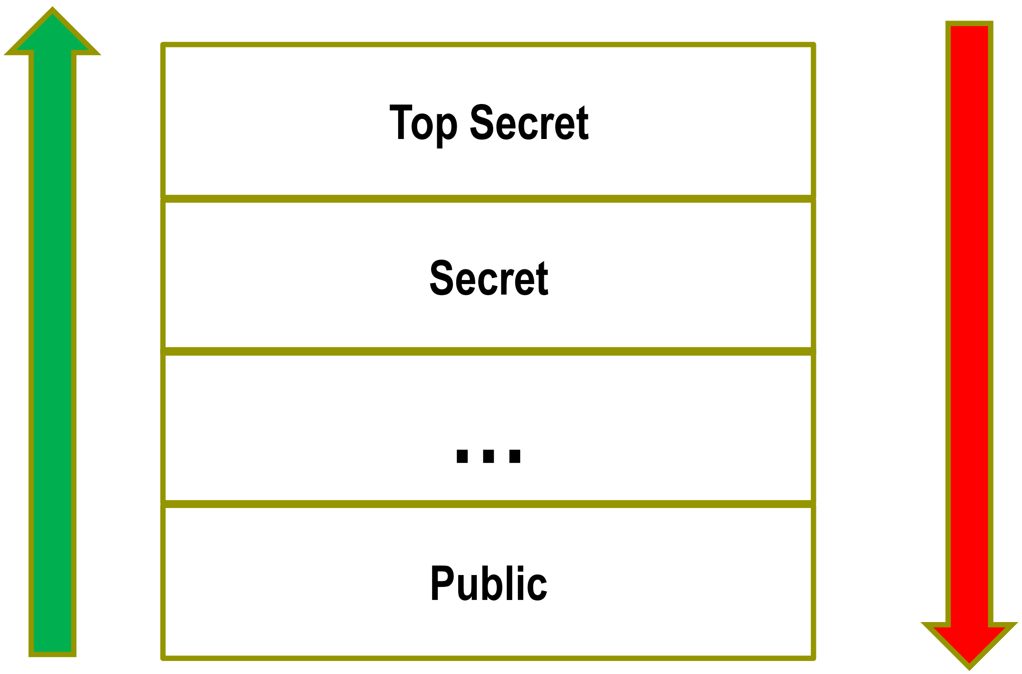
Mandatory Access Control data security models

- Subjects and objects are labelled
  - Subjects are labelled by the data that they can read
  - Objects are labelled by the data that they can contain
- There are label-based rules that determine
  - Which subjects can read which objects
  - Which subjects can write on which objects
- Simultaneously guarantees access control and flow control
  - Often considered too restrictive

The Bell-La Padula model

Read down

Write up



No read up

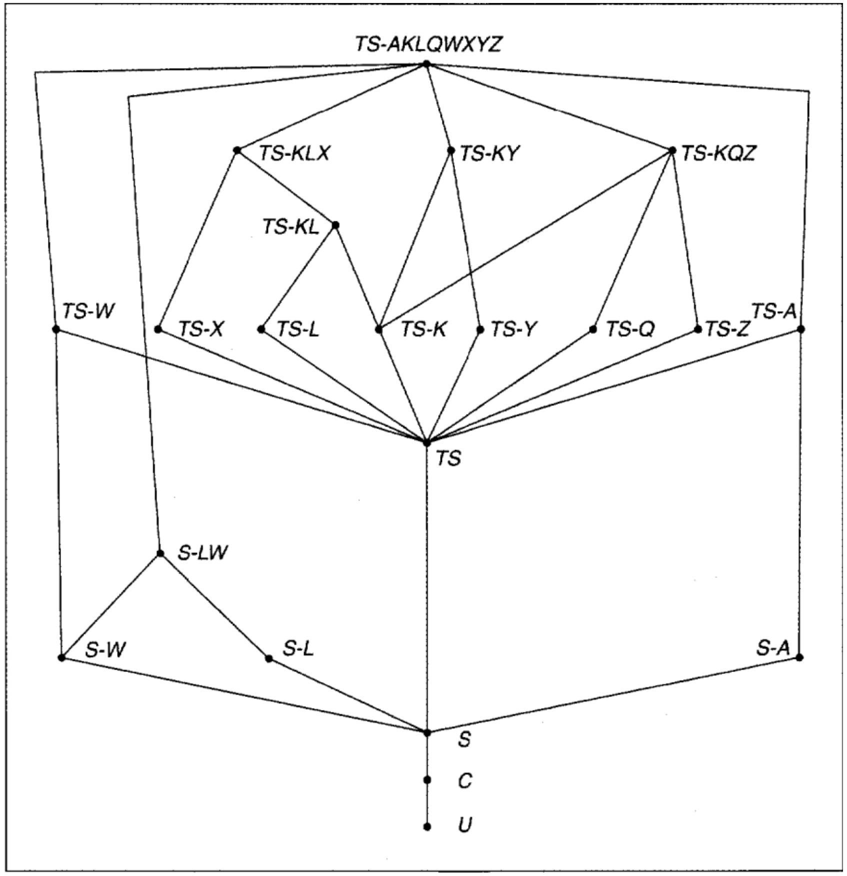
No write down

Data can flow only upwards

We generalize this model

An established generalization: Lattice model
(Denning 1974)

- Data can move only upwards in the lattice



Source: Sandhu, 1993

Success and critique of the lattice model

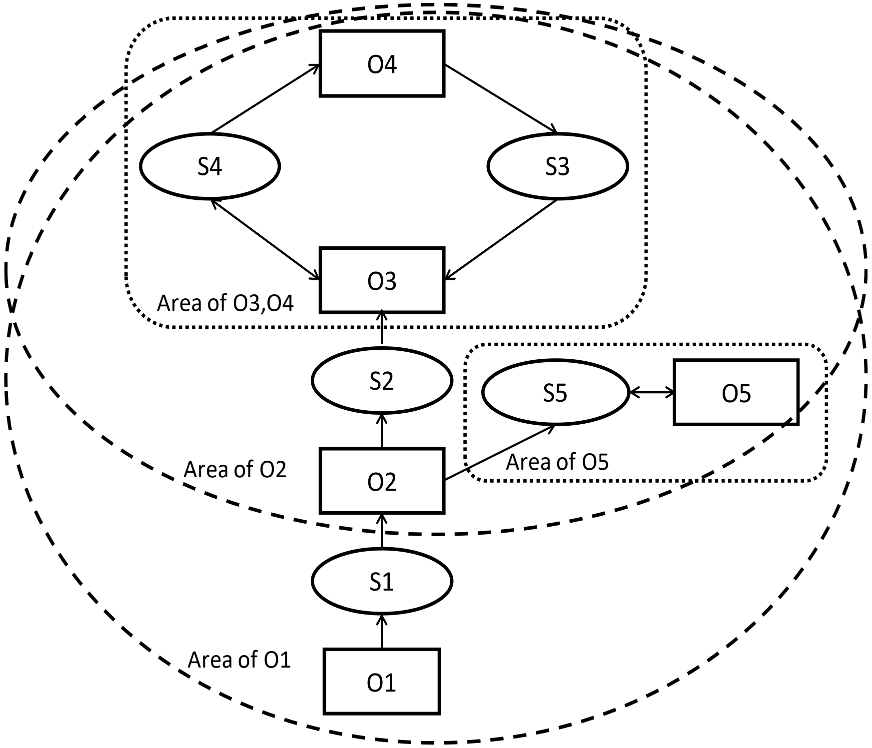
- **All** security data flow models in the literature are based on the lattice model
- But: The lattice model defines security properties in terms of itself!
  - Essentially: “A data flow is secure iff data can only move up in a pre-defined lattice of security classes”
- Also: it may make it necessary to include inexistent or impossible entities in order to have a lattice structure, e.g.
  - an entity that can know everything and
  - another that can know nothing
  - entities that contradict security constraints
    - E.g. if no entity is supposed to know both Bank1 and Bank2 data, it is still necessary to assume the existence of an entity that knows both!

Last year's FPS paper (Logrippo)

- Secrecy property for data item O is defined as a partitioning of a set of entities in at least two areas:
  - Area where the entities can know O
  - Area where the entities cannot know O
  - With an order relationship CanFlow such that CanFlow(X,Y) is true iff entity Y can know all the data items that entity X can
    - An order relationship generated by an inclusion property
    - It is a quasi-order
    - It can be represented as a digraph
  - Quasi-orders become partial orders if fully connected components are condensed into one component
    - Entities that can freely exchange data are condensed into one entity

Example:

not a lattice, only a quasi-order that can be transformed into a partial order of components

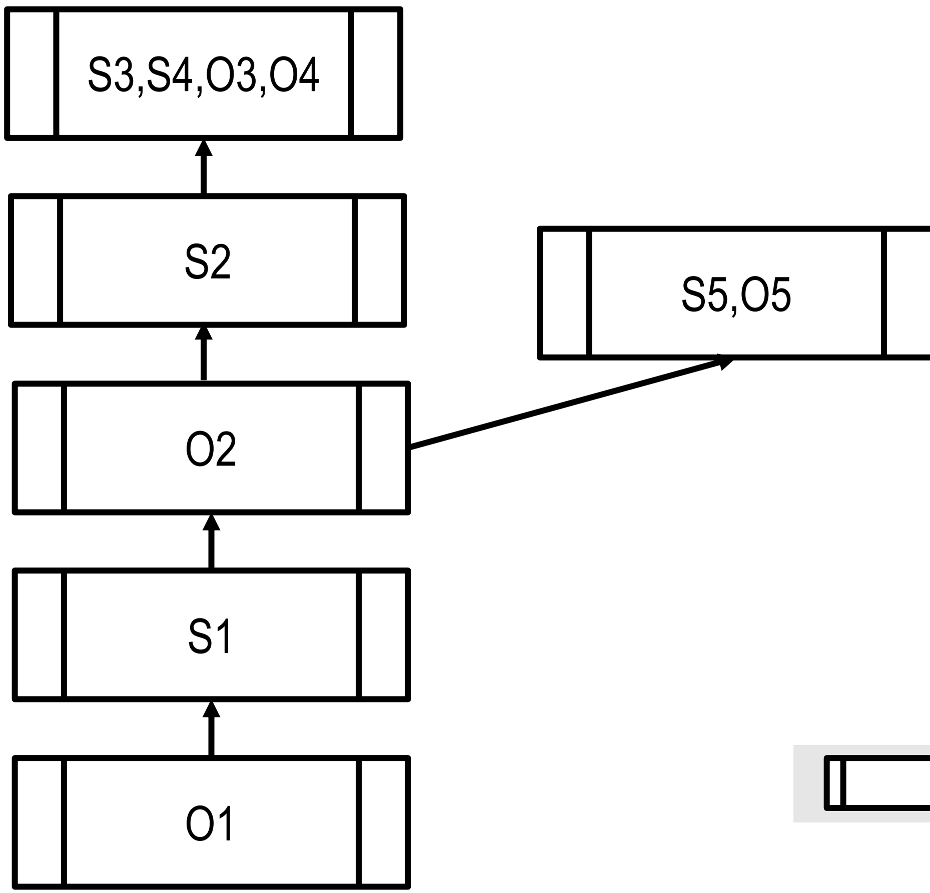


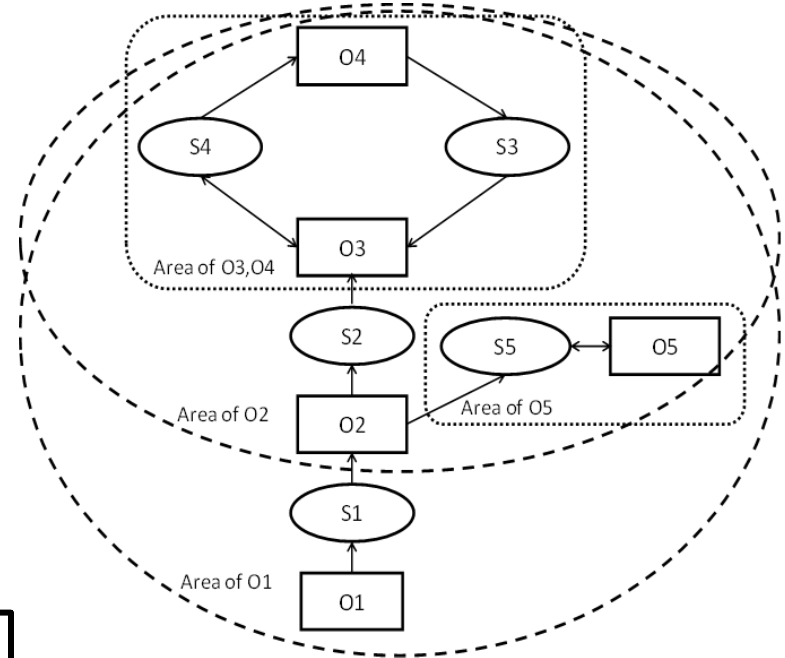
○Subjects, □Objects, → data flow

- Data in O5 cannot end up in O2, S4 etc,
  - The contents of O5 is a *secret* for O2, S4 etc.
  - Secret known only to O5 and S5
- Data in object O1 are the least secret
- Data in objects O3, O4, O5 are the most secret
  - Components are levels of security

The partial order

Collapsing contents-equivalent entities into single nodes





Equivalence class

Basic theoretical results

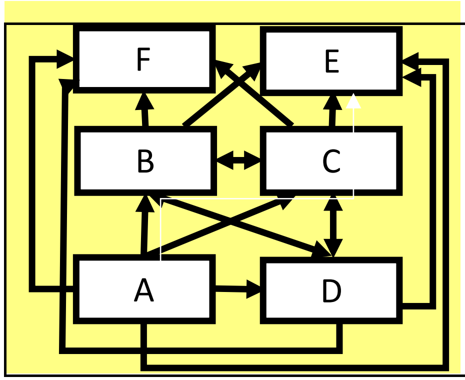
- Graph theory, order theory, relation theory say:
  - Any transitive, reflexive relation can generate a partial order of components
    - Components encapsulate equivalence classes generated by symmetries
- Algorithm theory says:
  - These partial orders can be found in linear time
    - Tarjan, Kosaraju algorithms

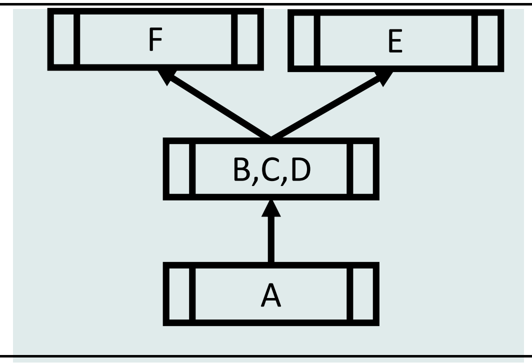
Our conclusions (references at the end)

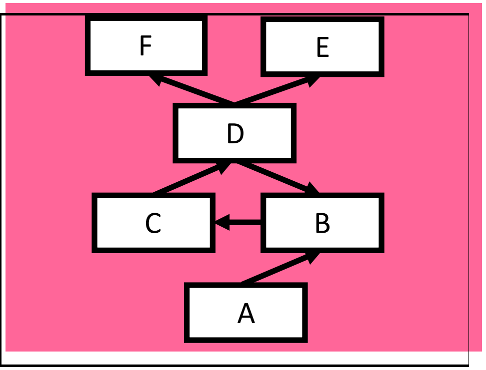
- Any data network has levels of security
  - From top secret to public (if we want to call them so)
    - Possibly only one level if there is no secrecy whatsoever (only one component)
      - (our FPS2017 paper)
- Lattice model is a sufficient model for secrecy, *partial order model is necessary and sufficient*
  - And partial orders always exist!
- Partial orders, i.e. levels of security can be found efficiently
    - (our IPL paper)
- Given security constraints, secure data networks can be constructed efficiently
    - (this paper)

Streamlining the results:

detecting components and their partial order







A data flow graph that could be obtained using a data inclusion relationship

Identifying the partial order of components (linear time algo.)

An equivalent, streamlined graph

Dynamic configuration of networks

- Entities can be added according to needs
- When a new entity is added, it must come with labels stating what data it can contain
  - CanHold or CH
- Communication channels are created according to inclusion relationships
  - CanFlow(X,Y) iff CanHold(X) $\subseteq$ CanHold(Y)
- The (efficient) partial order detection algorithm is run to clean the graph and leave only the necessary channels

Application to the Internet of Things

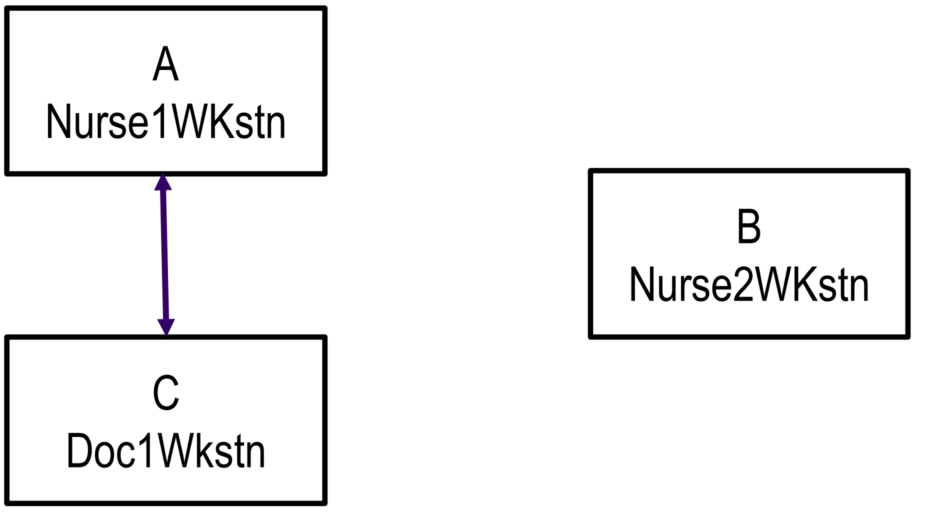
- IoT is a highly distributed, highly dynamic environment where data can flow among “things” in complex data flow configurations
- It is important that “**all and only**” secure data flows be allowed;
  - Available to all intended destinations
  - But only to those
  - As permissive as possible, but also as forbidding as necessary
- Very few **generic** solutions have been proposed for data flow security in the IoT
  - But here is one …

Hospital devices example (1)

- New(A) = Nurse1Wkstn{**SamPress, BobPulse, Stats1**}.
- New(B) = Nurse2Wkstn{SallyPulse,Stats2}.
  - No relation between CH(A) and CH(B)
  - No flow relationships

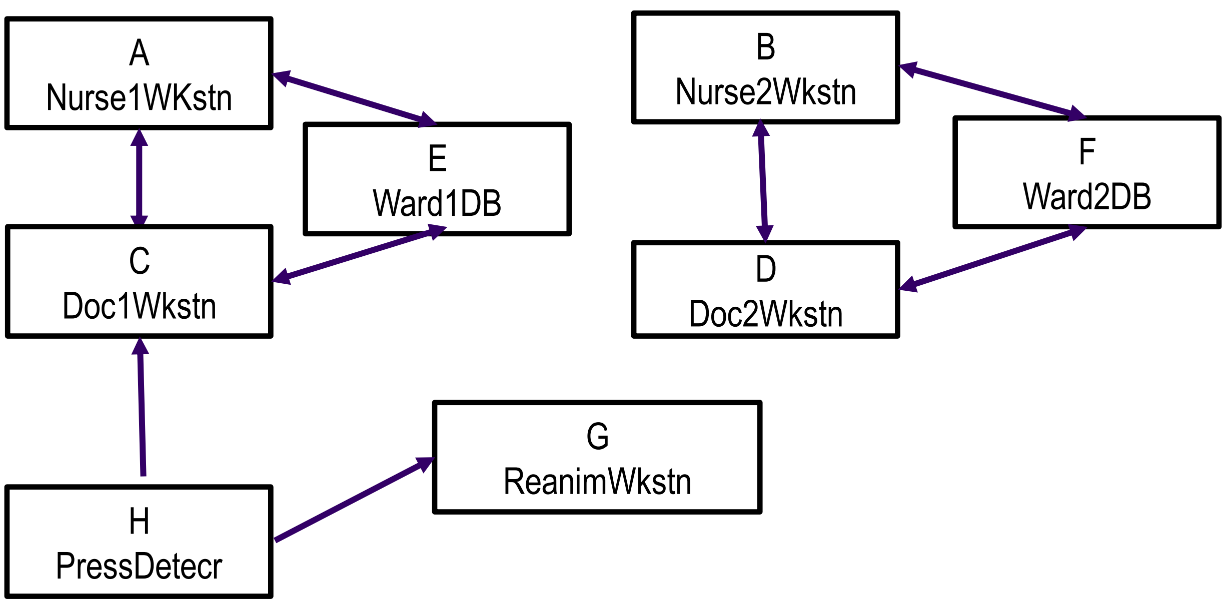
Hospital devices example (2)

- New(A) = Nurse1Wkstn{**SamPress, BobPulse, Stats1**}.
- New(B) = Nurse2Wkstn{SallyPulse,Stats2}.
- New(C) = Doc1Wkstn{**SamPress, BobPulse, Stats1**}.
  - Now, CH(C) = CH(A) so CF(C,A) and CF(A,C)

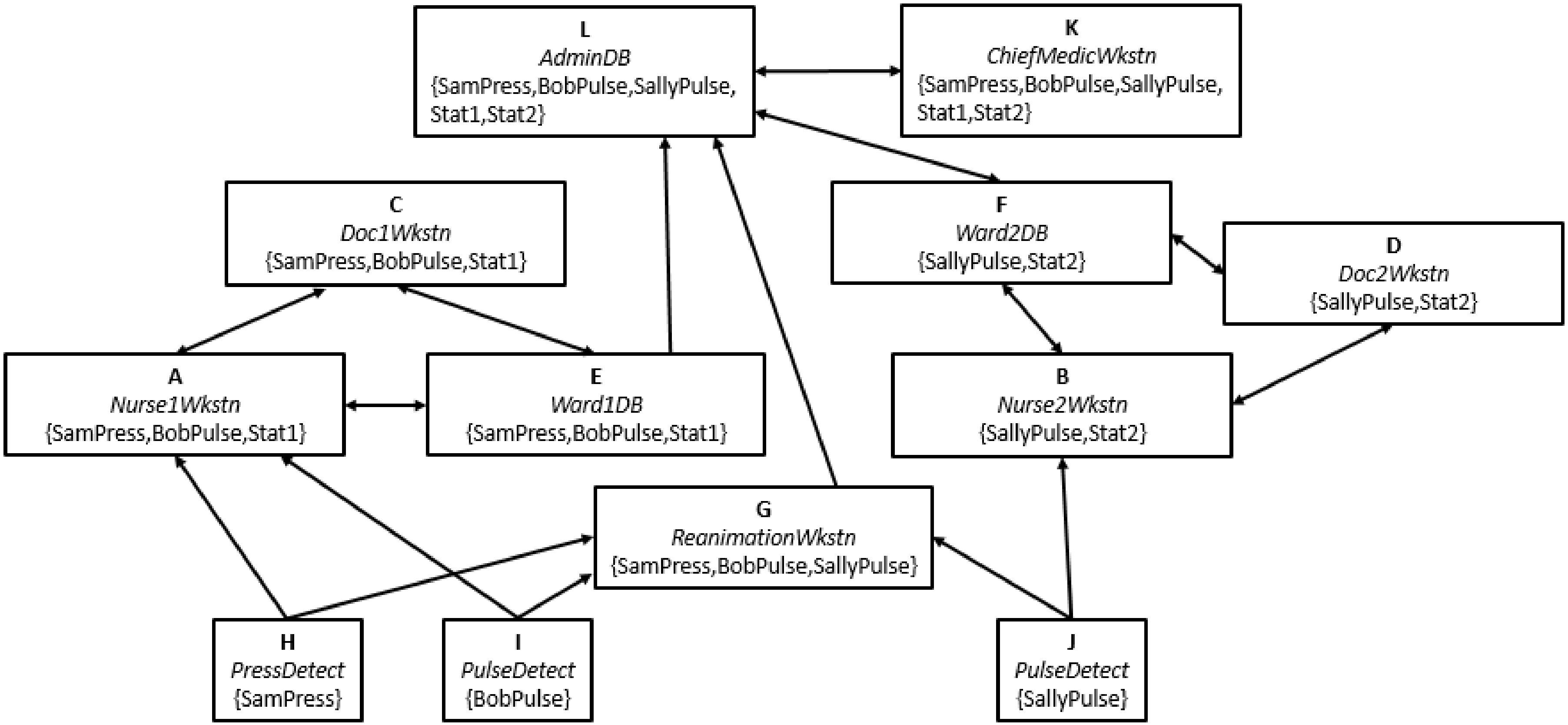


Hospital devices example (3)

- New(D) = Doc2Wkstn{SallyPulse,Stats2}.
- New(E) = Ward1DB{SamPress, BobPulse, Stats1}.
- New(F) = Ward2DB{SallyPulse,Stats2}.
- New(G) = ReanimationWkstn{SamPress, BobPulse, SallyPulse}.
- New(H) = PressDetect{SamPress)

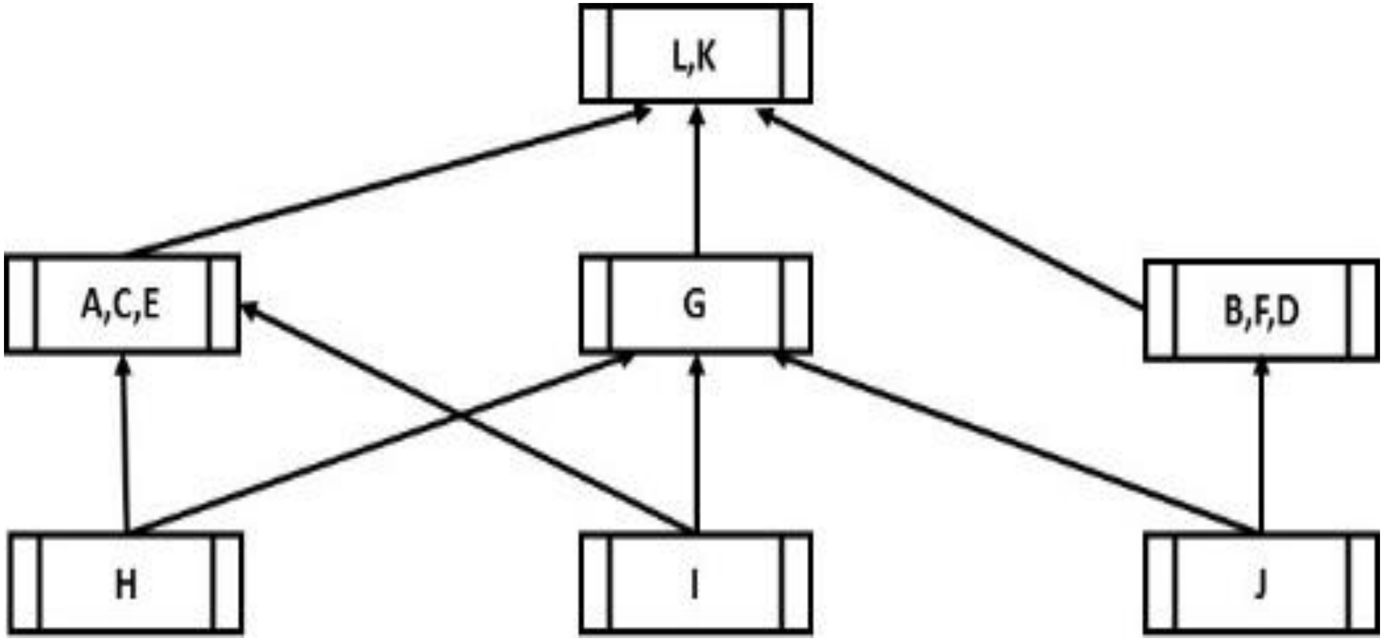


The full example in the paper



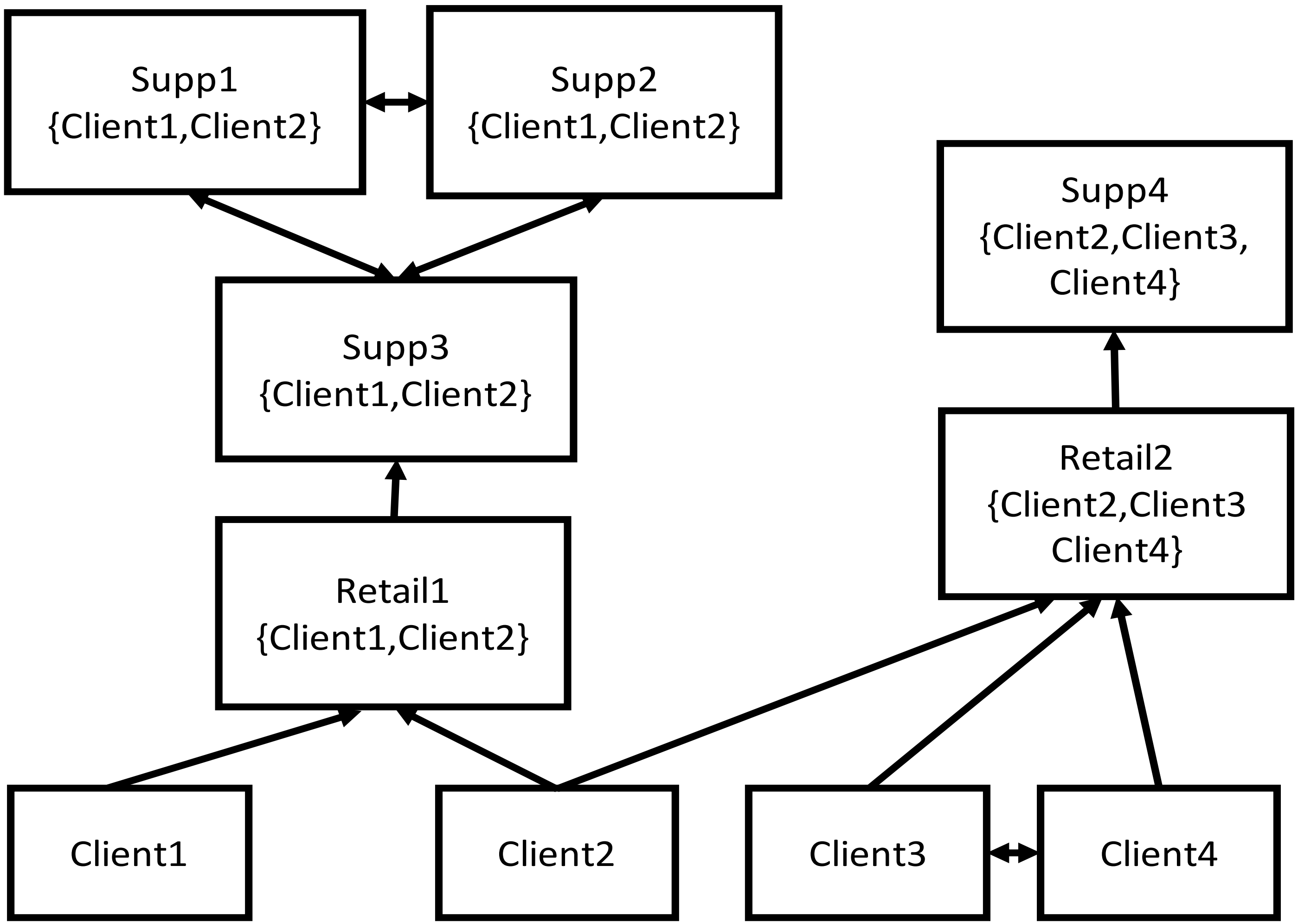
We will get to this structure independently of the order of creation of the entities

Its partial order of components

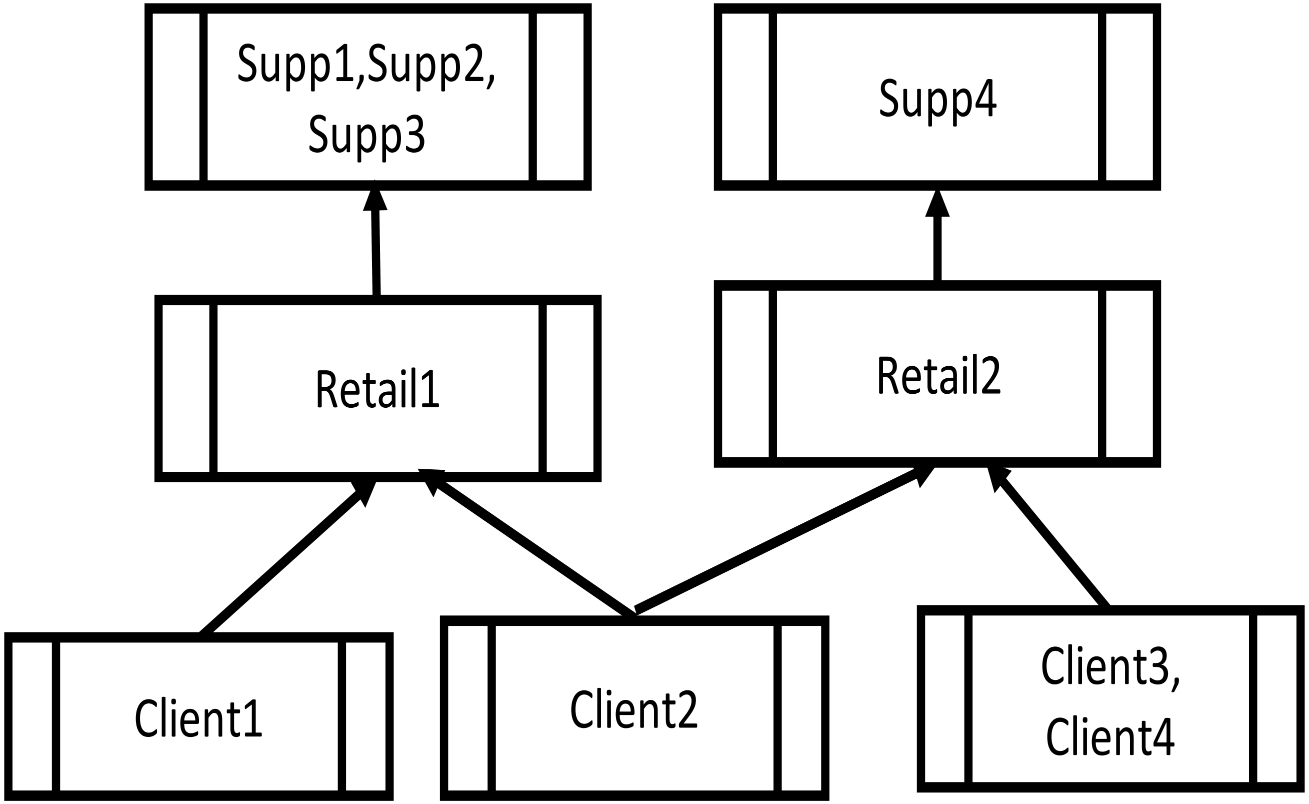


Secrecy grows together with knowledge as we move up

E-commerce example: orders data flow

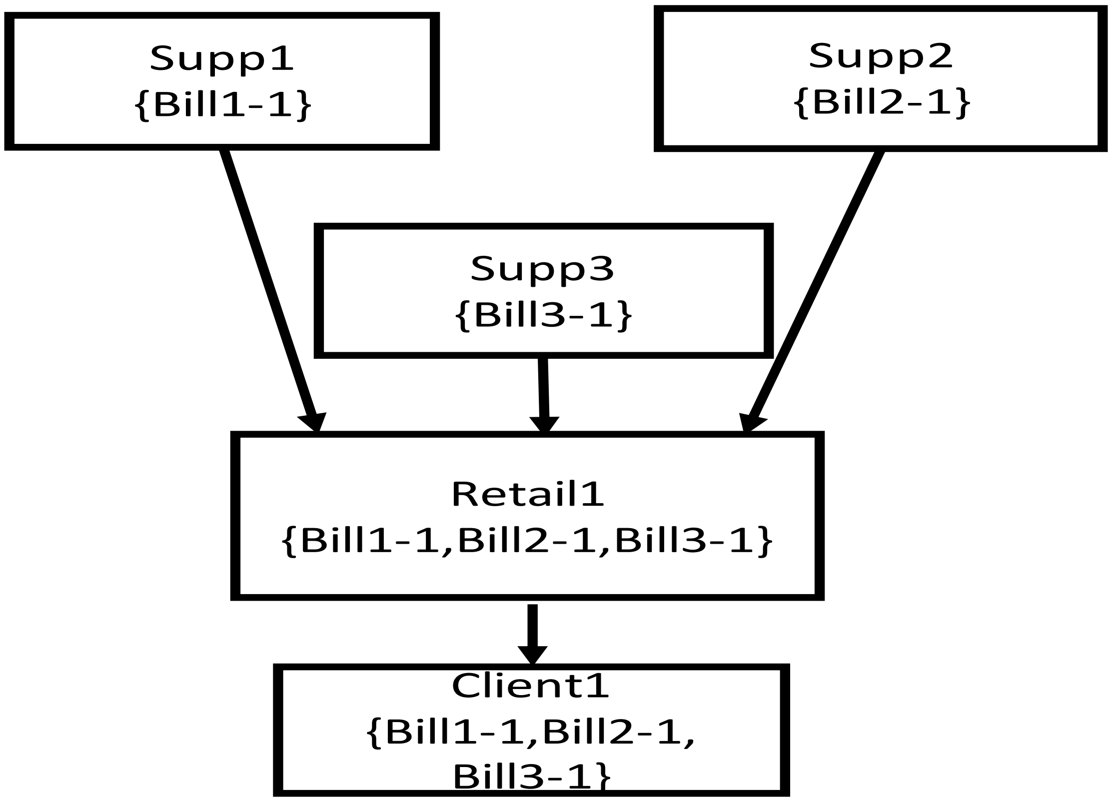


Partial order of components in e-commerce example



Another type of data flow in e-commerce

billing data flow for Client1



Coexisting data flows

- So, several data flows can coexist in a network
- Our method can handle them, by tagging data according to the data flow to which they belong

Language primitives for hospital IoT network definition

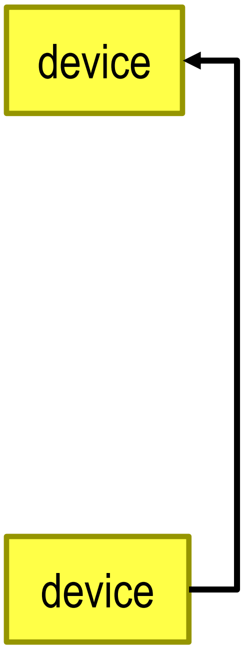
- For hospital examples, we could have the following types:
  - LType Patient(PatientId) (to define logical type Patient)
  - TType PressDetect(DetectId) (to define a device PressDetect with a DetectId)
  - TType PulseDetect(DetectId) (to define a device PulseDetect)
  - LType Ward(WardId) (to define a logical type Ward)
  - LType Nurse(NurseId)
  - TType NurseWkstn(WkstnId)
- and the following operators:
  - Assign (DetectId,Patientid) (assign a detector to a patient)
  - Assign (PatientId, WardId) (assign a patient to a ward)
  - Assign (NurseId,WardId) (assign a nurse to a ward)
  - Assign (WkstnId,WardId) (assign a workstation to a ward)
  - Etc.

And CanHold assertions, such as

- ***CH****(WkstnId,DetectId)*
  - ***if*** *Assign(PatientId,WardId(WkstnId))* ***and*** *Assign(DetectId,PatientId)*
  - If a workstation is assigned to a ward and a patient is also assigned to the same ward,
  - then there is a data flow (channel) between the workstation and the detectors for the patient

Progressive network construction:

- New Ward (Emerg).
- New NurseWkstn(EmergWkstn)
- New Nurse (Alice)
- Assign (Alice,Emerg).
- Assign (EmergWkstn,Emerg)
- New Patient (Sam).
- Assign (Sam,Emerg)
- New PressDetect(PRD0001)
- Assign (PRD0001, Sam)



- At this point, by the CH assertion, a channel is created from device PRD0001 to previously created device EmergWkstn
  - The emergency workstation has been assigned to the emergency ward and then Sam has been assigned to device PRD0001 and the same ward

Re-configurations

- IoT systems should be able to continuously reconfigure
  - Data de-classification and other updates due to changing requirements
  - Entity disappearance
- It might be possible to repair the network locally, or in the worst it might be necessary to execute our generic configuration algorithm

How to implement this?

- By access control mechanisms
- By routing mechanisms
- By encryption, to implement secure channels
  - If data flow from A to C through B, but B cannot read it, then the channel is only from A to C

RPL routing for IoT networks

- RPL: Routing Protocol for Low-Power and Lossy Networks
- Uses Directed Acyclic Graphs (DAGs) to express routing in IoT systems
- New devices are placed on DAGs according to Objective Functions (OFs)
- In current use, OFs express mainly efficiency constraints: minimum power usage
- Can our own DAGs be combined with RPL DAGs to include security constraints in RPL routing?
  - If so, it could be possible to program RPL routing to avoid certain nodes if these should not be part of the flow
  - Permissible data paths should run over existing links by using encryption?
    - Research topic …

Conclusions

- **Necessary and sufficient** conditions for data secrecy in networks can be obtained by generalizing traditional MAC and lattice concepts
- **Exactness:** By labeling entities in IoT networks according to the type of data they can hold, it is possible to configure data transfer channels such that **all and only** logically allowed flows are possible
  - Both **secrecy and integrity** are taken care of
- **Scalability:** Efficient algorithms exist for such configurations, which makes the solution **scalable** and practical
- **Implementability:**
  - Data must be tagged, entities must be labelled
  - Further research on protocols and encryption is necessary

Related work

- Although the literature in security and access control in the IoT is vast, there are very few papers with solutions for data flow control in IoT networks
- Previous to us, they were all based on the lattice model
- Many papers on security in IoT do not provide specific solutions

Some basic references (as of 2018)

- S. Khobragade, N. V. Narendra Kumar, R. K. Shyamasundar. Secure synthesis of IoT via readers-writers flow model. Proc. Intern. Conf. on Distrib. Computing and Internet Techn. (ICDCIT 2018), LNCS 10722, 86–104.
- T. Pasquier, J. Bacon, J. Singh, D. Eyers. Data-Centric Access Control for Cloud Computing. Proc. 21st ACM Symp. on Access Control Models and Technologies (SACMAT '16), 81-88. (+ other papers by same authors)

- Search ‘Luigi Logrippo papers’ and ‘Luigi Logrippo presentations’ and look at the most recent titles