Secure Programming via Libraries

Disjunction Category Labels

Alejandro Russo (russo@chalmers.se)

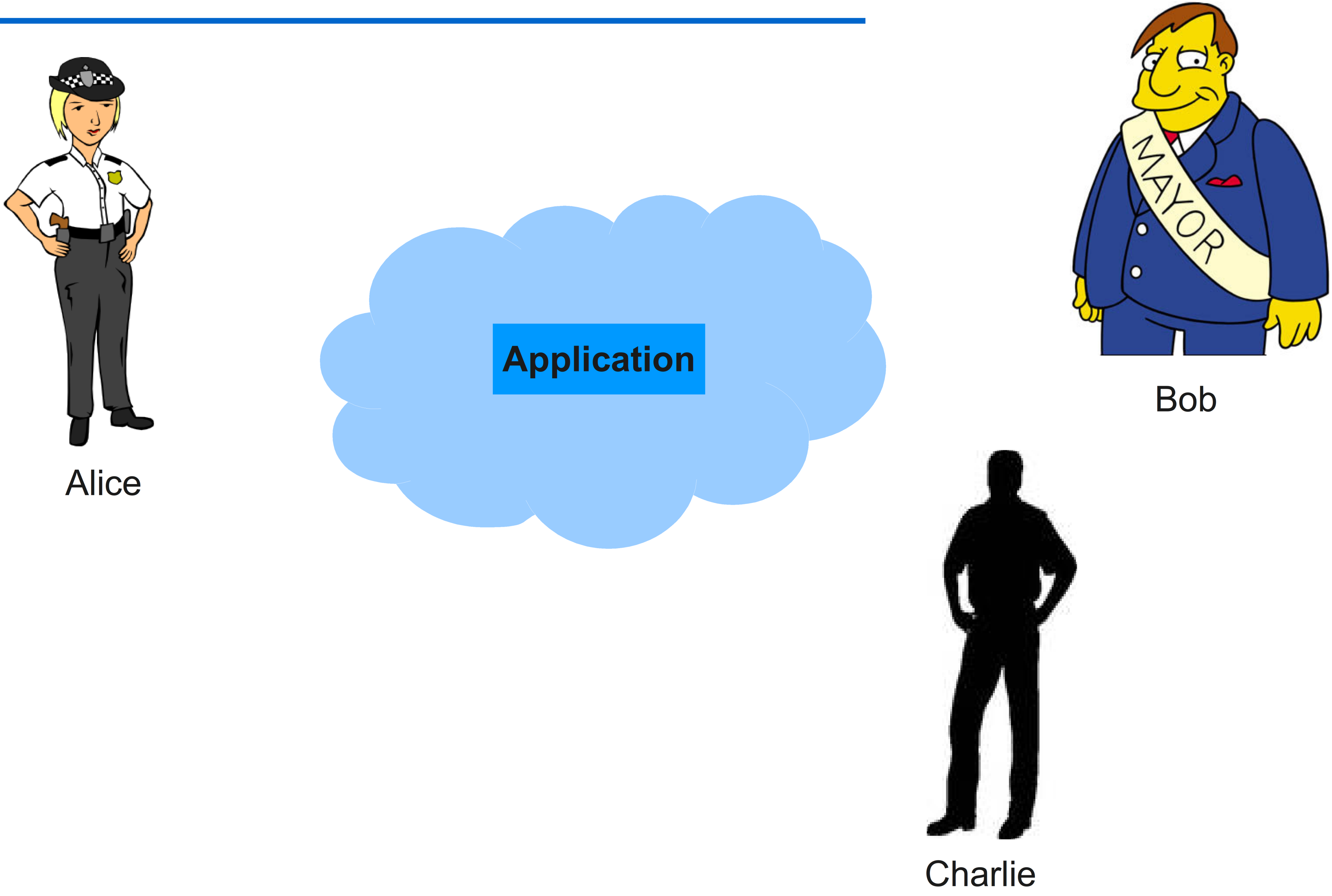
Escuela de Ciencias Informáticas (ECI) 2011
UBA, Buenos Aires, Argentina

CHALMERS

Motivation

- It is usually common to consider the simple two-point lattice to represent confidential and public information
  - Information flows from public to secret
- In scenarios of **mutual distrust**, things are a little bit more complicated
- Let us see a concrete scenario

Motivaton



Alice

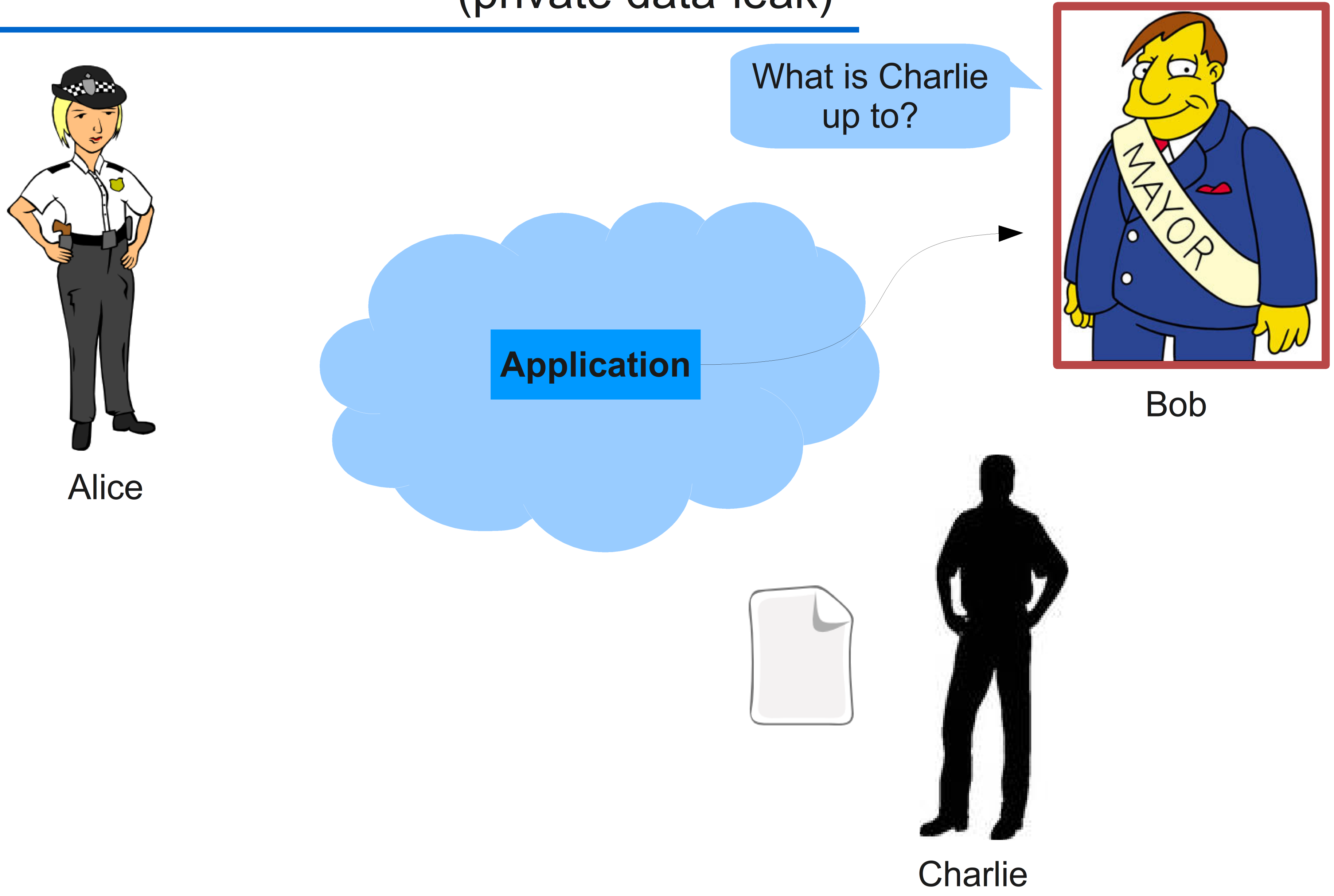
Application

Bob

Charlie

Motivaton: Confidentiality

(private data-leak)



What is Charlie up to?

Application

Alice

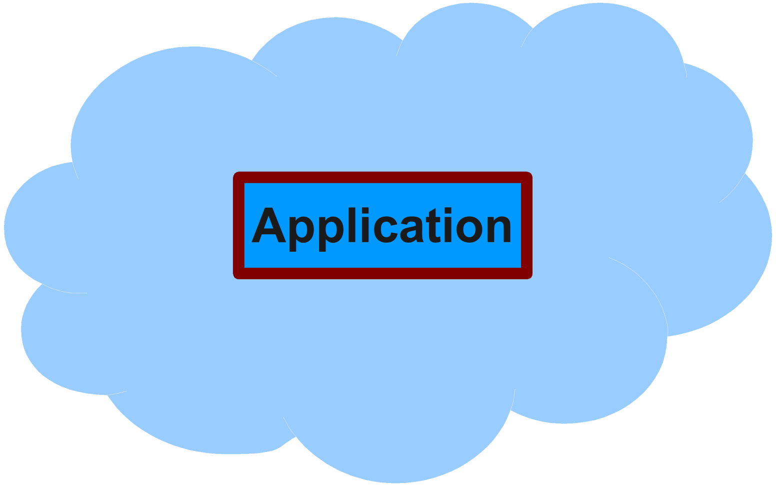
Bob

Charlie

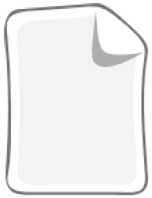
Motivaton: Confidentiality

(private data-leak)

Alice



Bob

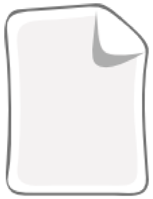


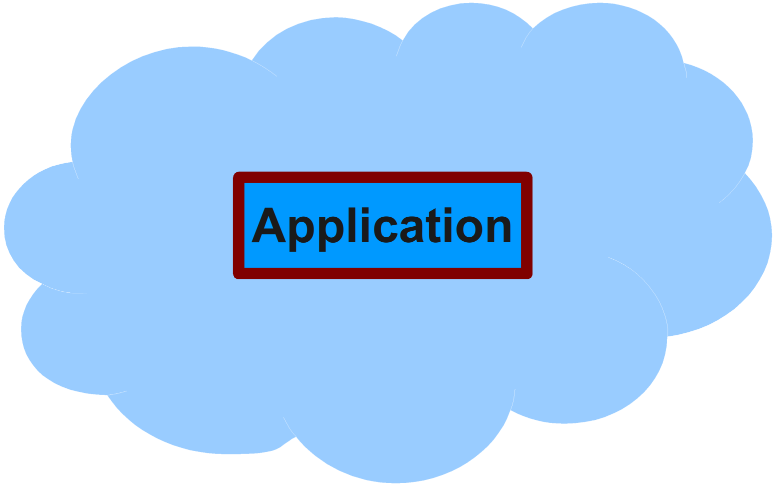
Charlie

Charlie is the owner of that information, therefore he **decides** where it goes. The system should respect that decision.

Motivaton: Confidentiality

(private data-leak)





Alice and Bob **collaborate in creating aggregated data**

The system must not send that data to Charlie unless Alice and Bob **agree** on that!

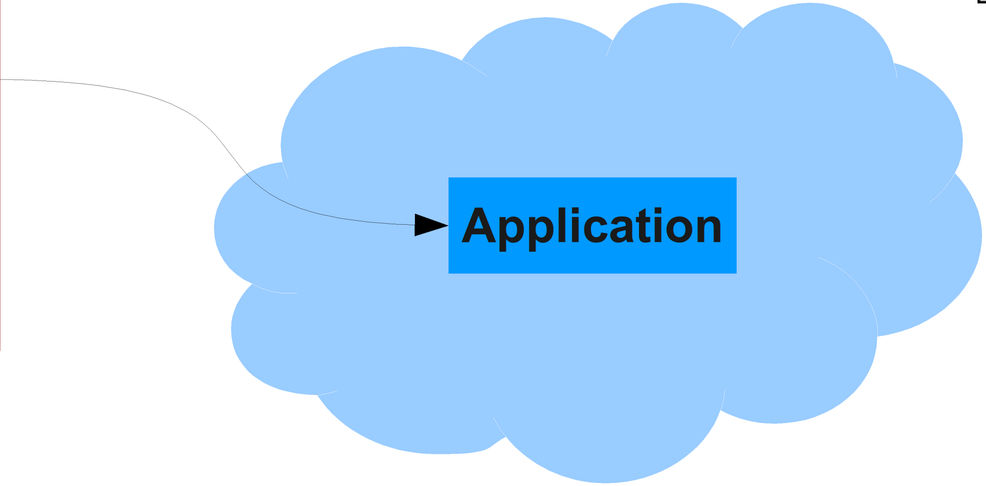
Motivaton: Integrity

(user-forged write)

What is this?

Let's involve the mayor in something illegal.

Alice



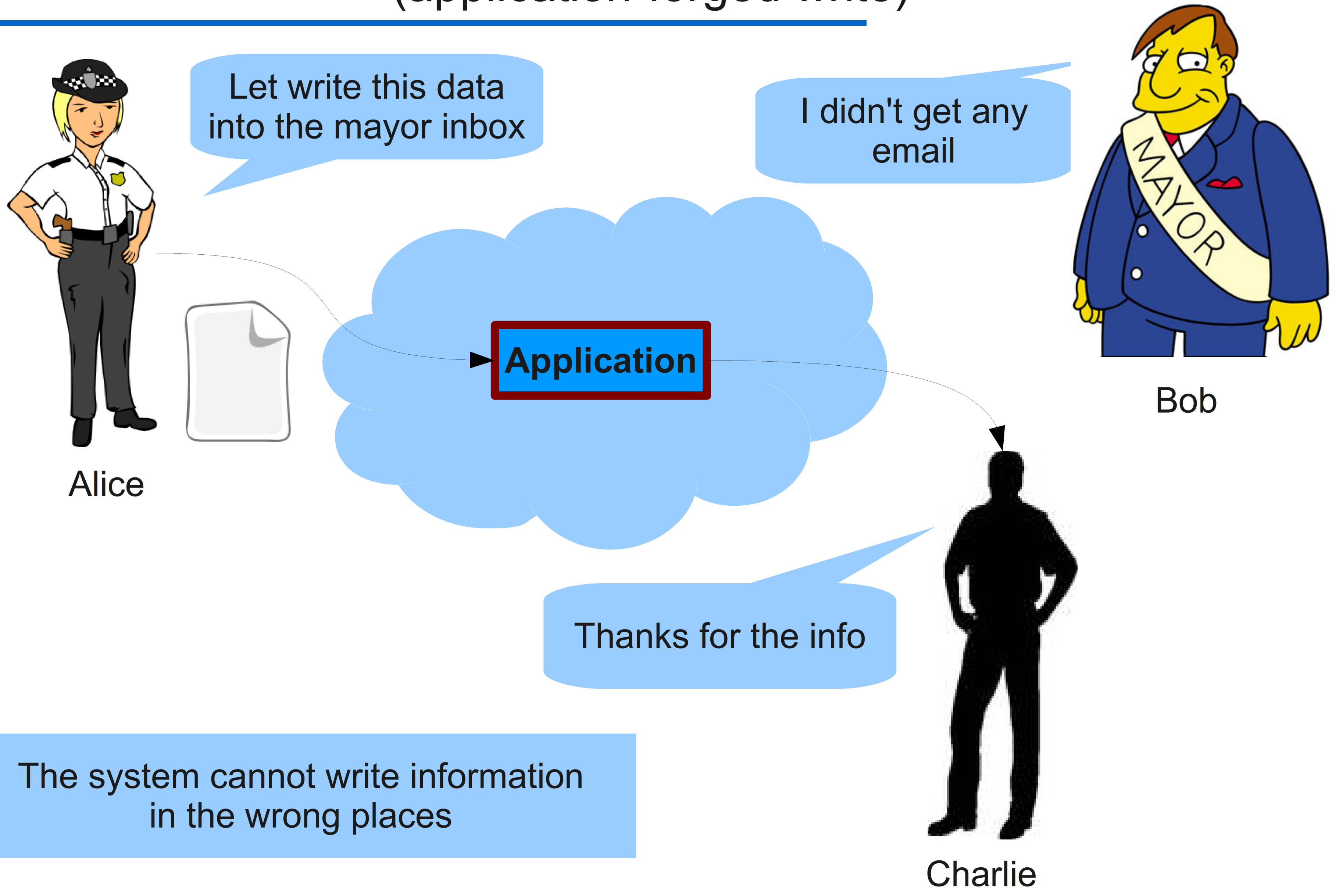
Bob

Charlie

Bob should be the only one **modifying** its own information (unless indicated otherwise)

Motivaton: Integrity

(application-forged write)



The system cannot write information in the wrong places

Disjunction Category Labels

[Stefan, Russo, Mazieres] **(work-in-progress)**

- For short: **DCLabels**
- It is a label system to express **restrictions** on data which allows to **reflect the concern of multiple parties**
- Principal
  - Source or authority (e.g., Alice, Bob, and Charly)
- Disjunction Category (just category)
  - Set of principals
  - Each principal is said to **own** the category
- Categories are associated to data

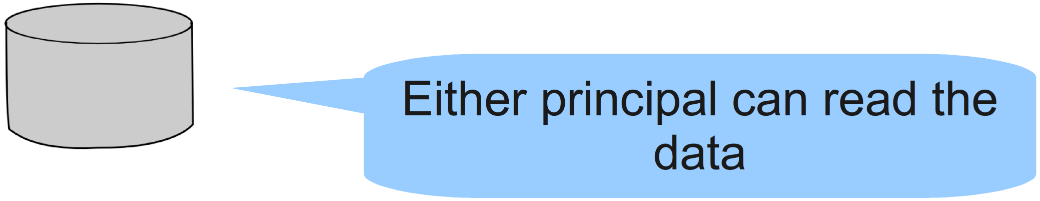
Disjunction Category

- Set of principals
  - $\{\mathrm{Alice}, \mathrm{Bob}\}$
- We write it as a disjunction
  - $[\mathrm{Alice} \vee \mathrm{Bob}]$
- What is the meaning?
  - They are restrictions
  - It depends if we are considering ***confidentiality*** or ***integrity***

Disjunction Category

- Confidentiality

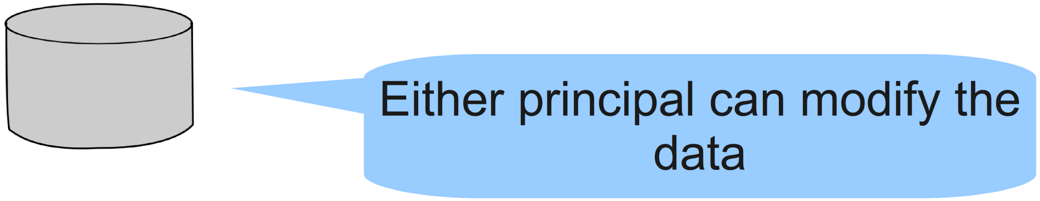
$$[\text{Alice} \vee \text{Bob}]$$



Either principal can read the data

- Integrity

$$[\text{Alice} \vee \text{Bob}]$$



Either principal can modify the data

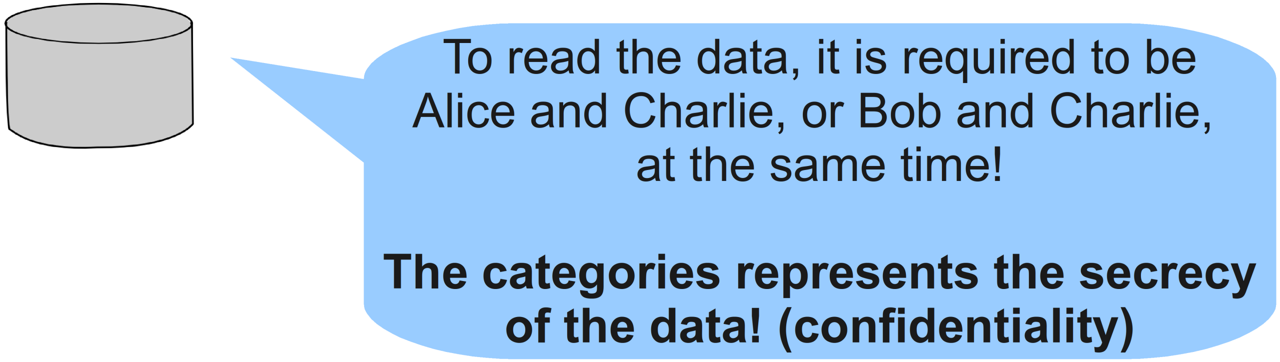
Set of Disjunction Categories

- Data can be associated with several categories
  - It represents data with different restrictions (perhaps imposed by different parties in the system)
  - $\{\{\mathrm{Alice}, \mathrm{Bob}\}, \{\mathrm{Charlie}\}\}$
- We write it as a conjunction
  - $[\mathrm{Alice} \vee \mathrm{Bob}] \wedge [\mathrm{Charlie}]$
- What is the meaning?
  - It depends if we are considering ***confidentiality*** or ***integrity***

Conjunctions of Disjunctions

- Confidentiality

$$[\text{Alice} \vee \text{Bob}] \wedge [\text{Charlie}]$$

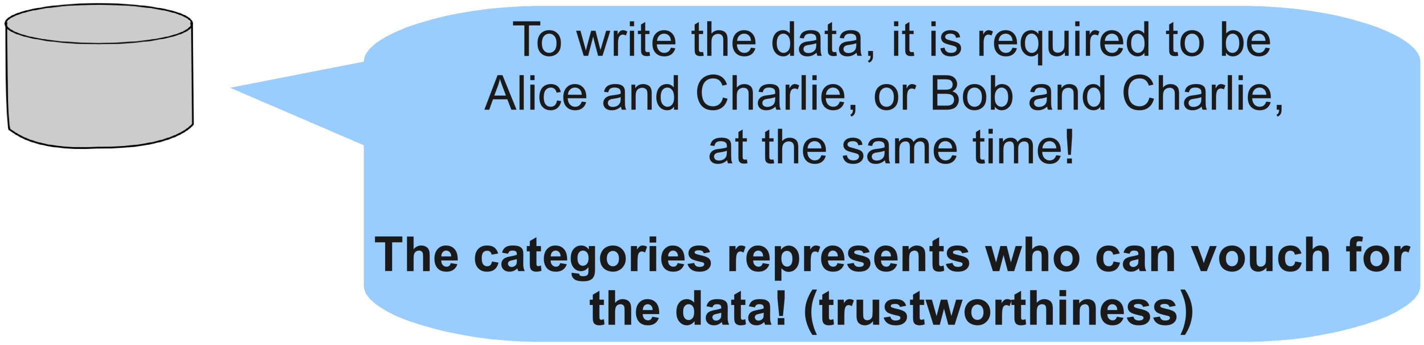


To read the data, it is required to be Alice and Charlie, or Bob and Charlie, at the same time!

The categories represents the secrecy of the data! (confidentiality)

- Integrity

$$[\text{Alice} \vee \text{Bob}] \wedge [\text{Charlie}]$$



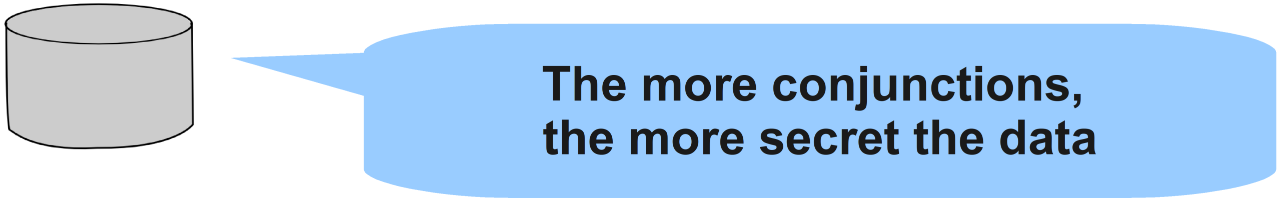
To write the data, it is required to be Alice and Charlie, or Bob and Charlie, at the same time!

The categories represents who can vouch for the data! (trustworthiness)

Conjunctions of Disjunctions

- Confidentiality

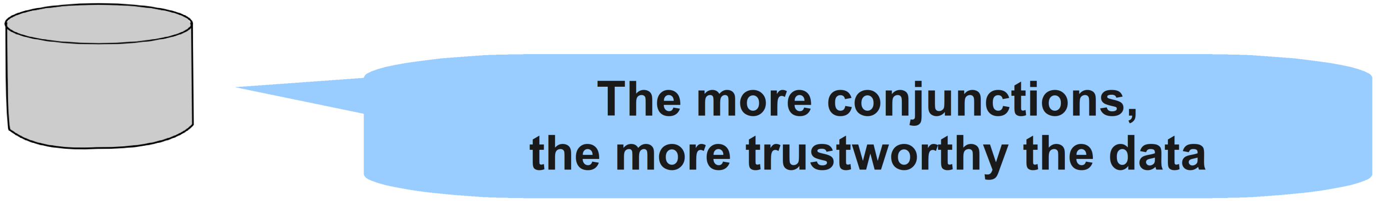
$$[\mathrm{Alice} \vee \mathrm{Bob}] \wedge [\mathrm{Charlie}] \wedge \cdots \wedge \cdots \wedge \ldots$$



The more conjunctions, the more secret the data

- Integrity

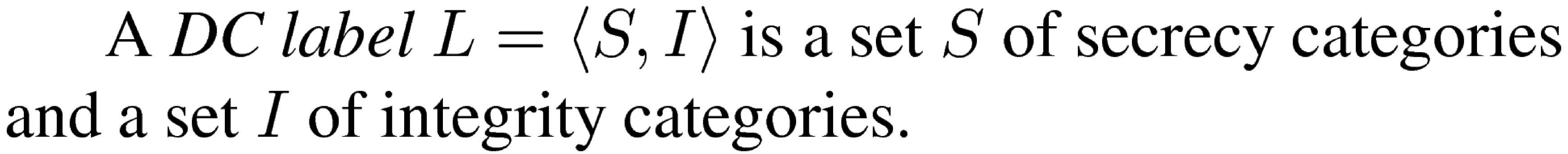
$$[\mathrm{Alice} \vee \mathrm{Bob}] \wedge [\mathrm{Charlie}] \wedge \cdots \wedge \cdots \wedge \ldots$$



The more conjunctions, the more trustworthy the data

DCLabels

- What is a DCLabel?



A *DC label* $L = \langle S, I \rangle$ is a set S of secrecy categories and a set I of integrity categories.

- The secrecy categories restrict who can read, receive, or propagate information
- The integrity categories restrict who can modify the data

Information-flow

- Information can flow if all categories are respected
- Confidentiality

$$\langle [Alice \vee Bob], [] \rangle \quad \bullet\!\!\not\longrightarrow\!\!\bullet \quad \langle [Alice \vee Bob \vee Charlie], [] \rangle$$

$$\langle [Alice \vee Bob], [] \rangle \quad \bullet\!\!\longrightarrow\!\!\bullet \quad \langle [Alice], [] \rangle$$

$$\langle [Alice \vee Bob], [] \rangle \quad \bullet\!\!\not\longrightarrow\!\!\bullet \quad \langle [Charlie] \wedge [Deian], [] \rangle$$

$$\langle [Alice] \wedge [Bob], [] \rangle \quad \bullet\!\!\not\longrightarrow\!\!\bullet \quad \langle [Alice \vee Bob], [] \rangle$$

$$\langle [Alice] \wedge [Bob], [] \rangle \quad \bullet\!\!\not\longrightarrow\!\!\bullet \quad \langle [Alice], [] \rangle$$

Information-flow

- Information can flow if all categories are respected
- Integrity

$$\langle [], [Alice \vee Bob] \rangle \quad \bullet \not\longrightarrow \bullet \quad \langle [], [Alice \vee Bob \vee Charlie] \rangle$$

$$\langle [], [Alice] \rangle \quad \bullet \longrightarrow \bullet \quad \langle [], [Alice \vee Bob] \rangle$$

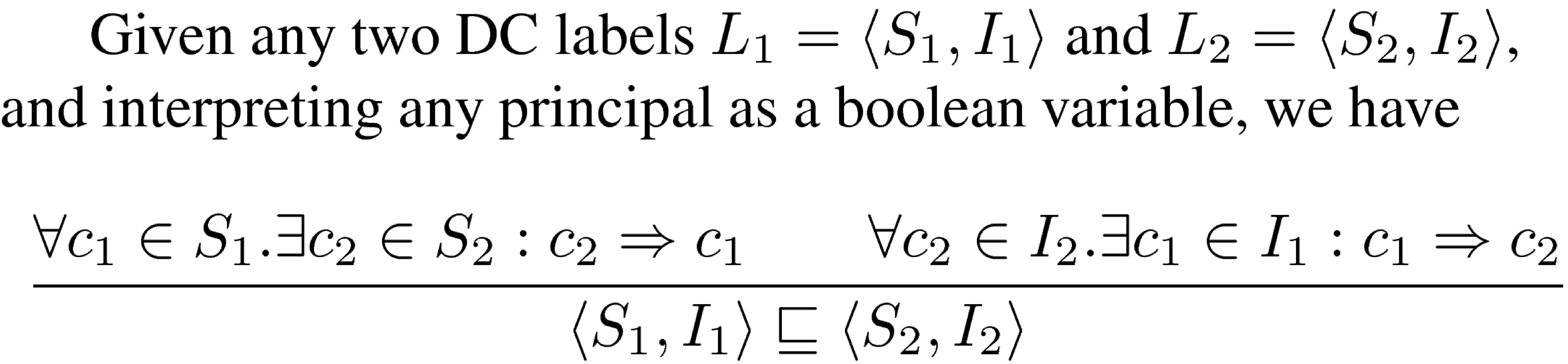
$$\langle [], [Alice \vee Bob] \rangle \quad \bullet \not\longrightarrow \bullet \quad \langle [], [Charlie] \wedge [Deian] \rangle$$

$$\langle [], [Alice] \wedge [Bob] \rangle \quad \bullet \not\longrightarrow \bullet \quad \langle [], [Alice] \rangle$$

$$\langle [], [Alice] \rangle \quad \bullet \not\longrightarrow \bullet \quad \langle [], [Alice] \wedge [Bob] \rangle$$

Partial Order Between DCLabels

- We formalize a *can-flow-to* relationship, i.e. a partial order relationship $\sqsubseteq$



Given any two DC labels $L_1 = \langle S_1, I_1 \rangle$ and $L_2 = \langle S_2, I_2 \rangle$, and interpreting any principal as a boolean variable, we have

$$\frac{\forall c_1 \in S_1 . \exists c_2 \in S_2 : c_2 \Rightarrow c_1 \qquad \forall c_2 \in I_2 . \exists c_1 \in I_1 : c_1 \Rightarrow c_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

Partial Order Between DCLabels

Given any two DC labels $L_1 = \langle S_1, I_1 \rangle$ and $L_2 = \langle S_2, I_2 \rangle$, and interpreting any principal as a boolean variable, we have

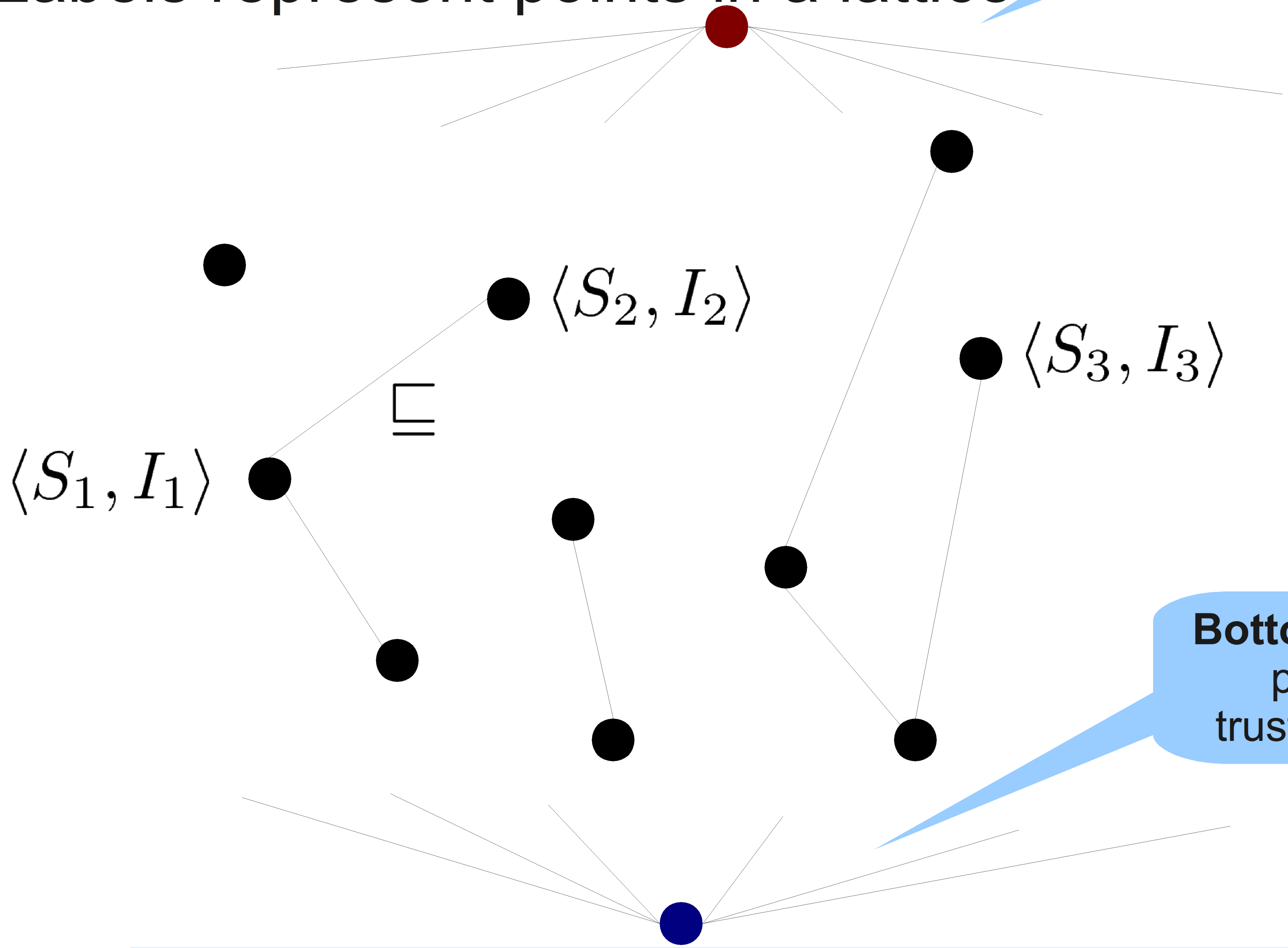
$$\frac{\forall c_1 \in S_1 . \exists c_2 \in S_2 : c_2 \Rightarrow c_1 \qquad \forall c_2 \in I_2 . \exists c_1 \in I_1 : c_1 \Rightarrow c_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

$$\Leftrightarrow$$

$$\frac{S_2 \Rightarrow S_1 \wedge I_1 \Rightarrow I_2}{\langle S_1, I_1 \rangle \sqsubseteq \langle S_2, I_2 \rangle}$$

Lattice

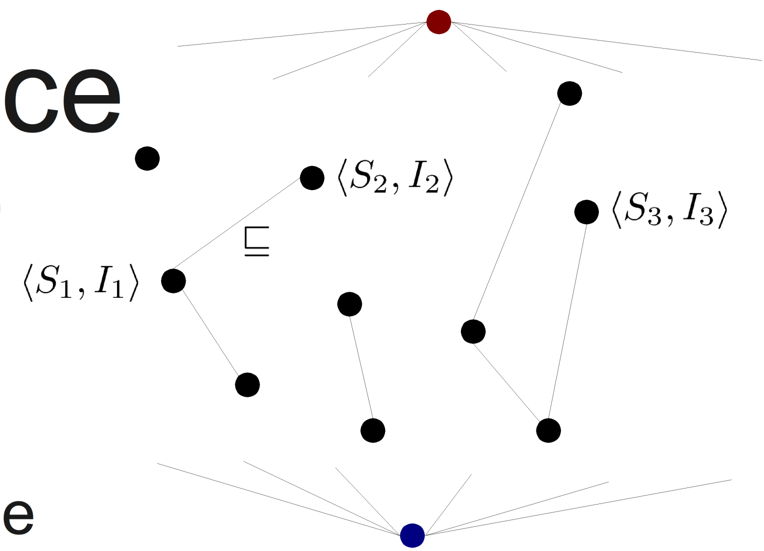
- DCLabels represent points in a lattice


$\langle S_2, I_2 \rangle$
$\sqsubseteq$
$\langle S_1, I_1 \rangle$
$\langle S_3, I_3 \rangle$

Top: the most confidential and untrustworthy data

Bottom: the most public and trustworthy data

Dynamic Lattice


$\langle S_2, I_2 \rangle$
$\langle S_3, I_3 \rangle$
$\sqsubseteq$
$\langle S_1, I_1 \rangle$

- Principals and DCLabels can be generated at run-time
  - This lattice might be modified on run-time
- In a system with several principals (e.g., users), it is difficult to fit the lattice in one picture
  - Gmail? Hotmail? Facebook?
- Top element: $\langle [P_1] \wedge [P_2] \wedge \cdots \wedge [P_n], [] \rangle$
- Bottom element: $\langle [], [P_1] \wedge [P_2] \wedge \cdots \wedge [P_n] \rangle$
- Problem?
  - We do not always know all the principals in the system
    - Principals can come and go

False

True

$\langle All, [] \rangle$

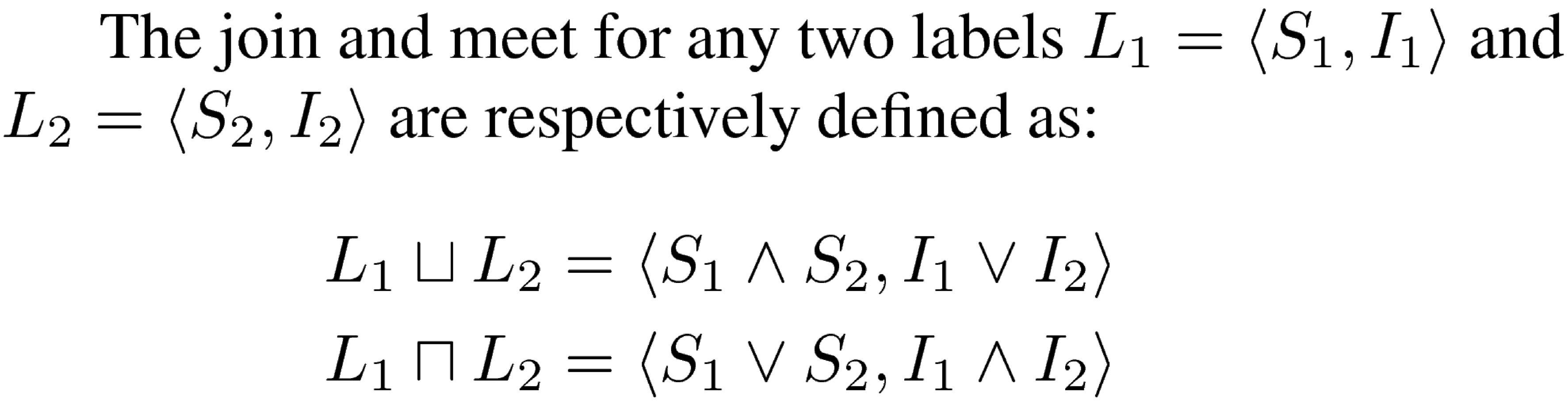
$\langle [], All \rangle$

Join and Meet Operations

- It is possible to define the join and meet operations and proof their correctness
  - The authors of DLM [Myers, Liskov 98] have not proved this formally
    - “The formula for meet is sound, but unlike the formula for join, it does not always produce the most restrictive label for all possible extensions P'”
    - “The result is that label inference must be conservative in some cases, which does not seem to be a significant problem”

Join and Meet for DCLabels

- They are simply defined as



The join and meet for any two labels $L_1 = \langle S_1, I_1 \rangle$ and $L_2 = \langle S_2, I_2 \rangle$ are respectively defined as:

$$L_1 \sqcup L_2 = \langle S_1 \wedge S_2, I_1 \vee I_2 \rangle$$
$$L_1 \sqcap L_2 = \langle S_1 \vee S_2, I_1 \wedge I_2 \rangle$$

- We proved that this is actually the join and meet (exercise)
- These operations might introduce categories which are redundant

Declassification/Endorsement

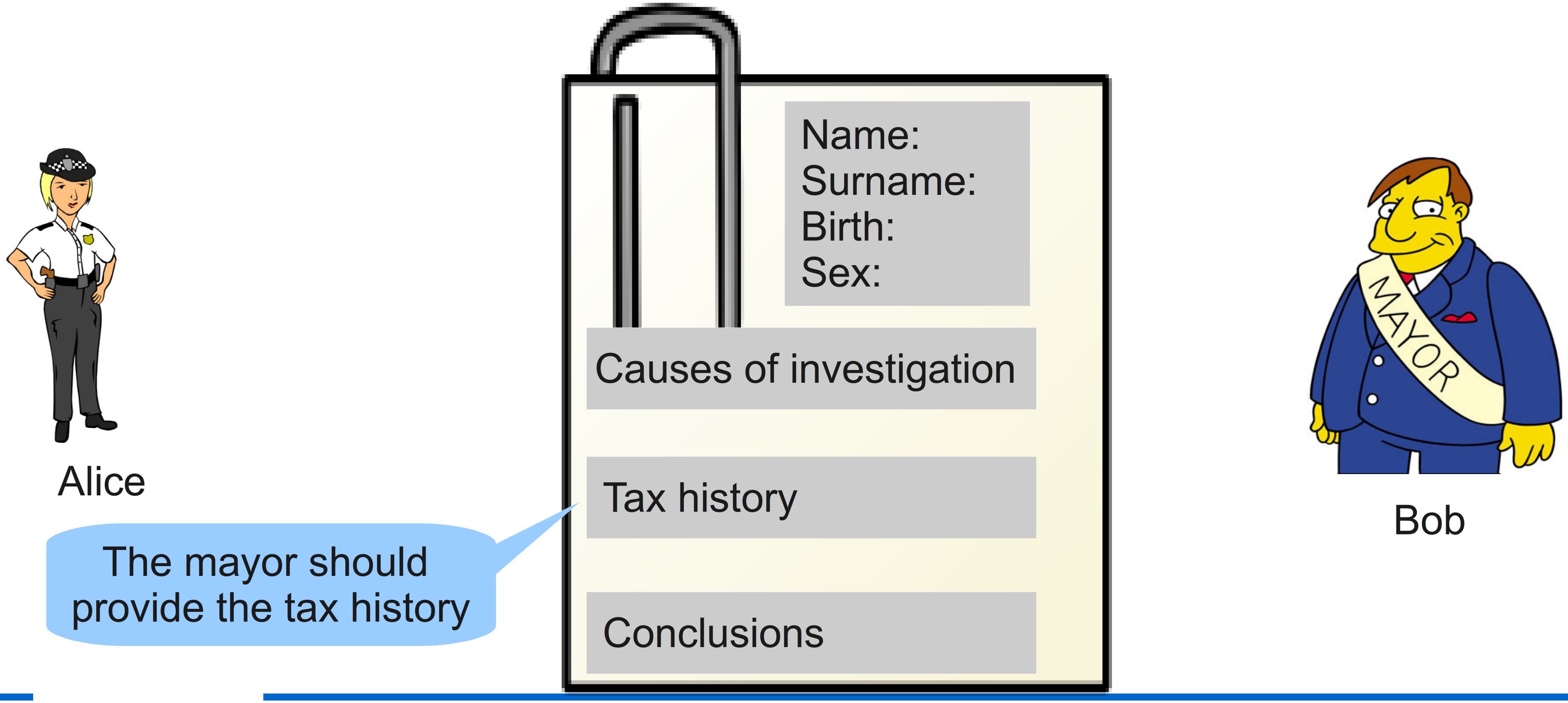
- Any system have some sort of intended release of information
- In a mutual distrust environment, it is necessary to declassify data after some collaborative effort

$$\langle [Alice] \wedge [Bob], [] \rangle \quad \bullet\!\!\not\!\longrightarrow\!\!\bullet \quad \langle [Alice], [] \rangle$$

- We describe a motivating example based on confidentiality but it also holds for integrity

Declassification

- Alice is carrying out an investigation and she needs the tax history of the suspect



Alice

Name:
Surname:
Birth:
Sex:

Causes of investigation

Tax history

Conclusions

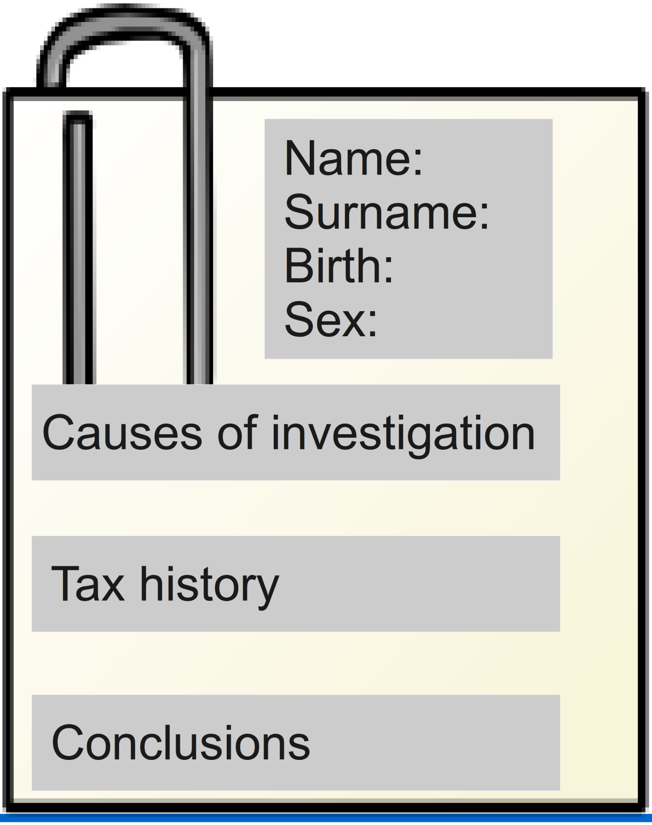
The mayor should provide the tax history

Bob

Declassification

- The code that Alice is running has the privilege “Alice”
  - It allows to ignore the principal “Alice” in the DCLabels
  - Privileges help to bypass $\sqsubseteq$

Alice



Name:
Surname:
Birth:
Sex:

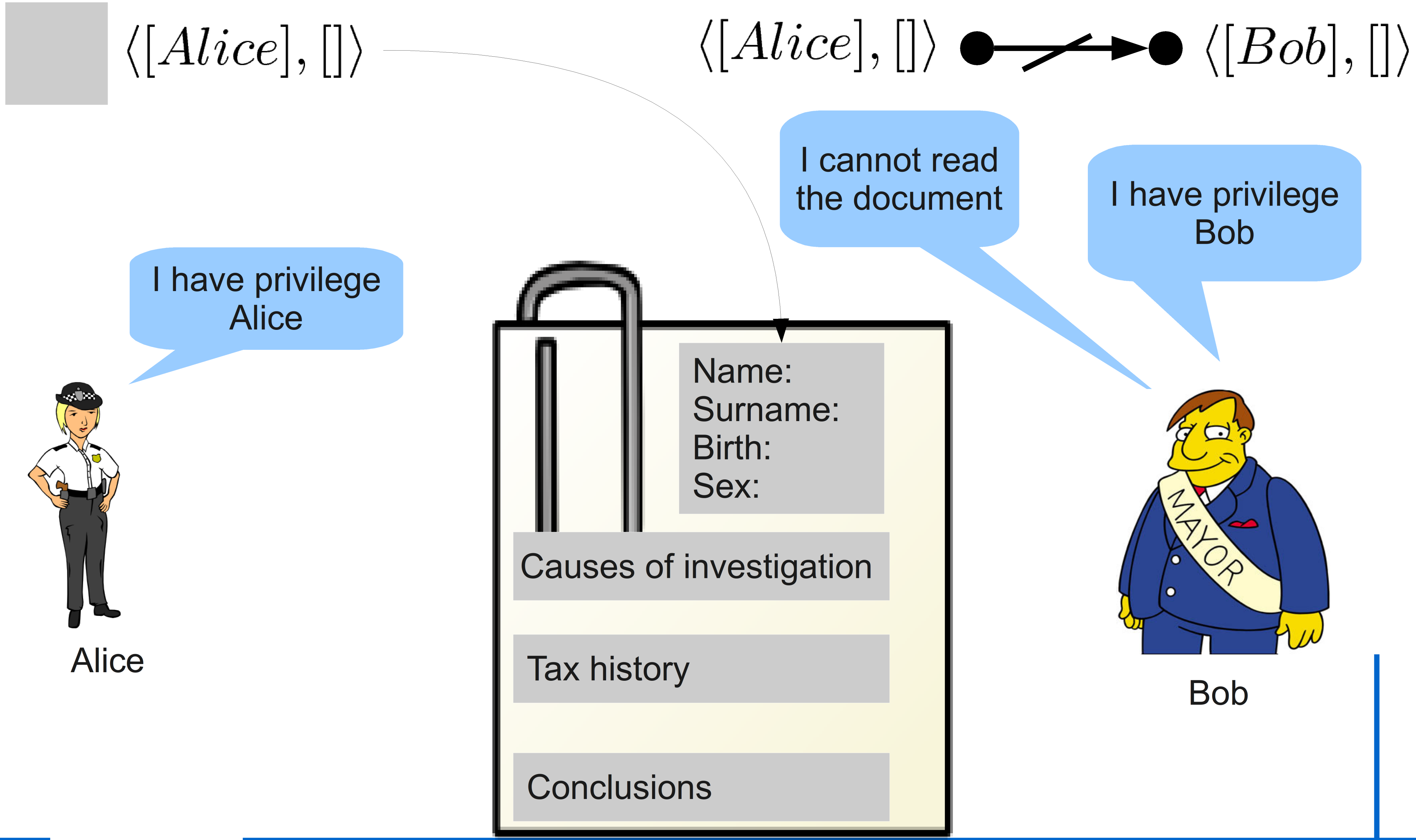
Causes of investigation

Tax history

Conclusions

Bob

Declassification



$\langle [Alice], [] \rangle$

$\langle [Alice], [] \rangle \; \bullet\!\!\not\longrightarrow\!\bullet \; \langle [Bob], [] \rangle$

I have privilege Alice

I cannot read the document

I have privilege Bob

Name:
Surname:
Birth:
Sex:

Causes of investigation

Tax history

Conclusions

Alice

Bob

Privileges

Given any two DC labels $L_1 = \langle S_1, I_1 \rangle$ and $L_2 = \langle S_2, I_2 \rangle$, and a privilege set P, we can alternatively define the "can-flow-to given P" relation as follows:

$$\frac{\langle S_1, I_1 \wedge P \rangle \sqsubseteq \langle S_2 \wedge P, I_2 \rangle}{\langle S_1, I_1 \rangle \sqsubseteq_P \langle S_2, I_2 \rangle}$$

$$\begin{tikzcd} \langle [Alice] \wedge [Bob], [] \rangle \arrow[r, "/" marking, "\sqsubseteq"'] & \langle [Alice], [] \rangle \\ \langle [Alice] \wedge [Bob], [] \rangle \arrow[r, "\sqsubseteq_{\mathrm{Bob}}"'] & \langle [Alice], [] \rangle \end{tikzcd}$$

Bob declassifies his data

$$\begin{tikzcd} \langle [], [Alice] \rangle \arrow[r, "/" marking, "\sqsubseteq"'] & \langle [], [Alice] \wedge [Bob] \rangle \\ \langle [], [Alice] \rangle \arrow[r, "\sqsubseteq_{\mathrm{Bob}}"'] & \langle [], [Alice] \wedge [Bob] \rangle \end{tikzcd}$$

Bob endorses the data

Declassification

▨ $\langle [Alice], [] \rangle$

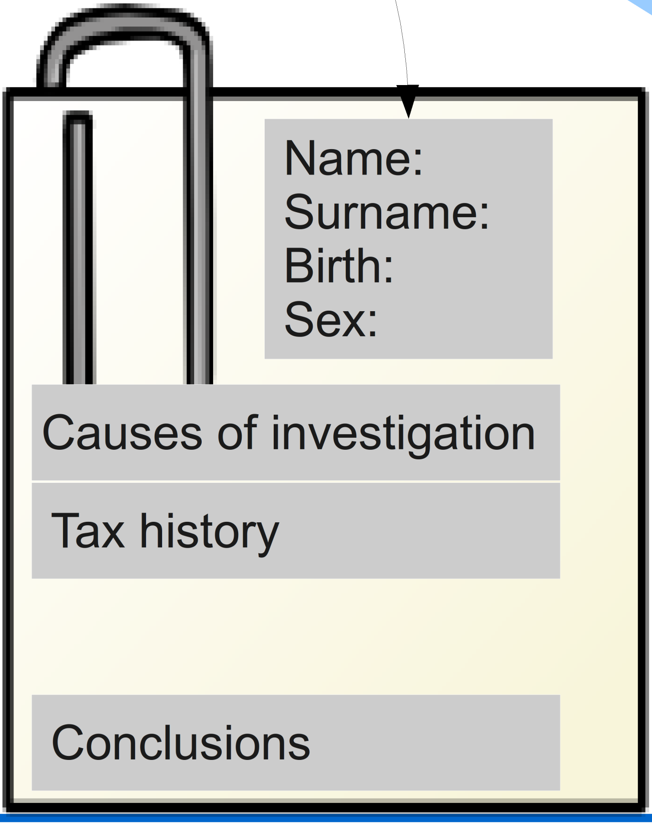
$\langle [Alice], [] \rangle \bullet\!\!\not\longrightarrow\!\!\bullet \langle [Bob], [] \rangle$

I cannot read the document

I have privilege Bob

I have privilege Alice

Alice



Name:
Surname:
Birth:
Sex:

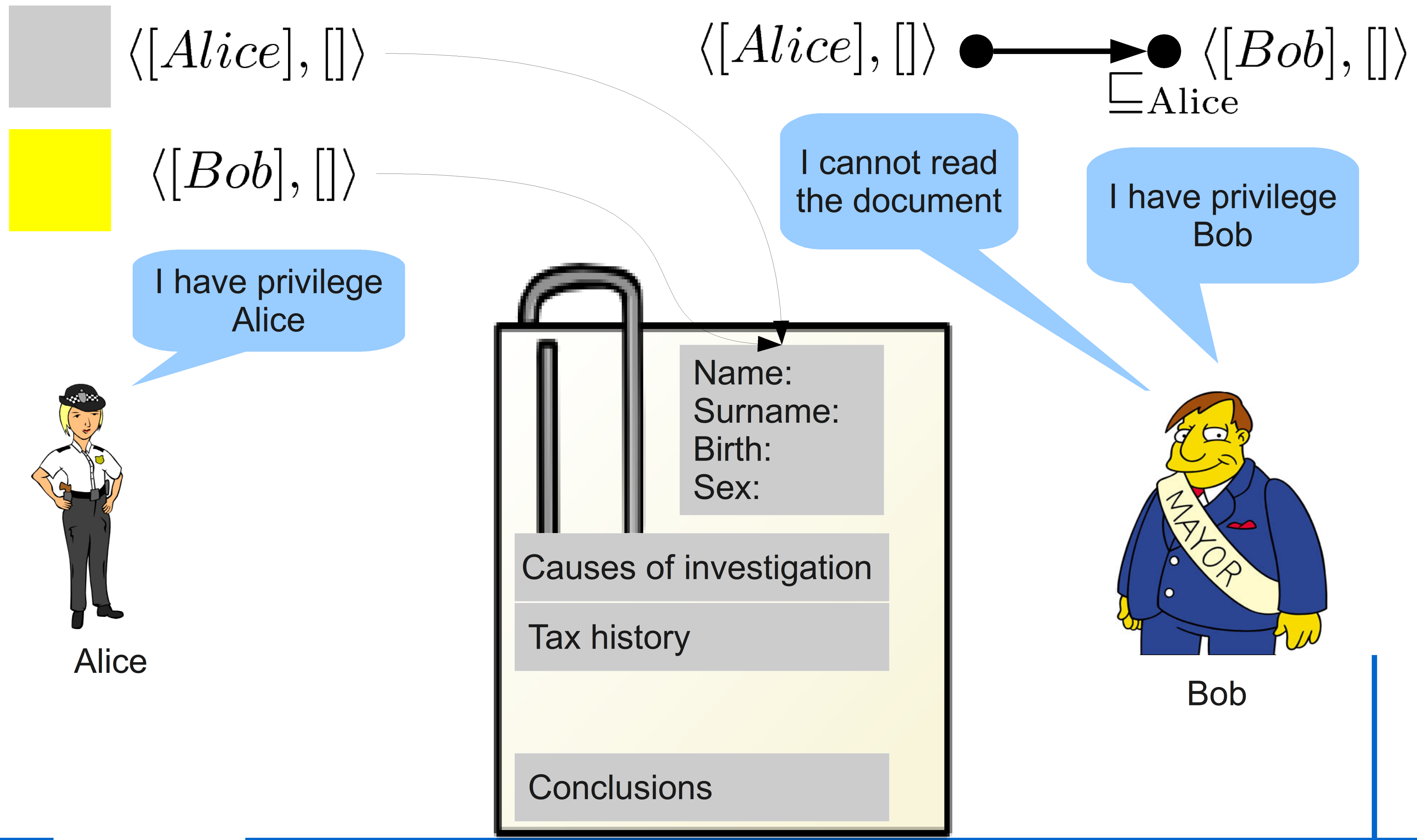
Causes of investigation

Tax history

Conclusions

Bob

Declassification



⬜ $\langle [Alice], [] \rangle$

🟨 $\langle [Bob], [] \rangle$

$\langle [Alice], [] \rangle \xrightarrow[\sqsubseteq_{\mathrm{Alice}}]{} \langle [Bob], [] \rangle$

I have privilege Alice

I cannot read the document

I have privilege Bob

Name:
Surname:
Birth:
Sex:

Causes of investigation

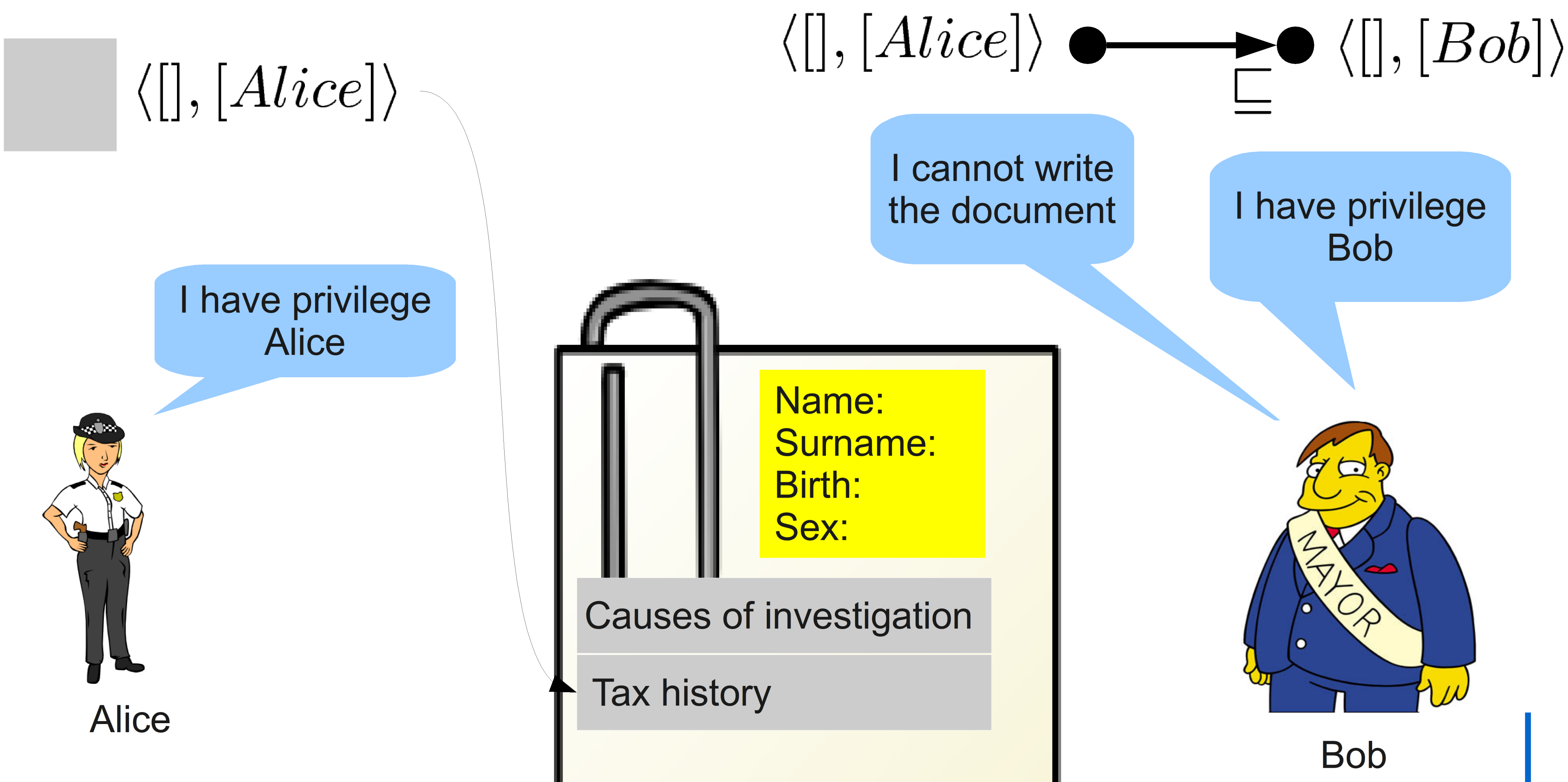
Tax history

Conclusions

Alice

Bob

Endorsement



$\langle [], [Alice] \rangle$

$$\langle [], [Alice] \rangle \xrightarrow[\sqsubseteq]{} \langle [], [Bob] \rangle$$

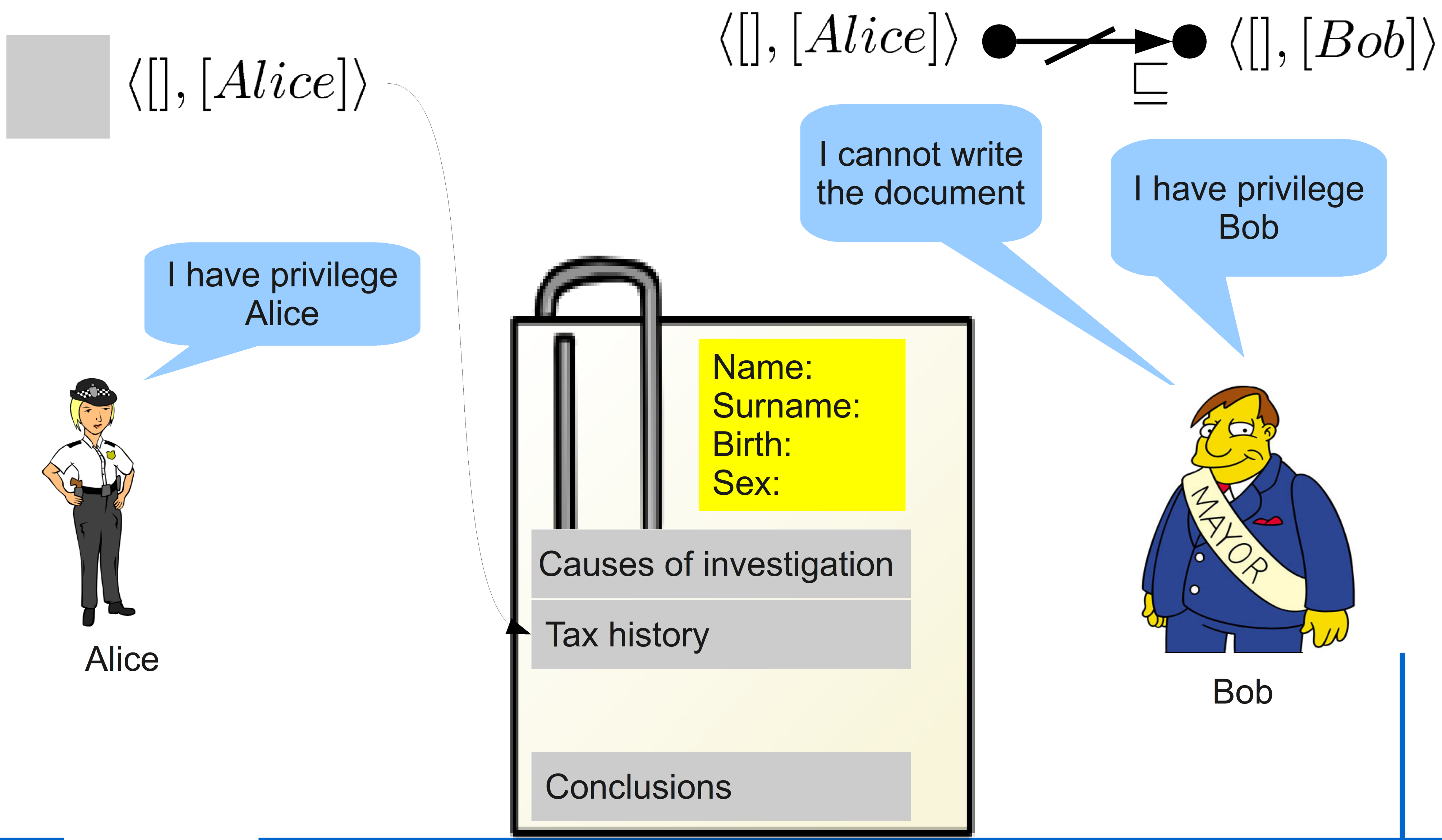
Name:
Surname:
Birth:
Sex:

Causes of investigation

Tax history

Conclusions

Endorsement



$\langle [], [Alice] \rangle$

$\langle [], [Alice] \rangle \underset{\sqsubseteq}{\not\longrightarrow} \langle [], [Bob] \rangle$

I have privilege Alice

Alice

Name:
Surname:
Birth:
Sex:

Causes of investigation

Tax history

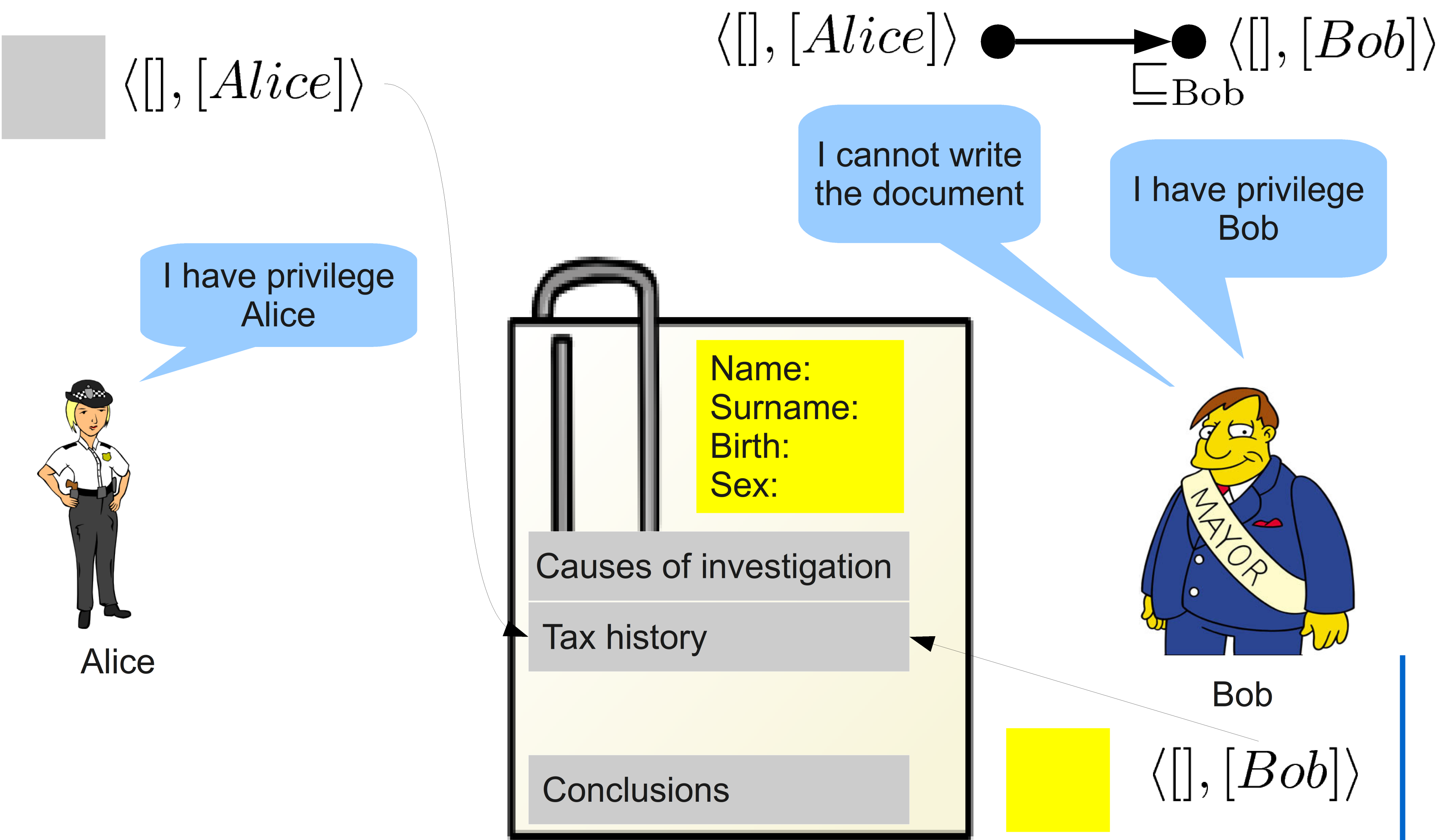
Conclusions

I cannot write the document

I have privilege Bob

Bob

Endorsement



$$\langle [], [Alice] \rangle \;\bullet\!\!\xrightarrow{\quad}\!\!\bullet\; \langle [], [Bob] \rangle$$

$\sqsubseteq_{\text{Bob}}$

$\langle [], [Alice] \rangle$

I cannot write the document

I have privilege Bob

I have privilege Alice

Name:
Surname:
Birth:
Sex:

Causes of investigation

Tax history

Conclusions

Alice

Bob

$\langle [], [Bob] \rangle$

A Library for DCLabels in Haskell

- It is in a experimental phase
  - Remember that it is work-in-progress!
- I adapted the library for this course
- In the future, you might refer to the official release
- Check the webpage of the course to get the installation instructions

Creating DCLabels

```
module Labels where

import DCLabel.Safe
import DCLabel.PrettyShow
```

It can use DCLabels without the capability to create privileges

```
c1 = "Alice" .\/. "Bob"
```

Categories (disjunctions)

```
l1 =  "Alice" .\/. "Bob" ./\. "Carla"

l2 = "Alice" ./\. "Carla"
```

Labels (conjunctions of disjunctions)

```
dc1 = newDC l1 l2

dc2 = newDC "Deain" "Alice"
```

DCLabels

Join, Meet, and $\sqsubseteq$

```
*ExamplesDCLabels> pShow dc1
<{["Alice" \/ "Bob"] /\ ["Carla"]} , {["Alice"] /\ ["Carla"]}>
*ExamplesDCLabels> pShow dc2
<{["Deain"]} , {["Alice"]}>
*ExamplesDCLabels> pShow $ join dc1 dc2
<{["Alice" \/ "Bob"] /\ ["Carla"] /\ ["Deain"]} , {["Alice"]}>
*ExamplesDCLabels> pShow $ meet dc1 dc2
<{["Alice" \/ "Bob" \/ "Deain"] /\ ["Carla" \/ "Deain"]} ,
 {["Alice"] /\ ["Carla"]}>
*ExamplesDCLabels> pShow dc1
<{["Alice" \/ "Bob"] /\ ["Carla"]} , {["Alice"] /\ ["Carla"]}>
*ExamplesDCLabels> pShow $ join dc1 top
<{ALL} , {}>
*ExamplesDCLabels> pShow $ join dc1 bottom
<{["Alice" \/ "Bob"] /\ ["Carla"]} , {["Alice"] /\ ["Carla"]}>

*ExamplesDCLabels> canflowto dc1 top
True
*ExamplesDCLabels> canflowto bottom dc1
True
```

Privileges

```
import DCLabel.Core
import DCLabel.PrettyShow
import DCLabel.NanoEDSL

l1 =  "Alice" .\/. "Bob" ./\. "Carla"

l2 = "Alice" ./\. "Carla"

dc1 = newDC l1 l2

dc2 = newDC "Deain" "Alice"

pr = createPrivTCB (newDC ("Alice" ./\. "Carla") )
```

Only trusted code can create privileges

Creation

Privileges

```
*ExamplesDCLabels> pShow dc1
<{["Alice" \/ "Bob"] /\ ["Carla"]} , {["Alice"] /\ ["Carla"]}>
*ExamplesDCLabels> pShow dc2
<{["Deain"]} , {["Alice"]}>
*ExamplesDCLabels> canflowto dc1 dc2
False
```

Secrecy category of `dc1` cannot be fullfiled by `dc2`

```
*ExamplesDCLabels> pShow $ priv pr
{["Alice"] /\ ["Carla"]}
*ExamplesDCLabels> canflowto_p pr dc1 dc2
True
```

Now it is possible given privileges

Final Remarks

- Label system for mutual distrust scenarios (DCLabels)
  - Conjunction of categories
  - Categories are disjunction of principals
- It allows to express the interest of different parties
- Precisely compute join and meet
- Work-in-progress
  - Comparison with DLM (we have a precise meet)
- More systems need to be built using DCLabels