Lecture 6

Monday, 26 September 2022 14:25

open system

parties aren't in the same security domain

access decisions are made according to credentials, not identity

how to compute trust ?

problem: how to collect credentials which grant access?

→ centralized storage leads to confidentiality/privacy issues

→ hence, we need to store at issuer or at subject

↓

credentials should be stored at parties willing to answer queries about them

store at issuer: must ask all parties which may hold credentials whether, according to them, they have a credential satisfying the query

↓ scalability issues

store at subject: must go through all of the subject's credentials; some credentials may be confidential

doesn't scale well

↓ combine solutions

subject shows credential, issuer verifies validity.

can be much more efficient

we need to establish where we store our credentials

you cannot store all credentials at some 'intermediary party'; they would be unfindable...

ways to search:

- bottom-up
- top-down

we have three types of queries

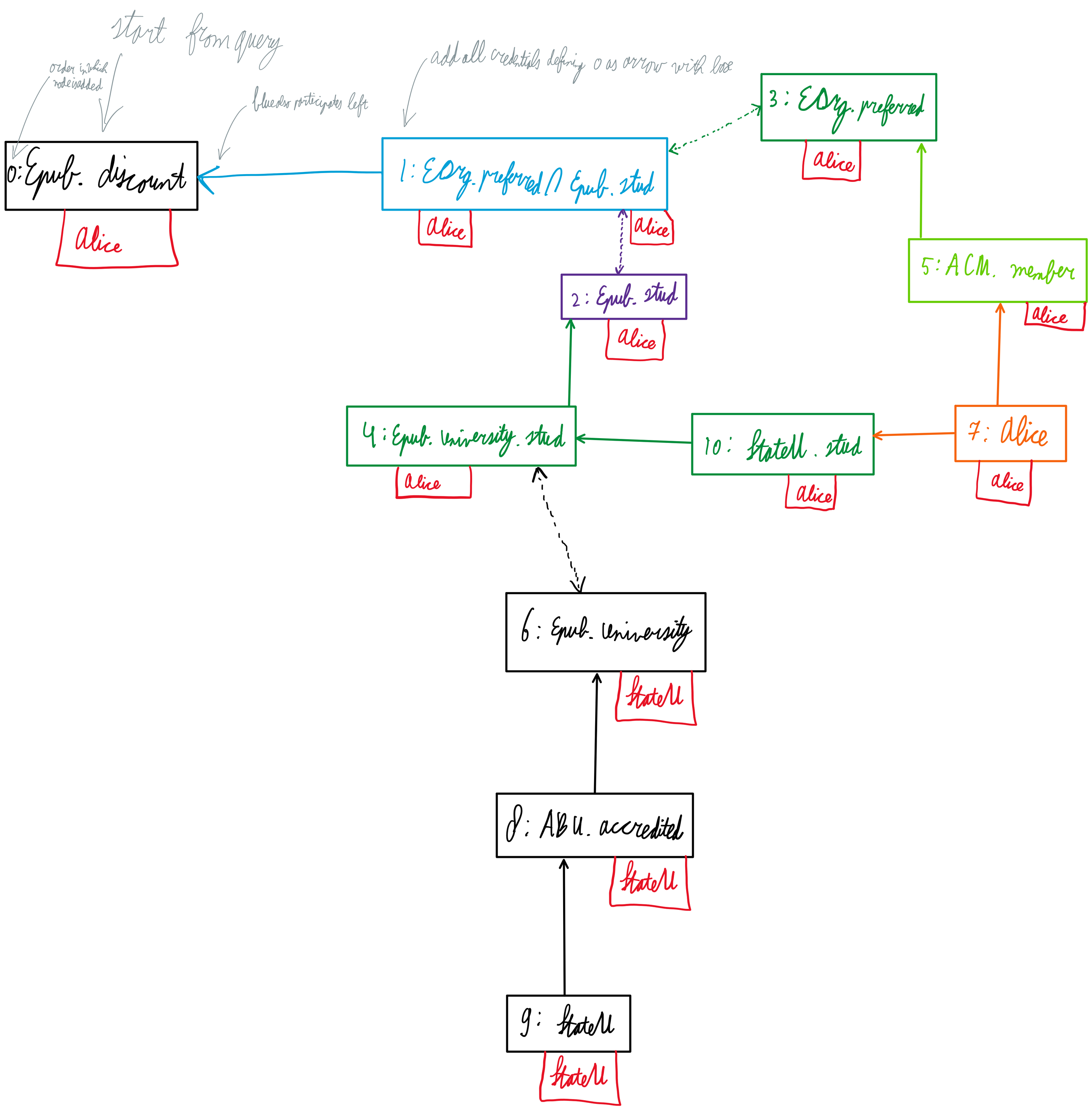
1. given A.r and B, check if $B \in [[A.r]]$
2. given A.r list all the members of $[[A.r]]$
3. given B, list all A.r such that $B \in [[A.r]]$

---

top-down algorithm

(backwards search algorithm by Li)

builds a graph in which nodes are labeled by roles



the top-down algorithm is

- complicated
- goal-directed
- decentralized

questions: who is doing the computation?

can part of the computation be delegated?

can we keep the credentials private?

where do credentials need to be stored for the algorithm to work correctly? → must be at the issuer

alternative: bottom-up algorithm (originally: Li's forward search algorithm) not so nice and probably not so relevant for the exam

↓

can be combined with top-down to find all relevant credentials

---

RT_0 is monotonic: credentials can only add permissions, not remove them

consequence: separation of duty is impossible