

RBAC requires engineering of roles



RBAC was designed for simple systems

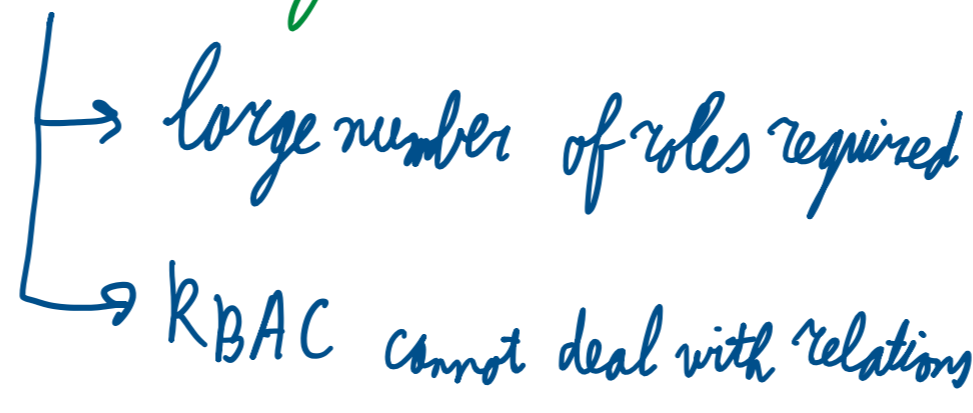
role designing + engineering is difficult & expensive

changes in business can lead to new role hierarchies

RBAC requires attention all the time

RBAC is not context-aware or dynamic
i.e. no restrictions involving location, time

RBAC does not really scale...



attribute-based access control

assign many attributes to users/resources

instead of many users to one role

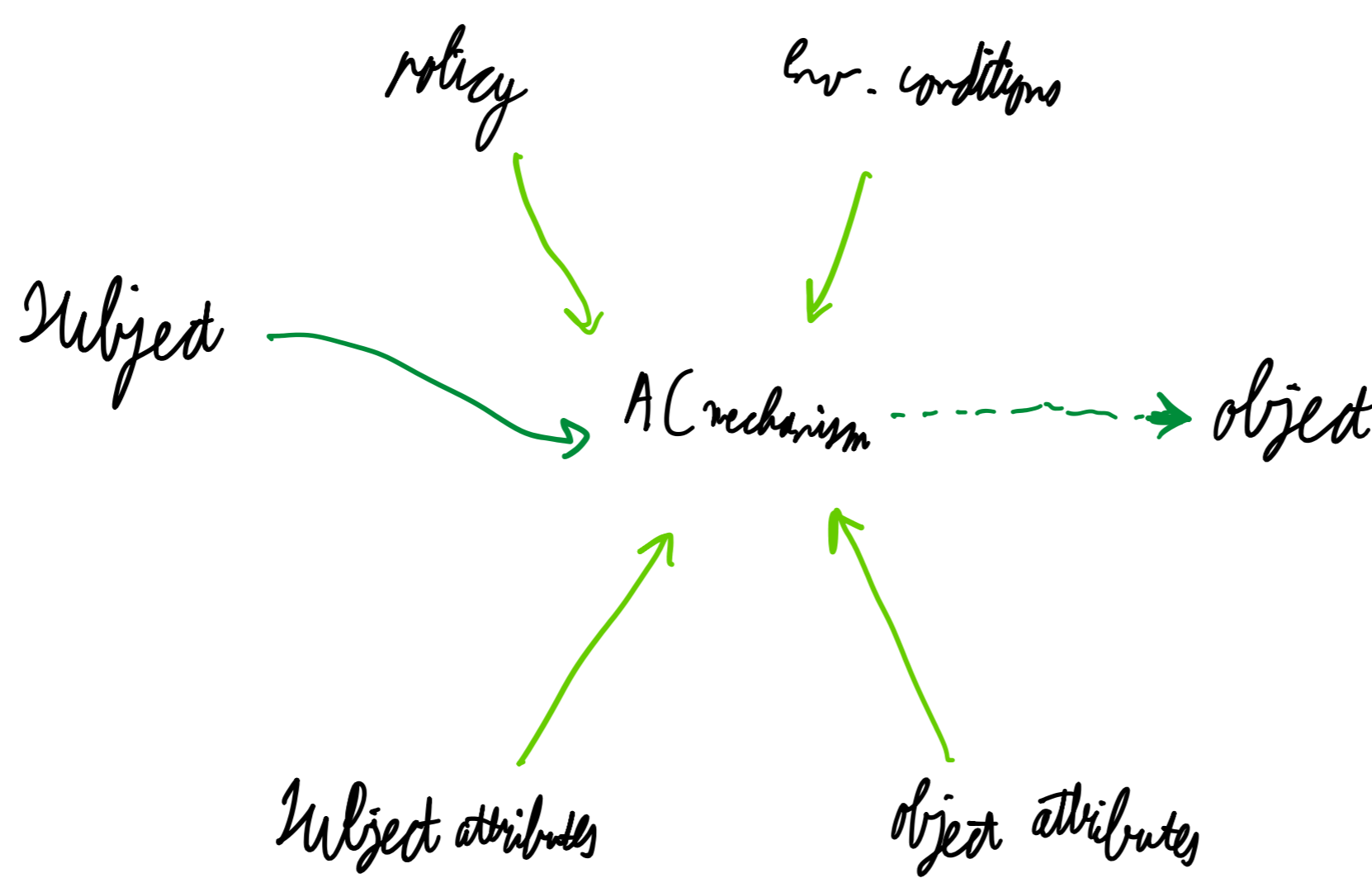
ABAC determines authorization by evaluating attributes against policy / rules

attributes define specific aspects of the subject / object / environment conditions / requested actions, etc...

environmental conditions, dynamics and independent of subject / object

ABAC allows both positive and negative authorizations
i.e. exceptions

format: if condition AND/OR condition then Permit/Deny



ABAC is more granular and easier to set up than RBAC

however, ABAC is more difficult to administer

ABAC is context-aware and can be more scalable than RBAC

questions:

- who defines policies?
- do policies reflect access requirements?
- from where are attributes retrieved?

problem: control over data is often decentralized

and this control may be delegated over the policy definition

how to handle multiple policies over one object?

e.g. use priority (such as permit-overrides or deny-overrides)

how to verify whether a policy meets its requirements?

in ABAC: attributes can be retrieved from external sources

what if these sources are unavailable?
or an incorrect value is provided?