Security in Multi Agent Systems

BY:

BRUNO RAFAEL ALVES

Case of study: Smart Parking

- Smart Parking aims to make citizens' life more comfortable using emergent technologies.

- Its focus is the car parking.

Smart Parking

- Why?
  - 30% of the traffic is generated by drivers trying to find a spot to park their cars (2015).
  - Time.
  - Pollution.
  - Money.

Smart Parking

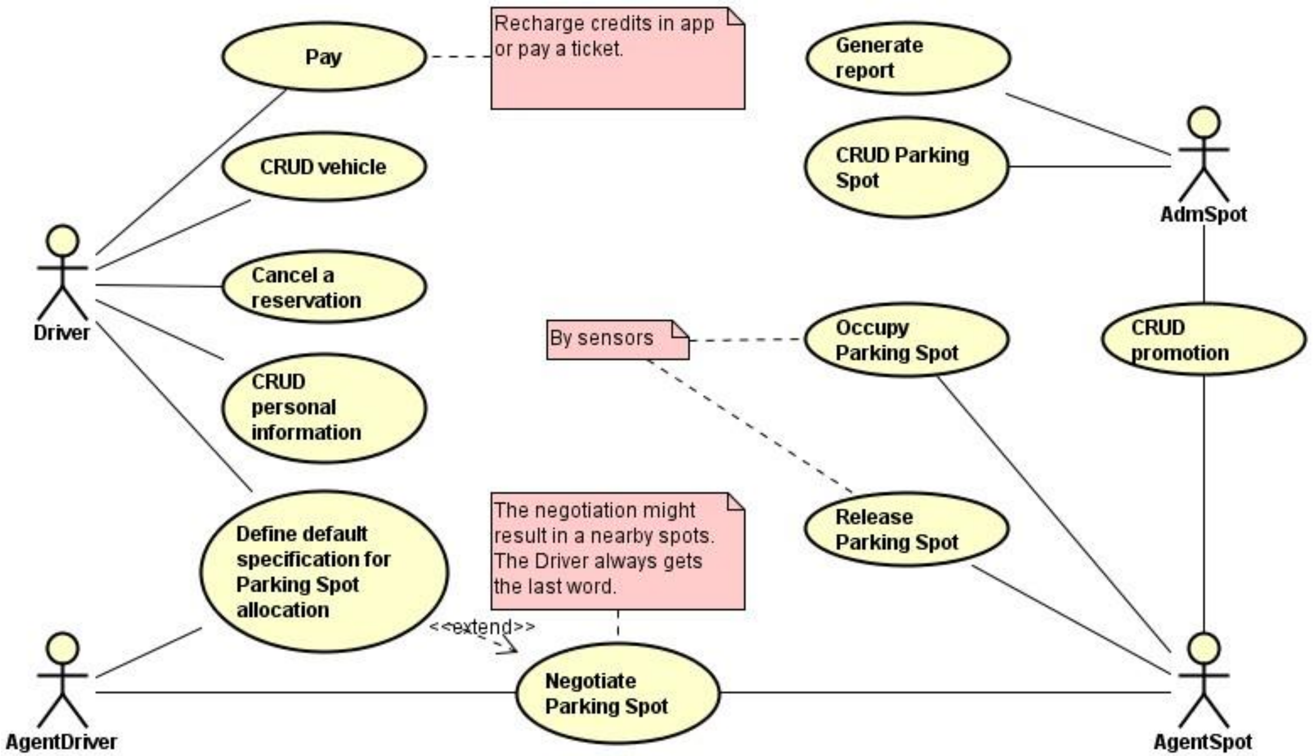
- How?
  - Mobile application.
  - Negotiate a good price.
  - Using Multi Agent System.

Intelligent Agent

ENTITY WHICH ACTS AIMING SOME GOAL.

IT HAS SENSORS AND ACTUATORS.

Architecture



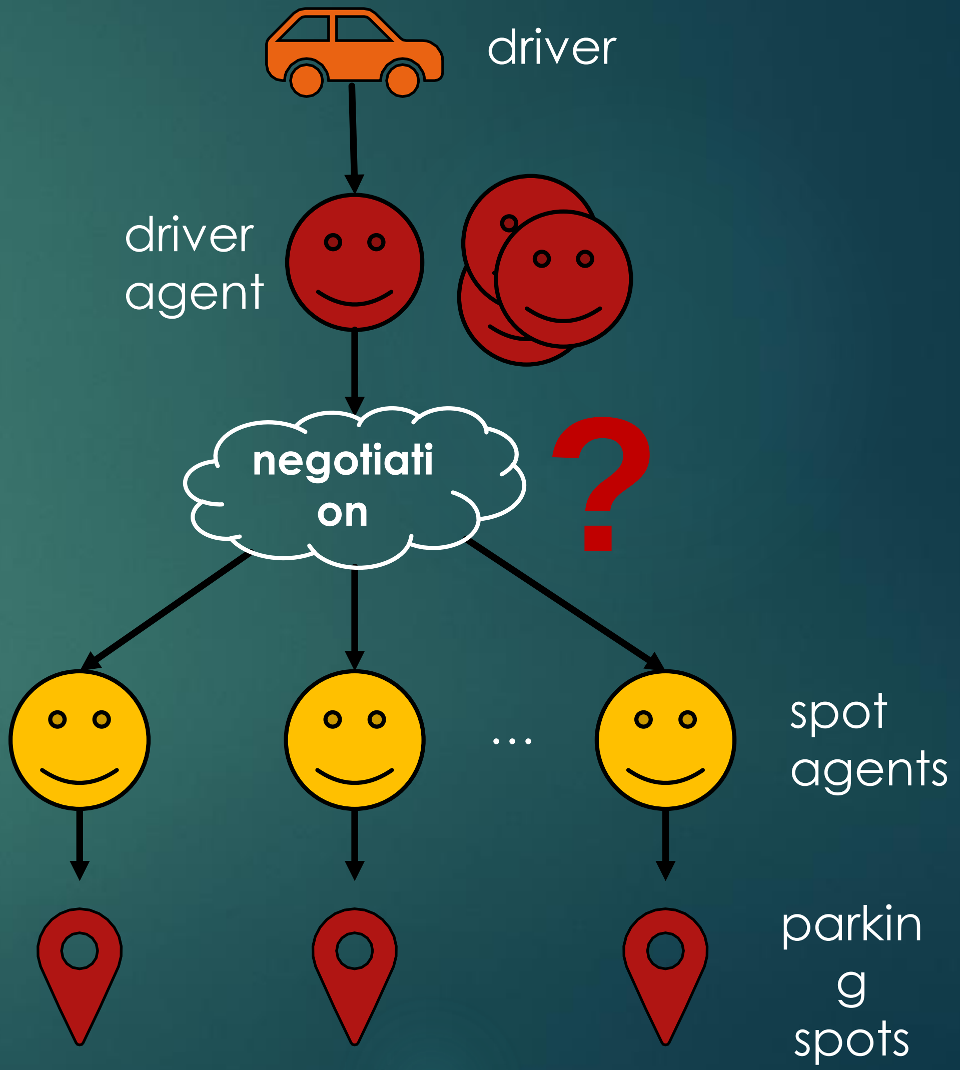
Recharge credits in app or pay a ticket.

By sensors

The negotiation might result in a nearby spots. The Driver always gets the last word.

Multi Agent System

- Agents:
  - Driver Agent.
  - Spot Agent.
- Agents negotiate and find a good price.



Cryptography

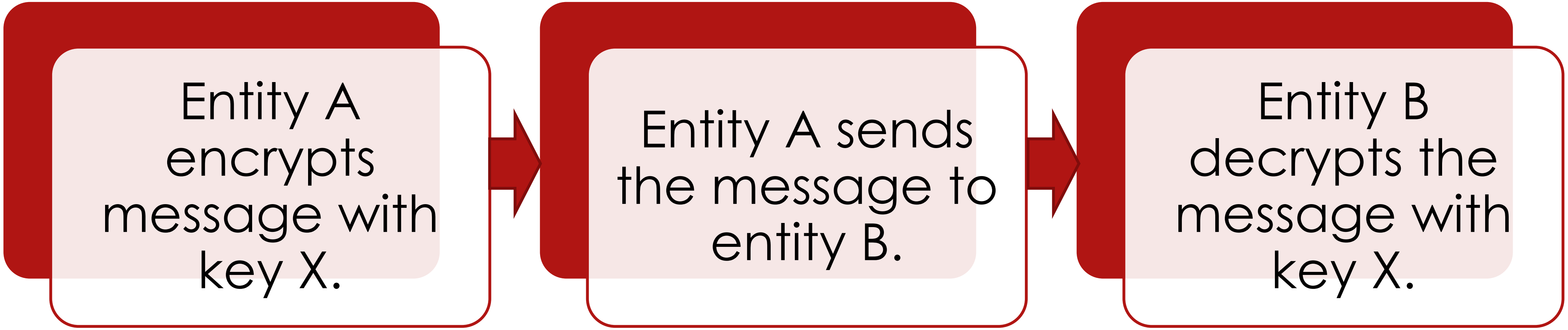
CONFIDENTIALITY.

DATA INTEGRITY.

AUTHENTICATION.

NON-REPUDIATION.

Symmetric key

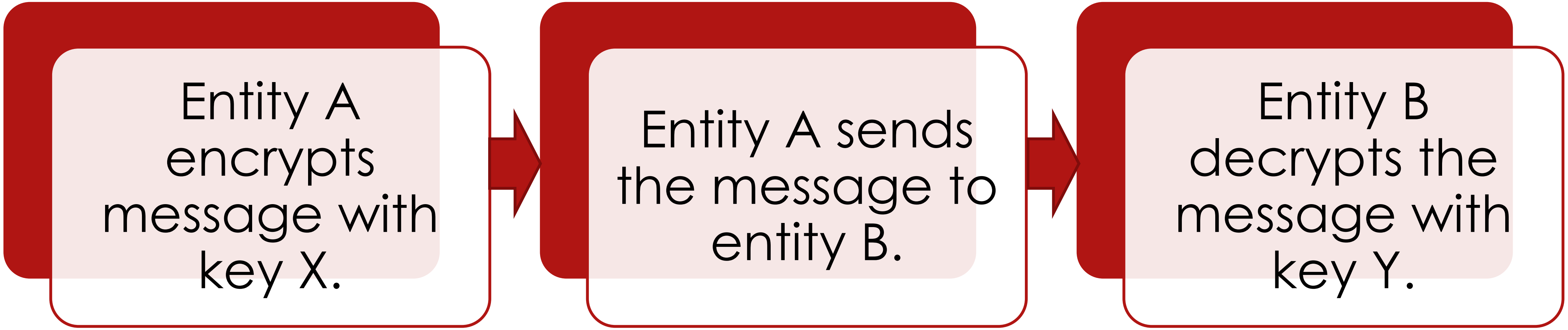


Entity A encrypts message with key X.

Entity A sends the message to entity B.

Entity B decrypts the message with key X.

Asymmetric key



Entity A encrypts message with key X.

Entity A sends the message to entity B.

Entity B decrypts the message with key Y.

Asymmetric key

Public key: all entities know.

Encrypt: only one entity opens.

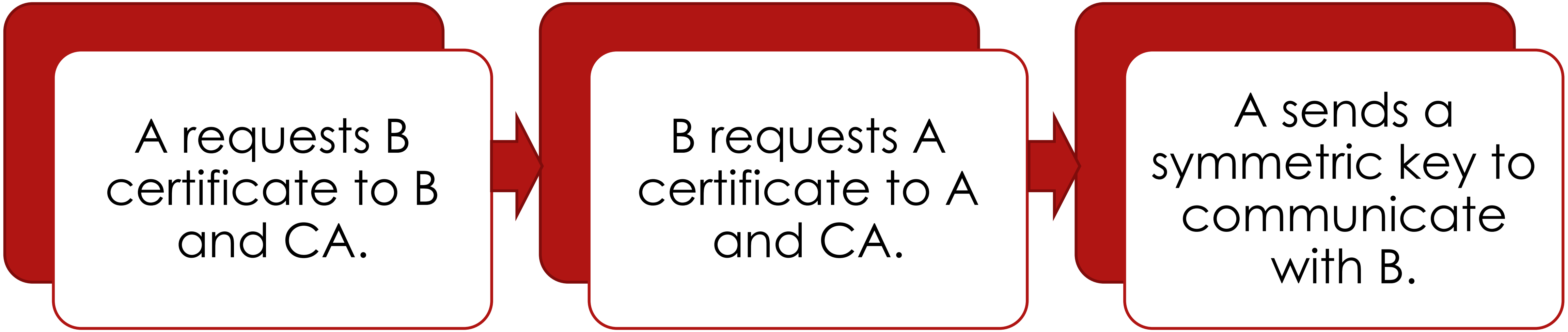
Private key: only one entity knows.

Encrypt: only one entity writes.

Certificate Authorities

- Trustworthy entities.
- Generates certificates using id, public key, etc.
- Database of certificates and public keys.

Secure Sockets Layer Protocol

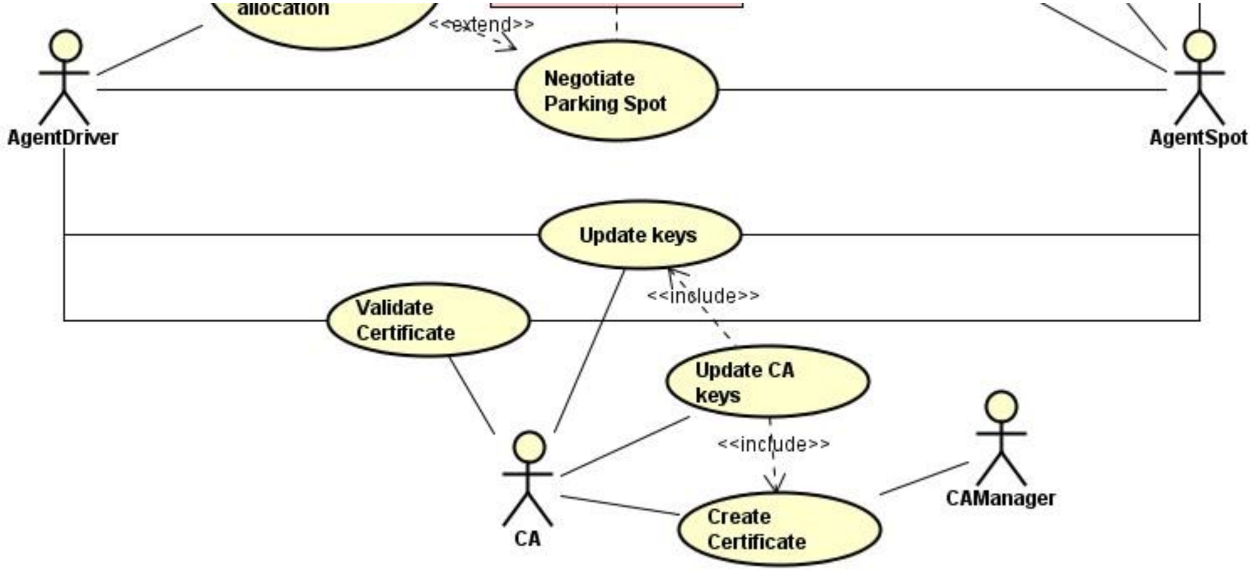


A requests B certificate to B and CA.

B requests A certificate to A and CA.

A sends a symmetric key to communicate with B.

Proposed Architecture

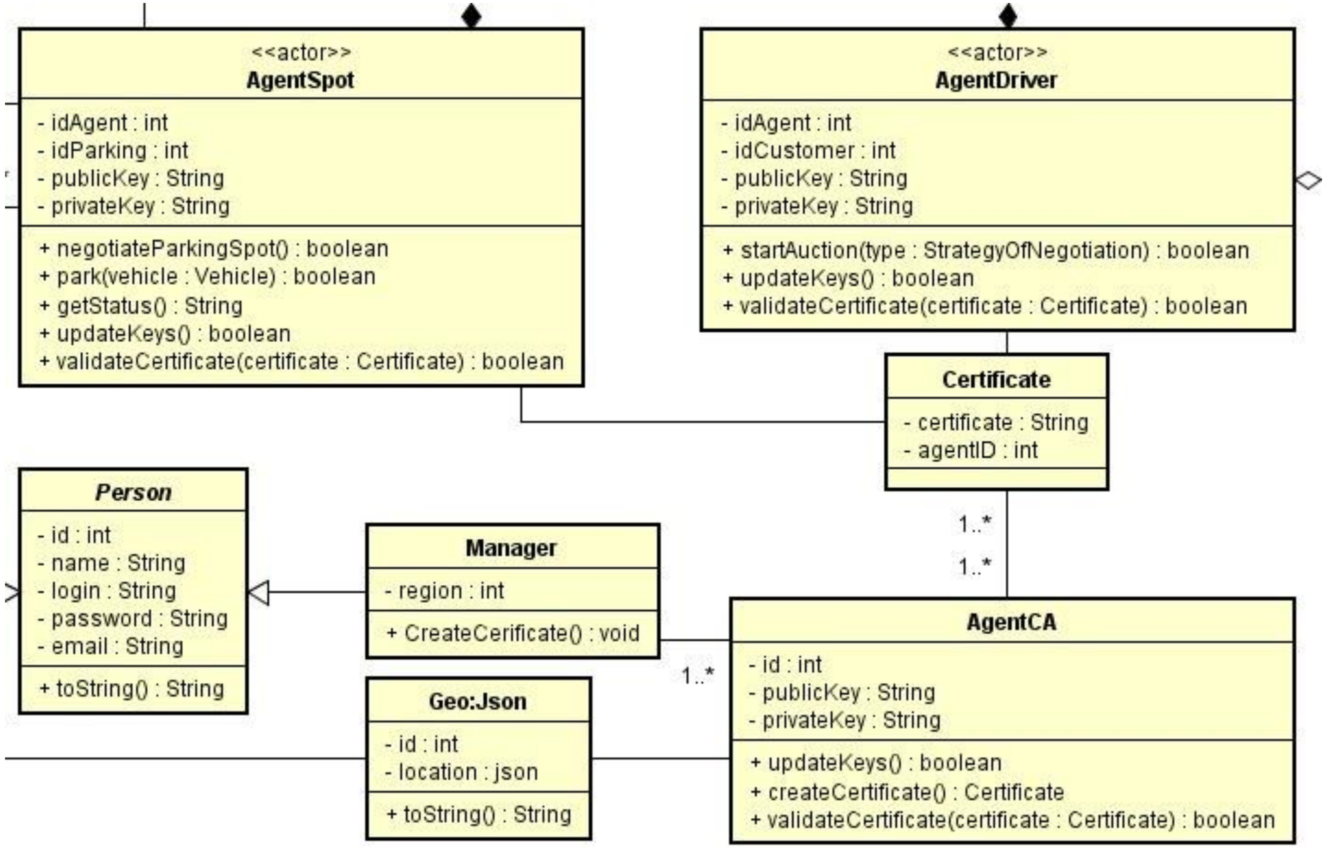


Proposed Architecture

Update Keys: from time to time generate other pair of keys.

Update CA Keys: Inform other CAs about the changes.

Proposed Architecture



<<actor>>
AgentSpot

- idAgent : int
- idParking : int
- publicKey : String
- privateKey : String

+ negotiateParkingSpot() : boolean
+ park(vehicle : Vehicle) : boolean
+ getStatus() : String
+ updateKeys() : boolean
+ validateCertificate(certificate : Certificate) : boolean

<<actor>>
AgentDriver

- idAgent : int
- idCustomer : int
- publicKey : String
- privateKey : String

+ startAuction(type : StrategyOfNegotiation) : boolean
+ updateKeys() : boolean
+ validateCertificate(certificate : Certificate) : boolean

Certificate

- certificate : String
- agentID : int

Person

- id : int
- name : String
- login : String
- password : String
- email : String

+ toString() : String

Manager

- region : int

+ CreateCerificate() : void

Geo:Json

- id : int
- location : json

+ toString() : String

AgentCA

- id : int
- publicKey : String
- privateKey : String

+ updateKeys() : boolean
+ createCertificate() : Certificate
+ validateCertificate(certificate : Certificate) : boolean

Proposed Architecture

All agents have a pair of keys and a way to update them.

Agents have a way to validate entities (CA).

Certificates are encrypted by CA.

Proposed Architecture

- CA might be a parking with some previous structure.
- CAs might care about an area, not the entire system.
- CAManager is a human. He creates the first certification of an agent.

Questions