Security in multi agent systems

Bruno Rafael Alves[1], André Pinz Borges[1], Gleifer Vaz Alves[1], Paulo Leitão[2]

[1] DAINF - Universidade Tecnológica Federal do Paraná (UTFPR),
Ponta Grossa-PR, Brasil.

[2]CEDRI - Instituto Politécnico de Bragança (IPB), Bragança, Portugal.

brunoa@alunos.utfpr.edu.br,{apbirges, gleifer}@utfpr.edu.br

pleitao@ipb.pt

Abstract. *Smart Cities use technology to improve the citizens' life. A branch of it is the Smart Parkings, that intends to solve the problem of parking cars, making it easier to find a spot to park and consequently decrease the traffic of cars. In this context, a cyber-physical architecture, based on Multi Agent Systems, was developed for a Smart Parking System for cars and bicycles by a previous group. It consists of a community of distributed, intelligent and autonomous agents, representing the parking spots and drivers, which negotiate to reach their objectives. This paper takes the security and the support of failures of the system in consideration, improving the architecture with the implementation of SSL and Certificate Authorities.*

Keywords: Multi Agent System; Smart Parking; SSL; Certificate Authority;

1. Introduction

Large cities are crowed places with several problems emerging every day, for example, pollution, energy, waste and traffic. A Smart City [Neirottia et al. 2014] aims to make the citizens' daily life more comfortable and convenient by using emergent technologies in order to provide accessibility to public information and services [Anthopoulos and Fitsilis 2010].

When it comes to traffic, the parking problem can be crucial for the improvement of the smart city concept, since 30% of the traffic is generated by drivers trying to find a spot to park their cars [Silva 2015]. Consequently, a parking that uses advanced technologies to improve its management and the provided services can contribute for a reduction of the traffic [Koster et al. 2014].

On a previous work, an architecture for a smart parking has been proposed [Alves 2019]. This architecture aims to develop a an agent-based solution to improve the management and the organization of a smart parking, focusing in the agility of the services and the profit. Furthermore, the driver is encouraged to cooperate with the system as a whole, for instance, the driver respect the agreement so he doesn't get to pay tickets.

A Smart Parking System is strongly based on the exchange of messages between agents, these communications become a point of interest in the security. This paper proposes a security architecture for the messages exchanged by the agents in the system

proposed before. Furthermore, this approach can be used in similar scenarios. For that, SSL and certificate authorities are used.

2. Smart Parking

Smart cities represents cities that use the technology to improve the well being of the citizens and the efficiency of the city services as a whole [Neirottia et al. 2014]. One branch of it is the Smart Parking. Smart Parking uses technology to solve the parking searching problem and consequently reduce the urban traffic congestion.

As an example of the using technology in a Smart Parking, can be given by means of a mobile application, where the parking spots maybe more accessible and visible to the drivers. Another way to improve the parking is to find the best price for the driver and for the parking manager, making the best agreement for both of them. One way to reach that goal is trough a negotiation, for that, agents can be used in a Multi Agent System (MAS), that is, a system with several agents reacting to each other at the same time. When a driver agent wants to find a place to park, the spot agents can make an offer.

3. Architecture

On a previous work, we have developed an agent-based architecture for Smart Parking. Figure 1 presents the main participants of the system. The Driver represents the person that drives the vehicle; the AgentDriver represents the Driver in the system (most likely a phone application); The AdmSpot represents the spot owner (most likely the parking administrator and, it can also have more than one spot); The AgentSpot is the agent that represents a specific spot in the system (most likely a micro controller).

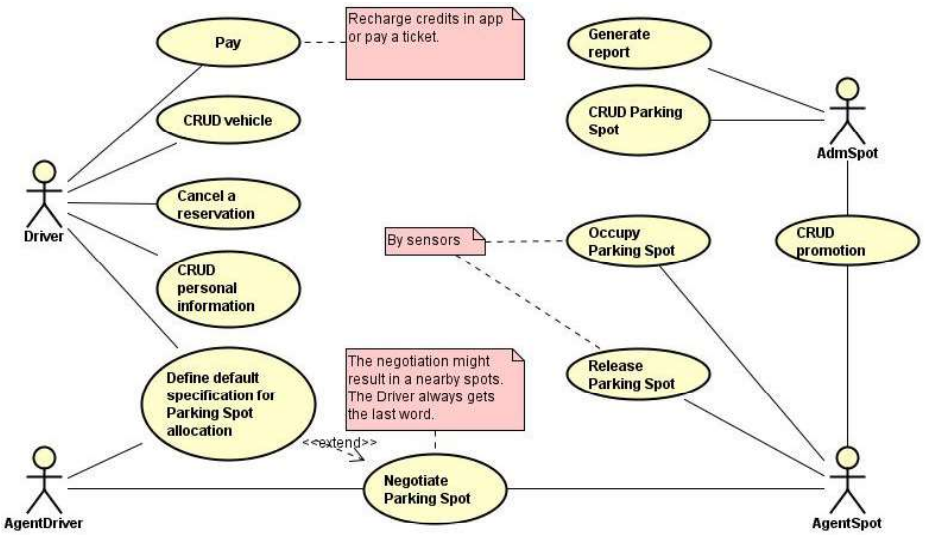

Figure 1. Use Case Diagram Base [Bruno Rafael Alves 2019]

In this architecture, each spot is a module that can be deployed to the system individually. That is, the system can be escalated. Furthermore, the system support failures. If an agent stops to work the system as a whole isn't affected by it. But, it's easy to put an agent with a virus in the system, for example.

The use case 'Negotiation Parking Spot' is a key aspect and will be approached in this paper, since it represents the core of the system. In this use case, the agents communicate with each other to get in agreement and decide the best offer. Moreover, this is the use case that has the higher amount of exchanged messages between agents, becoming one of the possible security breaches.

4. Cryptography

In 1978 Rivest, Shamir, and Adleman discovered the first practical public-key encryption and signature scheme, now referred as RSA. Based on that, in 1991 the first international standard for digital signatures (ISO/IEC 9796) was created [Katz et al. 1996]. This protocol is based on the goals of the cryptography, explained in [Katz et al. 1996]. They are:

1. Confidentiality: A service that does not allow others but the ones previously authorized to see the content of the information. For example, physical protection to mathematical algorithms, which render data unintelligible, can provide confidentiality.
2. Data integrity: A service that prevent the data to be modified by an unauthorized person. To assure that, the detection of an unauthorized change in the data must be identifiable.
3. Authentication: A service related to identification. Both sides must be able to identify the other when a message is received. The information about the data should also be authenticated. Usually divided into two major classes: entity authentication and data origin authentication.
4. Non-repudiation: A service that ensures all the commitments or actions. When an entity tries to deny some previous action, a third trustworthy party is needed to verify an confirm the operations.

5. Certificate Authorities and SSL

In order to guarantee the identity of the agents in the network the Certificate Authority (CA) certificates all the entities in the system. In other words, it's a database of certificates. The system can have more than one CA, each one having a copy of the data base. In that way, the system support failures.

The CAs contain information about the entities, like their version, serial number and public key. The certificate of that entity and all that information are encrypted. All the entities in the system but the CAs send messages using the public key of the destination entity, so only that entity can read the message. On the other hand, the CAs use their own private key to generate the messages, allowing any entity to read the certificates with the public key and making it almost impossible to fake.

For instance, if an entity A wants to communicate with an entity B: first of all, the entity A requests the certificate of the entity B for the entity B and to one CA. Entity A compares them to verify the agent B. After, entity B makes the same verification of the entity A. Once those are verified, they can exchange a symmetric key for a simpler and secure communication called Secure Sockets Layer (SSL) [Elgamal and Hickman 1997].

Now, only the entities that have the symmetric key can join the conversation. In this case, entity A and B.

6. Cryptosystem Proposed

As previously mentioned, our goal is to improve the Smart Parking architecture by adding a security layer. This layer comprises SSL and CAs in order to secure the messages in the system.

Initially, the first CA is generate. That CA now can validate the agents (entities) and other CAs. In that way, the system is protected, since, it doesn't rely only on one CA. It will make the system faster, since the response time will be smaller than if there was only one CA answering all the requests. Also, if the system has only one CA, it would probably be far from most of the agents increasing the response time.

The CAs can be the parking lot that already have some structure, for instance a parking with a data base of clients. And, above all, it must be reliable. To define it, the Certificate Manager (a person) is needed. All the agents deployed in the system will have at least one CA listed. Then, when a new agent is deployed it is certified by one CA. After, the CA can inform other CAs in the system.

Another improvement to the system is to have the CAs responsible of a region or a sector, not the entire system. When an agent requests a certificate of an outsider agent, the CA of the region asks others in other regions. Each region might have more than one CA to not compromise the integrity of a region. Also, it will not need a lot of space and processing in the CAs, since each one will handle a small part of the whole system. That architecture is shown in Figures 2 and 3.

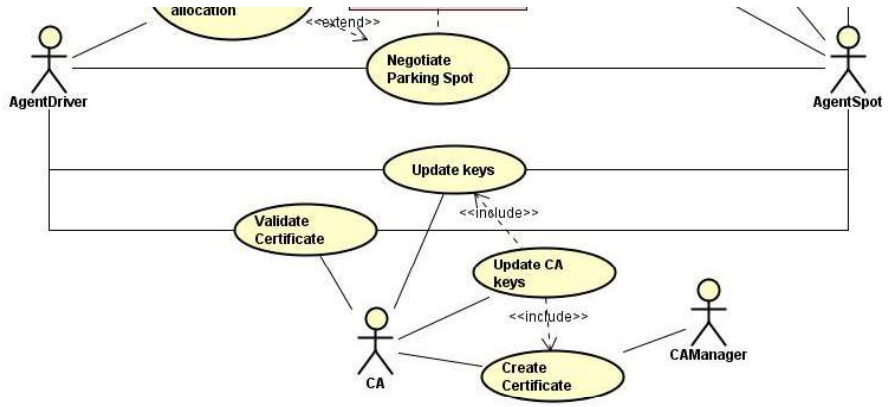

Figure 2. Use Case Diagram Proposal

The Figure 2 refers to the use case diagram of the proposed system. The difference are the additions of two actors and four use cases. The actors represents the person that manage the CA (CAManager) and its agent in the system (CA). With those use cases, all the CA operations can be established and a secure channel can be created (SSL). The use cases are:

- Create Certificate: When a new agent or CA is deployed in the system, one person (CAManager) needs to add it to one CA data base, generating the certificate of it.

- Update CA keys: When the CA update the keys or deploy some agent in the system, it needs to inform the other CAs to update their data base, informing the alterations.
- Validate Certificate: This use case is used when an agent needs to get a certificate about other entity in the system.
- Update keys: When a new agent is created or a key expires the CA create a new pair of keys.

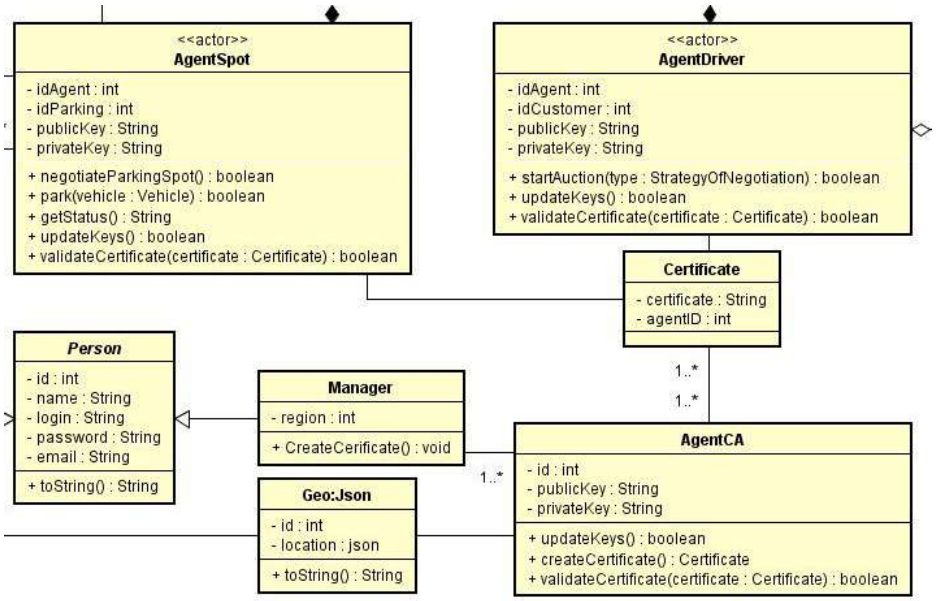

Figure 3. Class Diagram Proposal

Figure 3 shows a part of the class diagram relevant for this propose works. Three main classes are needed, the AgentCA, the Certificate and the Manager of the CA. The first represents the agent of the CA in the system that will store the certificates described in class Certificate. The last one is the previous mentioned CAManager. Those classes are explained bellow:

- AgentCA: This agent must have an unique id, publicKey and PrivateKey, just like any other entity in the system. Moreover, all entities must be capable of update the pair of keys (method "updateKeys"). This agent has two more important methods, the creation of a certificate and the validation of it.
- Certificate: May be the most crucial class in this architecture, it represents the certificates of every entity. Its content is encrypted with the CA private key, but everyone has its public key in order to read it but not fake it.
- Manager: The class that represents the CAManager in the system, capable of create a certificate for a agent that is being deployed in the system. Notice that is the Manager who has the region parameter. When a CA wants to communicate with others, it searches for other managers but its own. In that way, the communication is faster than check all the CAs in the system.

It‘s possible to the agent stores its public key in the CA aside with its own certificate, because, again, the CAs will not be needed to handle the whole system data, just a small part of it. At the end, the system will have several CAs grouped in regions or sectors holding the public keys and the certificates of the entities of the system. That can guarantee the security and integrity of the system as a whole.

Furthermore, the expiration of the keys is an important issue. From time to time, all the keys must be changed to guarantee the security of the system. That's because the algorithms are not unbreakable, but needs a lot of time to decrypt it. And if the agent itself change its keys from time to time, it makes the decryption of any message harder. The update of the keys can be made automatically by one CA and be distributed to the CAs in the region very fast.

7. Conclusion

As mentioned in the Introduction, the previous architecture on Smart Parking does not handle security issues in the exchange of messages between agents. Thus, here we propose an extension in our architecture in order to offer security services. Despite of the initial human need, once that are only agents in the system, it becomes autonomous again. Even when the keys expire, it does not take that much of the system to update itself, since just a few part of the CAs is needed to take care of that update.

Finally, this work proposed an architecture capable to improve the security of a MAS in the case of a Smart Parking. Nonetheless, our approach can be easily adopted in other scenarios, where a MAS is used.

As a future work, we intend to be focused in study other methods for cryptography that were not exposed here and may be better for this case of study, for instance the DSA [A.N.S.I. 1997]. Moreover, an architecture that uses more than one method could add some randomness to the system, improving the security. Furthermore, comparative performance tests about the system with and without this secure module need to be done. Finally, the implementation of Transport Layer Security (TLS) instead of the SSL should be studied, since the TLS is almost a new version of it.

References

Alves, B. R. (2019). Smart parking system: Architecture, protocols and prototype. Master's thesis, Instituto Politécnico de Bragana.

A.N.S.I. (1997). Public key cryptography for the financial services industry: Part 1: the digital signature algorithm (dsa).

Anthopoulos, L. and Fitsilis, P. (2010). Digital cities: Towards connected citizens and governance. In *Politics, Democracy and E-Government*, pages 275–291.

Bruno Rafael Alves, Gleifer Vaz Alves, A. P. B. P. L. (2019).

Elgamal, T. and Hickman, K. E. (1997). Secure socket layer application program apparatus and method. US Patent 5,657,390.

Katz, J., Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Koster, A., Koch, F., and Bazzan, A. L. C. (2014). Incentivising Crowdsourced Parking Solutions. In Nin, J. and Villatoro, D., editors, *Citizen in Sensor Networks*, volume 8313 of *Lecture Notes in Computer Science*, pages 36–43. Springer.

Neirottia, P., Marcob, A. D., Caglianoc, A. C., Manganod, G., and Scorranoe, F. (2014). Current trends in smart city initiatives: some stylised facts. *Cities*, 38:25–36.

Silva, J. (2015). Aplicativo de vagas pode reduzir trânsito nas cidades e facilitar sua vida.