nTicket - Architecture of a highly scalable NFC based E-Ticketing system

B. Zsoldos[1], F. Molnár[2],

[1]Budapesti Műszaki és Gazdaságtudományi Egyetem
- Közigazgatási Informatikai Központ
Magyar tudósok körútja 2, 1117 Budapest, Hungary

[2]Netlock Kft.
Expo tér 5-7, 1101 Budapest, Hungary
E-mail: molnar.ference@netlock.hu

Abstract: nTicket is a project that aims to establish a performance-based pricing system for Public Transport by means of the new generation of NFC Infrastructure. The project is partly financed by a governmental, KMR_12 r&d fund.

nTicket technology enables the customers' phones to handle business logic, therefore, no active devices – like terminals - are required to be installed. As a result of the above, the deployment of nTicket is much more cost-efficient than that of a general E-Ticketing system.

To avoid potential trickeries, nTicket uses a system-wide PKI-like solution. Changing the source of the phone application will not allow hackers to create or modify ticket information.

To serve the great number of transactions generated by users in a city/country, nTicket introduces a set of new components built on top of a modularized architecture.

Keywords: E-Ticketing, Public transport, NFC

1. Introduction

Synthetizing the positive experiences with the current electronic ticket systems and mixing them with the latest technological achievements, in this project, we establish a performance-based pricing system for public transport, which is going to be experimentally implemented on a new generation NFC-based (Near Field Communication) infrastructure.

The main point of the designed solution is - breaking with the former solutions - to equip stations, staging and stopping points with vandal- and fail-safe, intelligent passive NFC labels.

The registration of service utilizations is performed by the passengers NFC compatible tools (cell phone) connected to the central system of the service provider,

this is why it is no longer necessary for the service provider (transport company) to procure expensive validation devices.

2. Security challenges

In case an active terminal is installed on every station or vehicle, security issues can be handled inside the terminal. It is presumable that the code inside the terminal is not accessible by hackers; it can be updated only by the service provider.

The situation is different when the business logic is implemented in the customer's phone. It cannot be guaranteed that the customer does not decompile the application to change its sources. The business logic, therefore, must not be vulnerable to changes in the application code, which may enable the device owner or hacker to generate fake tickets.

Every NFC device contains a Secure Element (SE) that can be protected against code manipulation. These chips allow the programmers to run low level codes, which are mainly cryptographic algorithms. The owner of the phone cannot access and change the source located in the SE.

3. Implementation details

3.1. Passive tag

The passive tags have to meet the below requirements to ensure safe communication between the nTicket components even if the cell phone application is vulnerable.

1. They have to be able to communicate with NFC devices working as card-readers;
2. They have to be capable of creating digital signatures (e.g. ECDSA);
3. The public key signed by the certificate issuer has to be readable from the passive tag. (CERT)

3.2. Secure Element Applet

Every NFC device contains a Secure Element (SE) that can be protected against code manipulation. These chips allow the programmers to run low level codes, which are mainly cryptographic algorithms. The owner of the phone cannot access and change the source located in the SE.

An applet is uploaded to the customer's phone to ensure that the transport data cannot be modified retrospectively. The content of the SE Applet is the following:

1. Private key for digital signature;
2. Certificate in compact format (CERT);
3. Incrementer: a field whose value is incremented by one each and every time the Applet signs a transportation transaction. The value of the incrementer field will be an element of the signature;

4. Transaction log: a vector (array) storing the signed transaction since the latest synchronization;
5. Timestamp of the last server synchronization;
6. Public key of the nTicket certificate issuer.

3.3. Communication between the passive tag and the SE

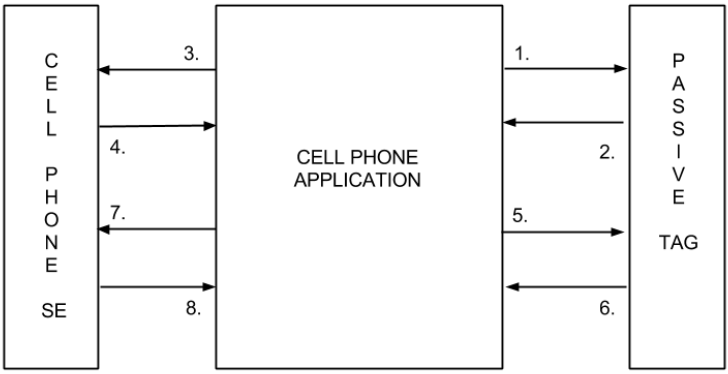

Figure 1. Using passive tags

1. The cell phone recognizes the passive tag and asks for the content of the CERT file in it;
2. The passive tag returns the CERT file and the cell phone checks whether it is from an original nTicket device or not;
3. The nTicket-Phone application forwards the transactional data, the passive tag signature and the CERT file in a compact format to the SE Cardlet;
4. The SE Cardlet does the following:
   - increments the current value of the incrementer;
   - concatenates the incrementer and the transactional data and signs them;
   - returns the signature to the nTicket-Phone application;
5. The application concatenates the transactional data and the signature returned by the SE Cardlet and sends it to the passive tag so that it can sign it;
6. The passive tag signs the data and returns them to the nTicket-Phone application;
7. The SE Cardlet does the following:
   - checks whether the data are from an original nTicket device or not;
   - uploads the new record to the cell phone

- returns a success value

8. The cell phone application notifies the passenger about the success of the transaction.

3.4. Using the phone with an active device

In some cases there may be active devices on the vehicles, so that it becomes possible to get information not only about the start and end points of a route, but also about the line the passenger travelled on. These information can be acquired by the help of the data found in the transaction log.

The active device can communicate with the SE Cardlet indirectly, the nTicket cell phone application has no role in the process.

3.5. Emptying the transaction log

The transaction log can only be emptied if the server confirmed that it got and saved the data. For this, the following steps are required:

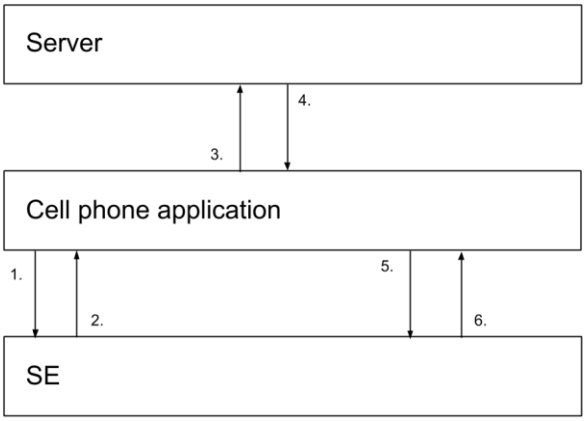

Figure 2. Server synchronization

1. The cell phone application gives the “read transaction” order to the SE;
2. The SE contains in its response the following:
   - content of the transaction log;
   - the content HASH of the log, signed with the private key;
3. The cell phone application uploads the following data to the server:
   - content of the transaction log;
   - transaction log content HASH by the SE;

- the public key signed by the CA (nTicket Certification Authority) and found in the SE

4. After the server checked the signature of the incoming data and saved them, it returns the following:
   - the current server time;
   - the number of the records not to be deleted;
   - the concatenation of the below data signed by its own private key: current server time; number of the records not to the deleted from the log; the HASH of the transaction log signed by the SE;
   - its own public key and its signature by the CA
5. The received data are forwarded to the SE by the cell phone;
6. The SE conductor uses the above data to read the value of the server synchronization stamp.