Trusted Access Control System for Smart Campus

Mohamed Amine BOUAZZOUNI ¹, Emmanuel CONCHON ², Fabrice PEYRARD ¹ and Pierre-François BONNEFOI ²

¹University of Toulouse; INP; IRIT
²University of Limoges, XLIM

Workshop on Smart and Sustainable City (SSC) 2016
1 Introduction and Context

2 Trusted Execution Environment

3 OP-TEE

4 Identity Based Encryption

5 Our proposition
Introduction and context

- NFC-based contactless cards.
- Many vulnerabilities allowing cloning.
- No user authentication.
Dematerialization

Solution and definition

Dematerialization means replacing the physical card with a smartphone application that provides at least the same features and security level as this physical card. This process allows us to:

- Save money since no physical card is needed.
- Deploy more complex security and authentication mechanisms (user authentication for instance).
- Combine several features into one secure mobile application.
Dematerialization of contactless cards (2)

Smartphone acts like a regular card

Card data
Access Rights

Dematerialization process
Dematerialization vs Regular cards

Dematerialization

- No need for a physical smartcard.
- Ability to deploy more complex algorithms.
- Mutual authentication between the user and the reader.
- More storage capacity for different applications.

Regular cards

- A physical smartcard is needed.
- The processing power is limited.
- No user authentication can be performed.
- Limited storage capacity -> limited number of applications.
Problematic : How to secure data in the mobile world?

The native OS of a mobile is not a suitable environment to deploy our secure application. Indeed:

- The mobile OS is an untrusted software, we have to consider it as an enemy.
- Many mobile applications are harmful for personal data on a mobile.
- The I/O operations are not secure.

Requirement

These potential vulnerabilities on the native OS led us to use a Trusted Execution Environment.
Abstract Architecture

1. Decentralized storage of the credentials.
2. Key exchange to perform challenge/response authentication.
3. Identities can be used as encryption keys.
4. Communications need to be secure.
A Trusted Execution Environment (TEE) is a secure environment allowing the secure storage and processing of sensitive data. The characteristics of such an environment are:

- **Isolated execution**: Every application should run independently from the other applications.
- **Secure storage**: Guarantee the integrity and secrecy of all data including the binairies (applications).
- **Secure provisioning**: The ability to install securely applications and to send securely data to a specific application.
**Introduction and Context**

**Trusted Execution Environment (TEE)**

**OP-TEE**

**Identity Based Encryption**

**Our proposition**

**TEE features**

- A trusted part of the processor or an entire processor dedicated to execute sensitive operations.
- A secure OS with a minimal set of instruction to allow a formal verification of its security level.
- A secure boot to verify the mobile OS and load sensitive modules.
TEE features

- A trusted part of the processor or an entire processor dedicated to execute sensitive operations.
- A secure OS with a minimal set of instruction to allow a formal verification of its security level.
- A secure boot to verify the mobile OS and load sensitive modules.

To deploy an application on a TEE, a developer should have the agreements of the manufacturer.
Motivation

The major obstacle to the use of the TEE is that the secure OS needed is proprietary and any application we want to execute on it has to be signed by the Secure OS provider.

OP-TEE:

- was designed to propose to developers an open source secure OS.
- can be executed on a physical device (ARM Juno).
- can be executed in a virtual environment.
OP-TEE components

- Two main parts: Client Application and Trusted Application.
- An internal API to facilitate the development and the communication between the two parts.
- Isolation between the Client and the Trusted Application.
OP-TEE components

- Two main parts: Client Application and Trusted Application.
- An internal API to facilitate the development and the communication between the two parts.
- Isolation between the Client and the Trusted Application.
Identity Based Encryption (IBE)

**Definition**

The Identity Based Encryption mechanism is based on the feature of a cryptographical system to use an identity as a public key. Then, no need for certificate to authenticate the owner of a public key.

- The identity is used by the cryptographical system for ciphering and generating the private key.
- The process need to be deterministic.
- The IBE should be at least as secure as a classical asymmetric encryption.
Identity Based Encryption: Online and Offline IBE (2)

- This is a full online solution.
- Removing the step 3 led us to an offline solution.
- Online vs Offline: no one wins.
This is a full online solution.
Removing the step 3 led us to an offline solution.
Online vs Offline: no one wins.
Final Authentication Architecture

1. A Decentralized OP-TEE server (Virtualized or Hardware) hosts an IBE system.
2. A protocol need to be designed to perform the challenge/response authentication.
3. The identities are used in an IBE to encrypt/decrypt messages.
4. The communications need to be secure (SSL-based solutions for instance.)
Challenges

To design and implement such an architecture, it is necessary to overcome some challenges:

- **Identity management**: In an IBE-based system, the management of the identities is critical.
- **User authentication with the server**.
- **Deal with the NFC timeout**: The NFC communication has a timeout of 5 ms.
- **Scaling problem**: Deal with thousands of simultaneous users.
Authentication Protocol

User

Online IBE

Reader

Request IDR

Response IDR

Request(C, IDR, IDU)

C' = E(C, IDU)

Send C'

Send C'

R = D(C')

R' = E(R, IDR)

Send R'

Send R'

Send R'

Compare R with D(R')

Grant access or
do not grant access

Bouazzouni et al.
First Results

**Encryption key generation: RSA vs IBE**

- **RSA**
- **IBE**

**RSA vs AES encryption time**

- **AES**
- **RSA**

Key length in bits:
- 512
- 1024
- 2048

Encryption time in seconds:
- 0
- 0.5
- 1
- 1.5
- 2
- 2.5
- 3
- 3.5
- 4

Data length in Bytes:
- 0
- 1024
- 2048
- 4096