SI5/M2II - IoT Security - September 16th, 2024
Yves Roudier - UCA / Polytech Nice Sophia
Smart Card Project Description

In this project, you will have to develop a terminal + a smart card supported system, which should implement an electronic purse for vending machines. This implementation will be a little more secure than a plain counter-based e-purse in the sense that we want transactions to be verified by a trusted back-end server in case the card gets stolen or if someone tries forging a YesCard as well as to protect from attackers pretending to be regular vending machines. The e-purse should also be working with different vendor machines.

The terminal application will send information about the transaction to the smart card. It is possible to generate an RSA private and public key pair within the smart card, which we will use to sign the transactions. The signed transaction will be logged at the back-end.

1) The smart card applet

You first have to implement a JavaCard applet that should support the following functionalities.

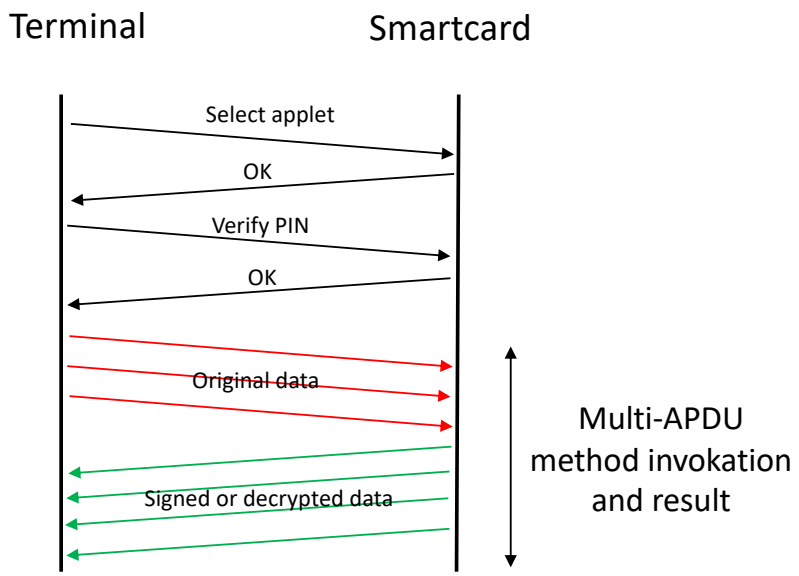
At installation time:

- The card should be initialized with a PIN code that will be provided by the end-user in order to prevent its usage in case of theft
- The card will have to generate an RSA 512-bit key pair whose public key will be registered by a verification server (simulating your bank) at initialization time. You will have to implement such a minimalistic server. The server will be in charge of verifying that the card has not been reported stolen, as an additional protection measure.
- The verification server will be equipped with an RSA 512-bit key pair, and its public key will be deployed onto the card.

At runtime :

- The vending machine application will send the card the description of the goods being bought. The card will sign and encrypt these data (transmitted by the terminal-side application) on demand, thus playing the role of a minimalistic TEE. The vending machine will have to check that the signature is correct together with the verification server (independently from the card)
- The card will also send the IP address of the verification server upon request of the vending machine in order to accommodate different banks depending on the end-user (you have to decide whether you want this signed).
- The signed and encrypted message will be sent to the verification server, where buys can be listed with the private key. To do so, the verifier server will be used as a terminal-side application itself (meaning that the listing of goods will only be available when the card is plugged into the reader, because only the card will be able to decrypt this list. The logs will not be kept within the card.

Because the messages exchanged with the card and containing the description of goods might be bigger than what can be carried within an APDU, you will have to implement the a small multi-message protocol which should behave according to the following graphical depiction:



The data carried by the multiple APDUs will have to be reassembled if too large to be sent over a single APDU. This will be used by the *pay()* and *decryptLog()* methods implemented in the card.

2) The terminal side applications

You will have to implement two standalone applications (the vending machine and the verification server) running on your computer, that is, on the terminal talking to the smartcard.

You can write these applications in Java or in Python, using one of the approaches outlined in the lab document.

These applications will need to interact with the end-user to request his PIN.

The verification server will be used to send the smartcard an authenticated time and date of the transaction (remember the card is not equipped with a clock). This will be used as a protection mechanism against rogue vending machines.

3) Report and Results

You have to provide:

- a small report describing the architecture you adopted and the cryptographic protocols defined between the three parties (you must argument the rationale behind your design).
- The report should also explain how you tested your development. It is suggested to test your applet methods using interactive APDU messages.
- The applet should be loaded onto your Javacard.
- You will have to provide your terminal-side applications and a small documentation. Make sure that they can be started from the command-line and don't forget to provide any dependencies that may be required.