E-Purse System

February 2019

1 Introduction

Design and implement an electronic purse à la Chipknip. This application consists of:

- an E-Purse applet,
- a reload terminal and,
- a POS (point of sale) terminal.

The balance of the E-purse applet can be credited at the reload terminal only. In a store the card can be debited at a POS.

The card should authenticate the reload terminal (only "valid" reload terminals are allowed to increase the balance). The POS should authenticate the card (payment is only possible with "valid" cards). When the keys on one card are compromised, the system as a whole should still be safe.

2 Security Requirements Engineering

In this section the results of the security requirements engineering for the electronic purse system are presented.

2.1 Use-cases

In the operation of the entire system a number of use-cases will occur. In the next sections, the use-cases, involved parties and goals are described.

2.1.1 Personalizing

This use-case involves the issuer and the card. In this use-case the issuer will make a card ready to be used for some specific user. To this end the issuer will generate and upload some unique and shared key material and certificates. So that the card can be used securely by the user.

2.1.2 Payment

This use-case involves a Point of Sale (POS) terminal, a smart card and the user. In this use-case the user wants to do some payment. To this end the smart card and terminal will create a record of the transaction and change the balance on the card.

2.1.3 Reload

This use-case involves a reload terminal, a smart card and the user. In this use-case the user wants to increase the balance on his card. To this end the reload terminal will increase the balance on the users card, after the user has paid for it.

2.1.4 Blocking a card

This use-case involves a user, who has lost its card. In this use-case the user wants to disable his card, so no futher transactions can be done with the card. The card will be disabled once it connects to an PoS terminal.

2.2 Assets

During operation of the system some assets must be secured, to guarantee correct operation of the system. These assets are:

1. Purse balance: There should not be any illegal modifications to the balance stored on a card.
2. Masterkeys: The masterkeys used for generating certificates whould remain secret.

2.3 Stakeholders

1. Merchants accepting payment by e-purse. Merchants accepting payments by the e-purse system will impose requirements on the system. For example the will want that customers cannot denied having payed. But also users want merchant to be unable to forge payment proofs.
2. Users paying with e-purse.
3. Banks supporting e-purse payments

2.4 Attacker model

In this section the attacker model is defined. This model describes exactly what the attacker is able and not able to do. The designed e-purse system should be secure in the presence of this attacker. If the system is insecure for some aspect, then the risks of this should be determined and accepted.

The attacker model for this design is based on the Dolev-Yao model[1] extended with the constraints given by the course[2] and considers malicious users. Therefor the following is assumed:

- The attacker has full control over the network between the card and a terminal:
  1. The attacker can block messages send on the network,
  2. The attacker can inject messages on the network,
  3. The attacker can replay recorded messages on the network.
- Perfect cryptography:
  1. The attacker cannot break the used cryptography,
  2. The attacker cannot predict nonces,
  3. The attacker cannot invert one-way functions.
- Tamper resistant cards:
  1. The attacker cannot install additional software on the card,
  2. The attacker cannot modify software of the card,
  3. The attacker cannot modify memory contents of the card.
- The attacker may compromise a card and retrieve the keys stored on the card using a side-channel attack. But only after acquiring the card, e.g. the attacker cannot do this during a transaction, but may do this for a card he owns.
- The attacker may acquire multiple cards for the same identity.

2.5 Trust Assumptions

To create an operational system capable of satisfying the security requirements, a number of processes, items and people need to be trusted.

To ensure the system remains secure, all personal involved with creation, usage and storage of the master keys must be trusted. Leakage or misuse of these keys trivially renders the system insecure.

3 Security Requirements

R_1: After personalization key material on cards cannot be changed.

R_2: No protocol should leak any secret key or information.

R_3: A terminal must require a smartcard to authenticate itsself.

R_4: A smartcard must require a terminal to authenticate itsself.

R_5: The integrity of data exchanged between the terminal and smartcard must be guaranteed.

R_6: The authenticity of data exchanged between the terminal and smartcard must be guaranteed.

R_7: The freshness of data exchanged between the terminal and smartcard must be guaranteed.

R_8: Completed payments must have non-repudiation

R_9: Payments should only be

R_{10}: Compromise of a single card, should not affect the entire system.

R_{11}: SmartCards can be blocked.

R_{12}: After blocking a card, no terminal should engage in any protocol with the card.

4 Protocols and Design Decisions

Protocol 1 Authentication Protocol

Goal: Perform authentication for the smart card

The protocol:

$C \rightarrow T : Initialize$

$T \rightarrow C : \{C, N\}$

$C \rightarrow T : \{|\#(Pin, T, N)|\}Sc$

Validate Nonce
$T \rightarrow S : \{|\#(Pin, S)|\}St$

Send the message to the server
$S \rightarrow T : \{|\#(Result, T)|\}Ss$

Result of Pin validation (approved in this case)
$T \rightarrow C : \{|\#(Result, C)|\}St$

~~Result of Pin validation (approved in this case)~~

Protocol 2 Reload Protocol

Goal: Increasing the balance on the smart card.

The protocol:

$Terminal \rightarrow SmartCard : test$

Protocol 3 Payment Protocol

Goal: Perform an payment with the e-purse and produce a proof of payment.

The protocol:

$Terminal \rightarrow SmartCard : cert$

$C \rightarrow T : Initialize$

$T \rightarrow C : N$

$C \rightarrow T : \{|(Pin, T, N)|\}Sc$

$T \rightarrow S : \{|(Pin, S)|\}St$

validate Nonce and send the message to the server $S \rightarrow T : \{(Result, T)$

Ss}Result of Pin validation (approved in this case) $T \rightarrow C : \{(Result, C)$

St}Result of Pin validation (approved in this case) $C \rightarrow T : \{|(Pin, T, N)|\}Sc$

$Terminal \rightarrow SmartCard : cert$

$Terminal \rightarrow SmartCard : cert$

$Terminal \rightarrow SmartCard : cert$

Protocol 4 Personalization

Goal: Upload the key material and certificates, and disable reinitialization of the card.

The protocol:

$Terminal \rightarrow SmartCard : -$

4.1 Key Distribution

4.2 Payment Protocol

4.3 Reload Protocol

5 Life Cycle

5.1 Persistent Life-cycle

5.2 Transient Life-cycle

References

[1] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[2] Erik Poll. Javacard project. `http://www.cs.ru.nl/~erikpoll/hw/docs/javacard_project.pdf`. Accessed: 16-02-2019.