E-Purse System Design Document

Group 14

1 Introduction

This paper proposes an electronic purse à la Chipknip system. It is intended for transactions sessions, with amounts either credited or debited to/from the card. It describes the protocol used and the design, taking into consideration the security aspects involved by such a system. The system consists of:

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

2 Functional Requirements

R_1: The system shall allow the user to reload the smart card balance, by crediting at the reload terminal.

R_2: The system shall allow the user to do a payment, by debiting the smart card balance at a POS.

R_3: The system shall provide smart card personalization.

3 Security Requirements

R_4: Key material on cards cannot be changed after personalization.

R_5: Secret key or information should not be leaked.

R_6: A POS must require a smart card to authenticate itself.

R_7: A smart card must require a terminal to authenticate itself.

R_8: Expired cards should not be used for performing transactions.

R_9: The integrity of data exchanged between the terminal and smart card must be guaranteed.

R_{10}: The confidentiality of data exchanged between the terminal and smart card must be guaranteed.

R_{11}: The authenticity of data exchanged between the terminal and smart card must be guaranteed.

R_{12}: The freshness of data exchanged between the terminal and smart card must be guaranteed.

R_{13}: Completed payments must have non-repudiation.

R_{14}: Credit and debit transactions must be logged on the smart card.

R_{15}: Compromise of a single card, should not affect the entire system.

R_{16}: The card should block after 5 unsuccessful PIN attempts.

R_{17}: After blocking a card, no terminal should engage in any protocol with the card.

4 Security Requirements Engineering

This section addresses the results of the security requirements engineering for the electronic purse system.

4.1 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.

4.1.1 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 load individual card holder data, shared key material and certificates, so that the card can be used securely by the user.

4.1.2 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 the terminal will create a record of the transaction and change the balance on the card.

4.1.3 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.

4.1.4 Blocking a card

This use-case involves the owner of the card and the issuer. There are two situations when a card can be blocked: the user had his card lost or stolen, or there have been more than 5 unsuccessful PIN attempts. In case the user has the card missing, he wants to disable his card, so no further transactions can be done with the card. The card will be disabled once it connects to a

POS terminal. In case the PIN has failed more than 5 times, the card will be automatically blocked. This most likely happens when the card has been stolen and someone is brute forcing the PIN.

4.2 Assets

During operation of the system some assets must be secured, to guarantee correct operation of the system. The following assets should be taken into consideration:

1. E-purse or the card.
2. Communication terminal: The mediator terminal between card and the server.
3. Server: Handling the back end tasks.
4. Reload terminal: Allowing the owner to reload the E-purse.
5. Secret and public keys: Keys for the server, terminals and cards.
6. Digital certificates: To only allow legitimate devices to communicate in the system.
7. Loyalty points: Act as credit for customers that can be used to pay.
8. E-purse balance: There should not be any illegal modifications to the balance stored on a card.
9. Master keys: The master keys used for generating certificates would remain secret.

4.3 Stakeholders

1. Merchants accepting payment by e-purse. Merchants accepting payments by the e-purse system will impose requirements on the system. For example, merchants want customers not to deny having payed and also users want merchants to be unable to forge payment proofs.
2. Users paying with e-purse.
3. Banks/ card issuers supporting e-purse payments.

4.4 Attacker model

This section defines the attacker model. 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, by detecting and reacting to the attack.

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. Therefore, 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.

4.5 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.

5 Protocols and Design Decisions

In this section following notations are used:

1. C: Card
2. T: Point of Sale Terminal
3. RT: Reload Terminal
4. PT: Personalization Terminal
5. P: Bank POS card reader

6. S: Server
7. SP: Sales person
8. Cid: Card identifier
9. M: Message
10. N: Nonce

Protocol 1 Authentication Protocol

Goal: Perform authentication for the smart card

The protocol:

$C \rightarrow T : Initialize$

Card initializes the connection.

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

Terminal sends the nonce to the card and asks for the Pin.

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

Terminal validates the Nonce and send the card information to the server.

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

Terminal sends the Pin to the server.

$S \rightarrow T : \{|Result, T|\}Ss$

Server validates the card id and its Pin, sends the result of to the terminal.

$T \rightarrow C : \{|Result, C|\}St$

Terminal sends the result to the card.

Protocol 2 Reload Protocol

Goal: Increasing the balance on the smart card after successful authentication.

The protocol:

$C \rightarrow RT : \{Requestforbalanceincrease\}$

Card initialize the connection with the reload terminal.

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

Reload terminal sends Nonce, asks for the amount and signature of the card.

$C \rightarrow RT : \{|\#(Cid, amount, N, RT|\}Sc$

Card signs the message and sends to the reload terminal.

$RT \rightarrow POS : \{amount, POS\}$

Reload terminal sends the request for the desired amount transaction to the POS device. We will omit the transactions between POS and the bank as it can be considered as an external one.

$POS \rightarrow RT : \{|result, RT\}$

POS device sends the result to the reload terminal and if successful it will go to the next stage, if not it will alert the user about unsuccessful transaction.

$RT \rightarrow S : \{|Cid, amount, S|\}Srt$

Terminal sends the Cid and the amount to the server to increase the balance.

$S \rightarrow T : \{|Result, T|\}Ss$

Server sends the result of the successful operation to the reload terminal. The amount will be added to the card balance and it will be logged in the transaction history and card history.

$T \rightarrow C : \{|Result, C|\}St$

Terminal sends the result to the card.

Protocol 3 Payment Protocol

Goal: Perform a payment with the e-purse and produce a proof of payment after successful authentication.

The protocol:

$SP \rightarrow T : \{amount, T\}$

Sales person sends the amount to the terminal.

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

Terminal sends Nonce, asks for the signature of the card.

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

Card signs the message and sends to the terminal.

$T \rightarrow S : \{|amount, Cid, S|\}St$

Terminal sends the card id and the amount to the server to check for the sufficient balance.

$S \rightarrow T : \{|Result, T|\}Ss$

Server sends the result of the successful (or unsuccessful) transaction based on the sufficient balance. The amount will be reduced from the card balance and it will be logged in the transaction history and card history.

$T \rightarrow C : \{|Result, C|\}St$

Terminal sends the result to the card.

Protocol 4 Personalization

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

The protocol:

$SP \rightarrow PT : Initialize$

Sales person initialize the connection with the personalization terminal.

$PT \rightarrow S : Request$

Personalization terminal asks the server for the required information to initialize new card that consisted of Cid, card secret key and required public keys and digital certificate.

$S \rightarrow PT : \{|M, PT|\}Ss$

Server removed the dedicated id from the database, updates required tables with the new card information and sends the required information (M) to the personalization terminal.

$PT \rightarrow C : \{M\}$

Personalization terminal puts the received information from the server to the new card.

$PT \rightarrow SP : Result$

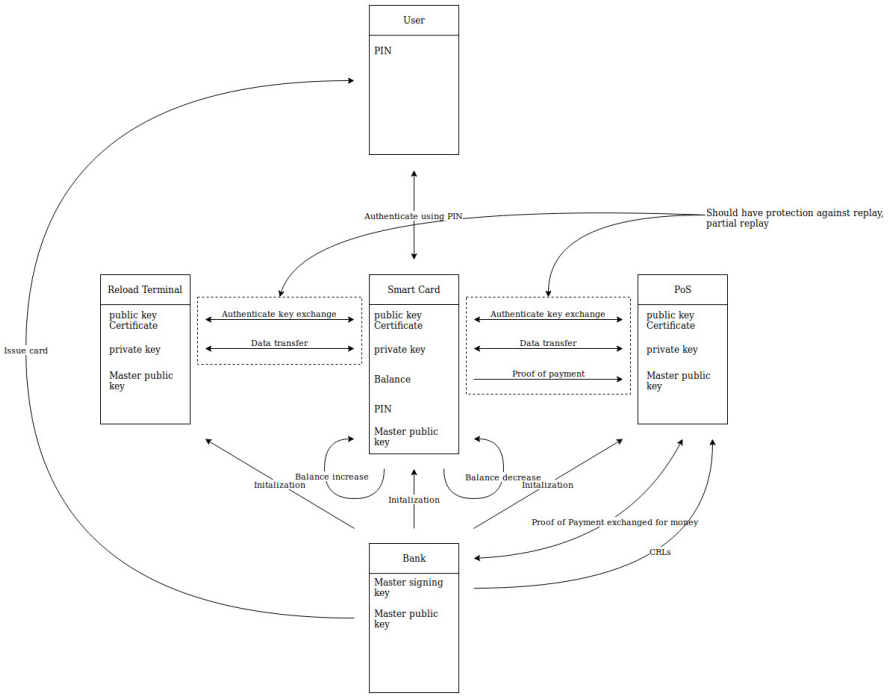

Figure 1: The e-purse system

Personalization terminal makes the sales person aware of the result.

5.1 Key Distribution

The smart card, the POS and the reload terminals all holding their own private key and also the master public key. Figure number 1 presents the entire system as a whole, highlighting also the distribution of keys.

6 Life Cycle

This section presents the life cycle of a card, both persistent and transient.

6.1 Persistent Life Cycle

The persistent life cycle involves several steps, presented in figure number 2: initialization, personalization, performing transactions - reloading (crediting) or debiting - and finally blocking it at user's request or because it is expired. Specifically, using raw cards and a personalization terminal, the sales represen-

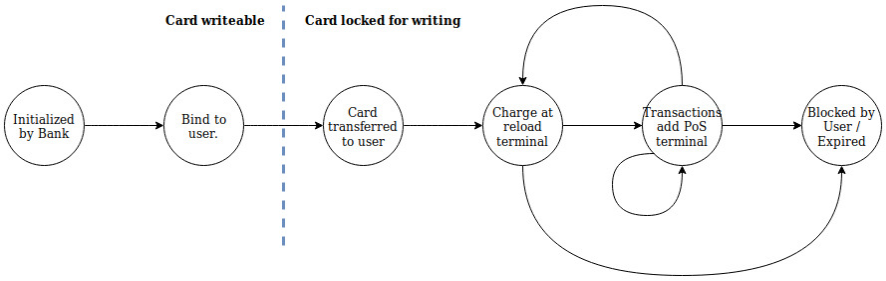

Figure 2: The persistent life cycle of the card

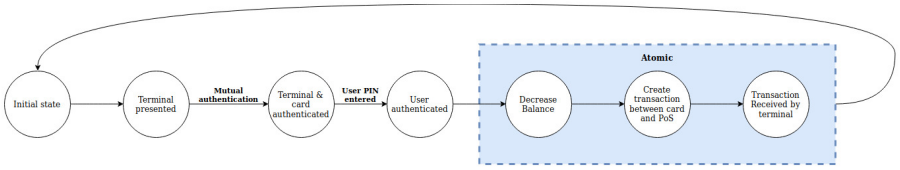

Figure 3: The transient life cycle of the card

tative will personalize the card for the customer. This involves writing files and application data to the card. After the database on the server gets updated through this procedure, the new card information is recorded and also logged. The card is now bound to the user.
Cards are retired after 5 years from the issuance date. No transactions will be possible using the expired cards, except the read operation, performed by the sales representative for analysis purpose.

6.2 Transient Life Cycle

The transient life cycle steps are presented in figure number 3, involving authentication of both parties (terminal and card) and atomic transactions, causing decrease or increase in the card balance.

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.