NFC Report

Tuomas Aura

Oksana Baranova (890401) & Arthur Carels (918147)

December 29, 2020

This report discusses the design of a multi-ride amusement park ticket on an NFC memory card application. For this exercise, a smart card ticket application for a small operator that uses cheap blank smart cards as the ticket medium is used. The basic idea is the following. The application should be able to (re)format the card, issue new tickets and top up the card with additional tickets. Card validation is done with an NFC-enabled Android device and is only possible if the card has not expired yet.

In the first part, the memory lay-out of the nfc card is discussed. The second part shows two flowcharts that illustrate the sequence of steps taken when a user clicks on the *issue* or the *validate* button. The third and last part of this report focuses on the measures that have been taken to ensure security.

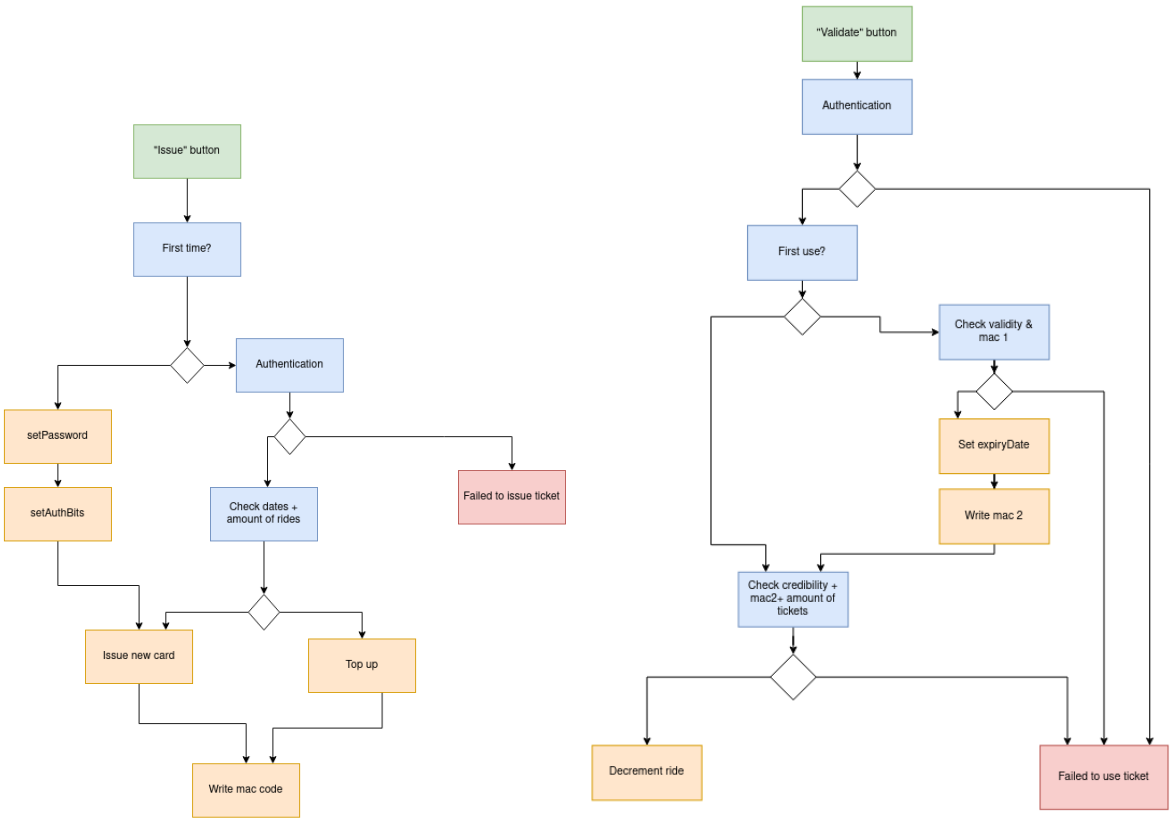
1 Lay-out

This assignment uses a NXP MIFARE Ultralight C smart card. It contains 48 pages of 4 bytes which totals 192 bytes of memory. This card is not programmable, but more of a secure memory card. The data sheet can be found at `https://www.nxp.com/docs/en/data-sheet/MF0ICU2.pdf`.
The memory lay-out of the card is represented in Table 1.

Table 1: Table containing page numbers and their content on the used smart card.

Paqe number	Content
1 - 3	UID + check bits
4	Application tag
5	Version Number
6	Initial counter
7	Ride counter
8 - 9	Validity date
10-11	Expiry date
12	MAC_1 (truncated)
13	MAC_2 (truncated)
14 - 39	Empty
40	Lock bits
41	16bit one-way counter
42	auth0
43	auth1
44 - 47	authentication key

2 Program flow



(a) The program flow when the *issue* button is pressed.

(b) The program flow when the *validate* button is pressed.

Figure 1: Simple representation of the program flows when the *issue* and *validate* buttons are pressed.

In this section, the program flow when the *issue* button, see Figure 1a, and the program flow when the *validate* button, see Figure 1b, are pressed is listed.
Note: these figures are simplifications and they are mainly meant to help in understanding the big *structure* of the program.

3 Security Measures

This section talks about all the security measures taken to make the NFC tickets as safe as possible.

- Upon issuing the card, a new password is set. This password is unique for every card, which prevents large-scale security failures caused by breaking one card. To obtain a unique password, the following is done: *key K = hmac(master secret | UID)*.
- Authentication is done by built-in-shared-key authentication (3DES cryptography).
- AUTH0 and AUTH1 bits have been set. This prevents reading and writing the memory contents of the card without authentication. Setting reading protection (AUTH1 parameter), makes debugging harder for officials, but it ads a protection against *replay* attacks. Setting this bit has been discussed during the demo.
- A MAC code is set to prevent MitM attacks. If an attacker changes any *sensitive* page, this will be noticed because of 2 different MAC codes.
  This MAC is quite short to keep the amount of pages to write as small as possible. The *full* MAC code is truncated to be only 4 bytes long. Even with this length, it would take the attacker on average 2^{31} tries to make a malicious and unnoticed change to the card's contents.
  (**Note:** instead of taking a mac over the UID as suggested in the paper, we based the MAC password on the UID. This is done with an approach similar to the construction of the unique password.)
  (**Note:** We make a difference between *Validity date* and *Expiry date*!)
- The MAC code is not updated during normal operations to prevent tearing. We use two different MAC codes: MAC_1 and MAC_2. MAC_1 is written when issuing a card or when topping up an unused card. In this case, no expiry date has been set yet. When first using the card, the expiry date is set and a new MAC (i.e. MAC_2) has to be written to include this one. The counter is updated as the last step in this process to prevent tearing.
- Cloning to a different card is impossible for two reasons:
  - You cannot read card contents without authentication (because of AUTH bits).
  - Even if this worked, the password would be wrong because of the different UID.
- Safe limits are set & checked on the amount of cards that e.g. can be issued at the same time.
- We use a counter which can never decrement (= unary counter). This ensures rollback prevention
- We implemented pass-back protection using the following approach: every time the card is validated, a valid ticket is subtracted.
- All the info necessary for validating tickets is available on the card. This enables the ticket vendor to continue work even in the case of an internet interruption.

- Reasons for ticket invalidity are not shown directly on the screen to prevent malicious users to gain knowledge of the reason what went wrong (could update their techniques to counteract the mistake then). During the demo it was mentioned that we should maybe consider showing a statement when the tickets have all been used, as this info is not useful for attackers. It would however be of much use for legitimate users.
  However, all errors are logged in the console so that the ticket vendor can see what went wrong for logging purposes

4 Conclusion

This report detailed the design of a secure multi-ride amusement park ticket using an NFC card. Concepts such as oneway counter, MAC codes, unique passwords,... have been used to make the cards as secure as possible.