Network Security Project 1: NFC Ticket App

Group 36: YANG Je−Ruei, XU Zehui

Table of Content

1 Ticket App Features

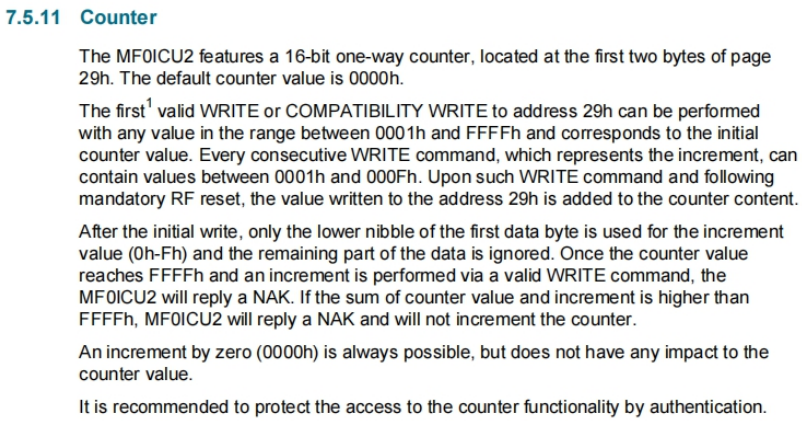
- 5 rides are issued everytime
- When using the ticket, we check integrity, expiry time and remaining rides, and then increment the counter thus reducing remaining rides
- Each ticket uses its own key derived from the master key and UID
- The tickets are valid for one day (86400 seconds) from the time when they were used for the first time
- Start the validity period only when the ticket is used for the first time
- If the tickets have expired or they have been fully used, cards are reformatted and then issued with new tickets (savings on blank tickets)
- Issue 5 more rides to a card without erasing any valid ticket
- A cooldown time of 2 seconds to prevent double tap

2 NFC Card Memory Layout

Page	*Byte number*			
	0	*1*	*2*	*3*
4	*max counter*			
5	*init counter*			
6	*expiration time*			
7	*HMAC*			
41	*counter page*			
42	*auth0*			
43	*auth1*			

3 Implementation Details

1. We utilize the built−in 16−bit one−way counter of the MIFARE Ultralight C to record the number of used tickets. We record the initial value and the maximum counter value at the time of issuing, as a safeguard against rollback attacks.



7.5.11 Counter

The MF0ICU2 features a 16-bit one-way counter, located at the first two bytes of page 29h. The default counter value is 0000h.

The first[1] valid WRITE or COMPATIBILITY WRITE to address 29h can be performed with any value in the range between 0001h and FFFFh and corresponds to the initial counter value. Every consecutive WRITE command, which represents the increment, can contain values between 0001h and 000Fh. Upon such WRITE command and following mandatory RF reset, the value written to the address 29h is added to the counter content.

After the initial write, only the lower nibble of the first data byte is used for the increment value (0h-Fh) and the remaining part of the data is ignored. Once the counter value reaches FFFFh and an increment is performed via a valid WRITE command, the MF0ICU2 will reply a NAK. If the sum of counter value and increment is higher than FFFFh, MF0ICU2 will reply a NAK and will not increment the counter.

An increment by zero (0000h) is always possible, but does not have any impact to the counter value.

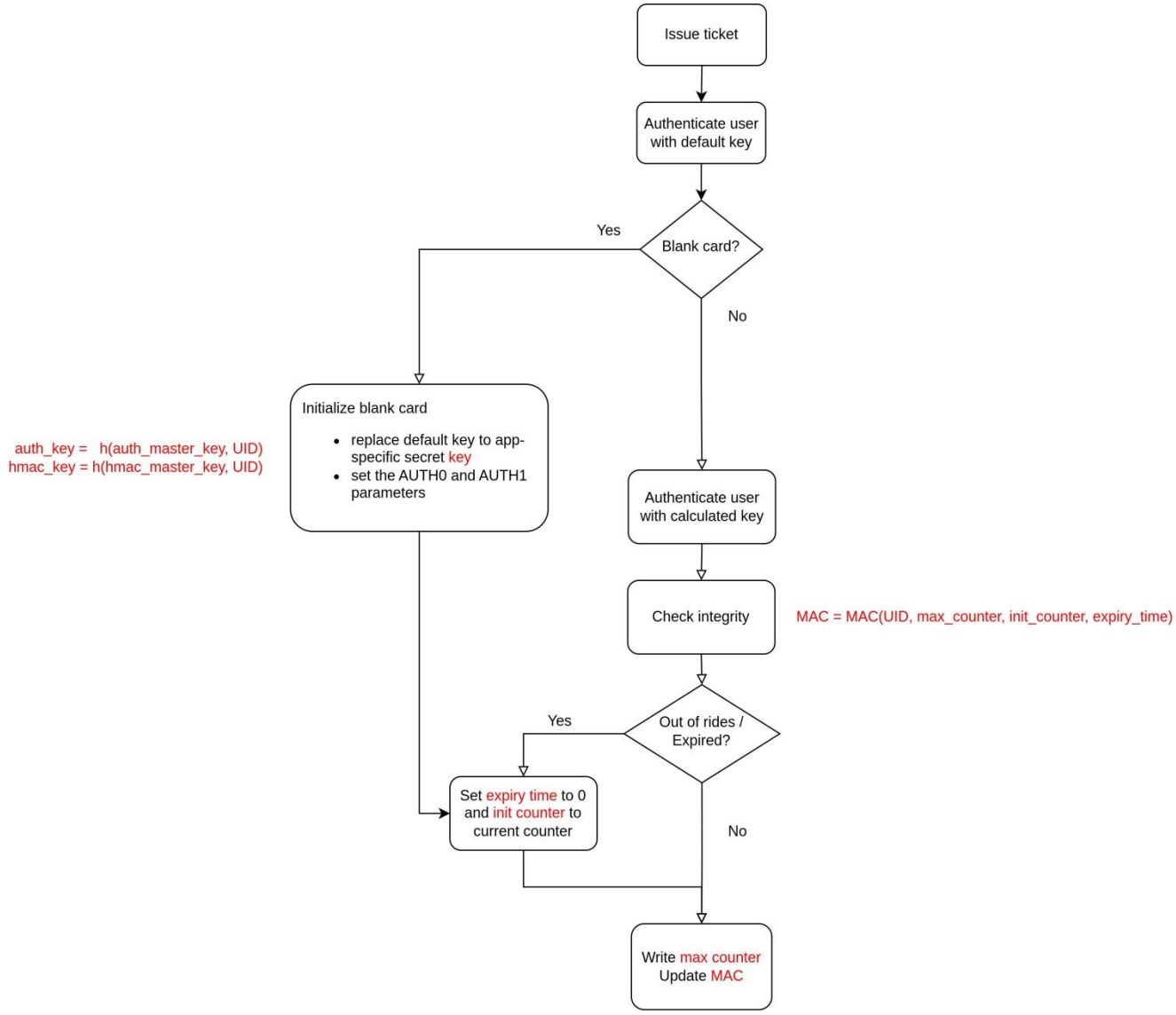
It is recommended to protect the access to the counter functionality by authentication.

2. During the card issuing and first use, we set the data (maxCounter and expiryTime) which remains unchanged during regular use. This is done as a measure against replay attacks.
3. The card authentication key and HMAC key is calculated using the slow hash algorithm PBKDF2WithHmacSHA512(master key, UID). This helps prevent brute force attacks while ensuring unique keys for each card.

```
private byte[] generateKey(byte[] uid) {
    byte[] key = new byte[KEY_LENGTH];
    PBEKeySpec spec = new PBEKeySpec(new String(authenticationKey).toCharArray(), uid, iterationCount: 1000, keyLength: 256);
    try {
        SecretKeyFactory keyFactory = SecretKeyFactory.getInstance( algorithm: "PBKDF2WithHmacSHA256");
        byte[] hash = keyFactory.generateSecret(spec).getEncoded();
        key = new byte[KEY_LENGTH];
        System.arraycopy(hash, i: 0, key, i1: 0, KEY_LENGTH);
    } catch (Exception e) {
        Utilities.log("getKey error", true);
    }
    return key;
}
```

4. To prevent double tapping, we record the timestamp of the last operation in our card reading device (Ticket App).
5. The ticket expiration time is set only upon first use and remains valid within the day (86400 seconds).
6. We record an HMAC of one page (4 bytes) to ensure message integrity. This HMAC is calculated from the UID and the writable data stored on the card (maxCounter, expiryTime, initCounter).

4 Issue flow

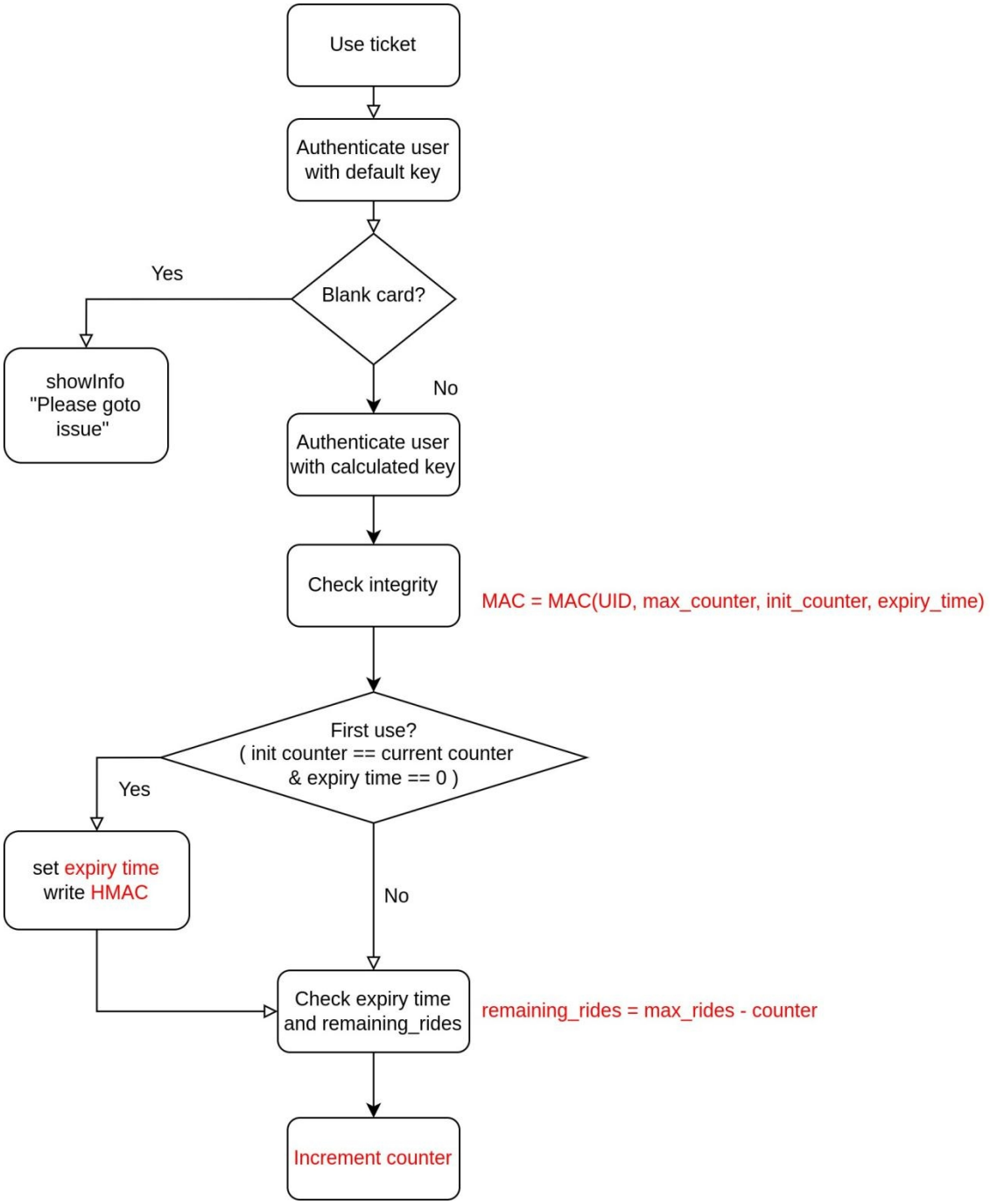


1. First, we attempt to authenticate it using the default key. If this is successful, it indicates that the card is a blank card. We then set a new auth_key and set auth0 and auth1. If authentication fails with the default key, it signifies that the card has already been initialized.
2. We compute the auth_key based on master_key and UID, then read the ticket data. The validity of the ticket data is subsequently checked based on the HMAC value.
3. When the ticket expires or is depleted, we reset the expiry time to zero and the init/expected counter to the current counter value at the time of issuance, which will be checked at the first use.
4. In a typical usage scenario, we issue additional ride counts to the card (max counter +=5)

without deleting any tickets that are still valid.

5. Based on the data written, we calculate the HMAC and write all data onto the NFC card.

5 Use flow



1. First, we attempt to authenticate it using the default key to check whether the card is blank. If authentication is successful with the default key, it indicates that the card is a blank card, then we advise the user to proceed to the card issuance point.

2. If it's not a blank card, we authenticate it using the calculated auth_key. After authentication, we read the ticket data from the NFC card, and use the hmac_key to calculate the mac value for data integrity verification.
3. If it's the card's first usage, meaning the initial counter value on the card matches the current counter value and no expiry time has been set (expiry time is 0), then we set the expiry time to be one day from now.
4. We check whether the ticket has expired or been used up. If so, we inform the user to go to the card issuance point.
5. Increase the counter (single use of the card).