Tuomas Blomqvist - Network Security assignment 5 – NFC reader

Card memory layout (first 10 pages, rest are empty):

Page addr	Byte 1	Byte 2	Byte 3	Byte 4
00h	UID0	UID1	UID2	BCC0
01h	UID3	UID4	UID5	UID6
02h	BICC1	INT	LOCK0	LOCK1
03h	OTP0	OTP1	OTP2	OTP3
04h	T	I	K	T
05h	Ticket expiry time in 32 bit integer format			
06h	Number of allowed rides in 32 bit integer format			
07h	HMAC (over first 6 pages) first 4 bytes of hash			
08h	HMAC second 4 bytes of hash			
09h				

java Main issue:
1) Checks whether it's formatted correctly with the right application ID.
2) Write expiry time in 4-byte timestamp to page 5 in minutes before Unix epoch.
3) Writes number of allowed uses in 4-byte integer to page 6.
4) Calculates HMAC-SHA2 over the data in pages 0-6 (while assuming OTP and LOCK bytes equal 0) using HMAC secret statically defined in TicketMac.java.
5) Writes first 64 bits of the 160bit HMAC into pages 7 and 8.

java Main use:
1) Reads all the cards data into memory.
2) Checks if the card is properly formatted and displays an error message if not.
3) Calculates HMAC-SHA2 over the data in pages 0-6 (while assuming OTP and LOCK bytes equal 0) and compares that to HMAC stores in pages 7 and 8. If the values match then HMAC is valid.
4) Check that current time is less than expiry time and decode currentUses from OTP bits. Uses are stored in integer calculated with (2^uses-1). Check if any allowed uses remaining (OTP decoded uses less than allowed on uses stored in page 6)
5) Flip next bit in OTP bits to increment amount of used rides

java Main reissue
Mostly same as issue. Rewrites pages 5-6 with new values and calculates a new MAC and writes it into pages 7 and 8. Has some sanity checks to disallow setting less allowed rides than used OTP bits.

Java Main lock
Locks pages 5-15 by writing 0xF0 into LOCK0 and 0xFF into LOCK1.

Card is limited to 32 rides maximum since there are 32 OTP bits available. Almost left it to 30 maximum because java integer overflows at 2^31 but wanted to see those all ones in the Card memory dump printout. Also when running the program from Eclipse, System.console() will return NULL and reissue command will not work since Eclipse does not provide the executed java application with a real console. Works perfectly when running from for example cmd.exe.
I chose to not include OTP bits in MAC and recalculate MAC after each use because I was told it was okay to assume an attacker cannot obtain an Ultralight-card with changeable UID.

Security

Counterfeiting tickets is prevented with 64-bits of HMAC-SHA2. Anyone wanting to create a new ticket has to know the HMAC secret key specified in the source code.
Copying the ticket is prevented by factory programmed unique and unchangeable UID's[1]. Anyone acquiring a blank card and copying card data could not be able to change the UID bytes and this would result in invalid HMAC.
Incrementing ticket count and modifying expiry time will also result in invalid HMAC.
Replaying old tickets is prevented by storing rides currently used in OTP bits. Once OTP bit is set it cannot be unset.
I chose to not lock pages by default. With write access attacker can only make the ticket unusable. Anybody can make the ticket unusable by setting all 32 OTP bits into one. No real added security with locking pages. A proper ticketing application would utilize a smartcard-based tag such as Mifare Classic or Desfire and prevent this with write keys.

[1] This is a false assumption. UID changeable ”magic mifare” cards are available