Exploring 49er ID Card's Security

https://www.youtube.com/watch?v=XlmZaW5pHng

Sagar Shah
Parag Mhatre
Himagiri Yathagiri
Sai Sravya Charugulla
ITIS 6010: Wireless Security
Weichao Wang
May 10, 2019

Description

The 49er ID card used by UNCC is used across the campus to provide a wide-range of services to the students on campus like access to buildings and rooms, purchase almost anything on campus, event access, CATS transit access, parking deck access, dining services, Student Health Center, etc. The card is the key to many of the essential services on campus, so it's only natural for someone (our group) to be attracted to test the security of the card. The goal of the project is to find as much information about the card as we can, and spark a discussion on ways to potentially use the card in any way that was not intended by its manufacturer and/or the university.

Original research plan

We broke up our project into three phases. Phase 1: gather as much information available about the card. FInd out all the technologies used and find areas to research. Phase 2: capture the data and identify anything specific we can target. We expected to research manufacturer documentation, wireless protocols, authentication protocols, and finding out university's implementation of the card. We also planned to learn the hackRF SDR. Phase 3: iidany vulnerabilities in the manufacturer's implementation of the card or UNCC's implementation of the card.

Relation to wireless security

The card can use two modes of communication namely magnetic and RFID. More specifically, the card uses the high-frequency NFC subset of RFID technology. Since there is communication happening on the wireless spectrum we believe there is value in exploring the security of this card. Wireless communication opens up the card to replay/relay attacks and perhaps even card cloning if there is enough data leaked over the air.

Potential Impacts

If we are successful in detecting and/or exploiting vulnerabilities in the UNCC student ID then based on the vulnerability found, it could disrupt everyday operations across all universities that use this type of card. The predecessor to this model MiFare Classic has already been cracked easily by using an average smartphone, so breaking the latest model (MiFare DESFire EV1) will probably have severe consequences for NXP semiconductors.
Students who rely on the card for everyday services such as dining services will be exposed to a huge risk of monetary loss if there is a malicious attacker nearby.. Universities such as UNCC will have to rethink and reimplement the way they protect access to sensitive areas of the campus and also prevent monetary loss from fraudulent transactions. NXP semiconductors will have to redesign and issue a new batch of cards and maybe subject to litigation.

Efforts Conducted

Preliminary research/Basic information

The card holds two numbers, the 800 student ID number and a 16-digit card number. The card number is used to link the user to the transaction and will change if the student is issued a new card. The card is manufactured by NXP Semiconductors and the model is MiFare Desfire EV1 (MF3ICD41). It operates at the 13.56MHz frequency. It uses the ISO/IEC 14443 Type A standard. It has an operating distance of 100mm, and the maximum data transfer speed of 848 kbit/s. Data integrity is ensured by 16/32 bit CRC, parity check, bit coding and bit counting.

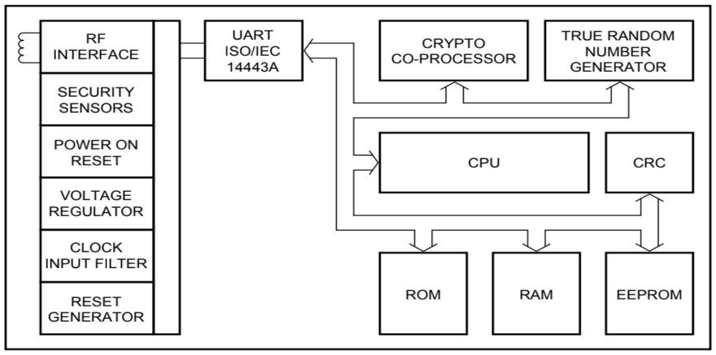


Figure 1: Block diagram of MF3ICD41

Security

Each card has a unique 7-byte UID which can't be overwritten. For encryption, the card uses the highly secure 3DES and/or AES encryption algorithms. Whenever the card is tapped on the reader, the card verifies the reader with a three-pass mutual authentication protocol. The following paragraph summarizes the three-pass authentication protocol between the reader and the card.

When AES authentication command is issued by the reader along with a key number, the card will know which AES key is used. The tag will select the AES key, generate a 16 byte random number (RndB), encrypt it with the selected AES key and transmit it. When the reply is received by the receiver, it decrypts the reply with the AES key, generates a new random number (RndA), left shift RndB by 8 bits and creates a new 32 byte value by concatenating RndA and RndB. The result is encrypted using AES key and transmitted to the card. When the card receives the command, it decrypts this command using AES key, splits the 32-byte value into equal halves (16-byte values) and rotates the RndB. If the received and the generated RndB (RndB') match, it means that the packet has been received successfully. The same procedure is followed by the RndA to generate RndA' when a reply is received by the receiver (RndB). Now the reader will decrypt the reply with the AES key using acquired RndA'. This RndA' can be generated by left shifting RndA by 8 bits and comparing them. If RndA and RndA' match, the authentication procedure is considered to be successful.

Basic read using Android phones

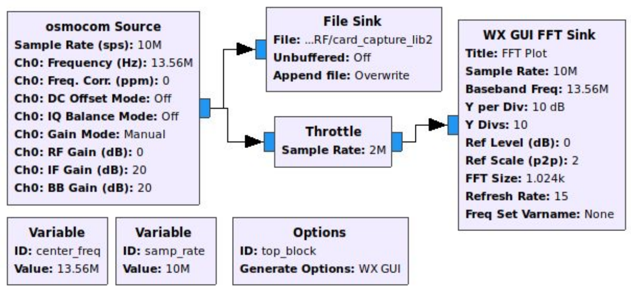
We started with the basics by using the NFC chip on our phone to extract information from the tag. We used the following apps from the play store: NFC Tools, MIFare Desfire Tool, and TagInfo. Other than basic information, TagInfo revealed some more details about what exactly is stored on the card. It seems that there are 6 applications on the UNCC card.

- Application 0x000000: Seems to contain PICC key configuration information
- Application 0x0112F2: 14 AES keys, 13 encrypted files, 1 plaintext file (CATS #)
- Application 0xBBBBBB: 1 AES key, 2 encrypted files
- Application 0xB2BBBB: 2 AES keys, 1 encrypted file
- Application 0xCCBBBB: 2 AES Keys, 1 encrypted file
- Application 0xDABBBB: 2 AES keys, 1 encrypted file

Learning hackRF/Captures/Car attack/Data Analysis

We used tutorials available on the internet to learn the basics of the hackRF. We installed the pentoo linux distribution, which comes with the built-in utilities required for a project like this. We used GNUradio-companion to interface with the hackRF. We used the xxd utility to generate a hex dump of the raw binary captures from GNUradio. We also used inspectrum to visualize the captures.

To get as much data as possible we captured several different types of actions that can be performed by tapping the card on a reader. We captured a balance inquiry transaction, a real transaction (cookie purchase) at the subway in cone. We also captured the communication that occurs when a card is tapped on a reader that grants access to restricted areas such as the lavender room in cone and the library main entrance after hours.



osmocom Source
Sample Rate (sps): 10M
Ch0: Frequency (Hz): 13.56M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

File Sink
File: ...RF/card_capture_lib2
Unbuffered: Off
Append file: Overwrite

Throttle
Sample Rate: 2M

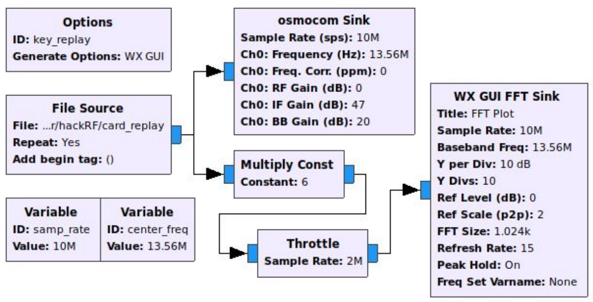
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 10M
Baseband Freq: 13.56M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

Variable
ID: center_freq
Value: 13.56M

Variable
ID: samp_rate
Value: 10M

Options
ID: top_block
Generate Options: WX GUI

Figure 2: 49er ID capture flow-graph



Options
ID: key_replay
Generate Options: WX GUI

File Source
File: ...r/hackRF/card_replay
Repeat: Yes
Add begin tag: ()

Variable
ID: samp_rate
Value: 10M

Variable
ID: center_freq
Value: 13.56M

osmocom Sink
Sample Rate (sps): 10M
Ch0: Frequency (Hz): 13.56M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 20

Multiply Const
Constant: 6

Throttle
Sample Rate: 2M

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 10M
Baseband Freq: 13.56M
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Peak Hold: On
Freq Set Varname: None

Figure 3: 49er ID replay flow-graph

Based on the time duration and file size of each capture it's safe to assume each sample is recorded in 64-bits. We set hackRF to capture 10 million samples per second. A 3 second capture should yield a 240MB file (3 seconds * 10M samples * 64-bits), and we confirmed that with a ~2.97 second capture that yielded a file size of 238.7MB.

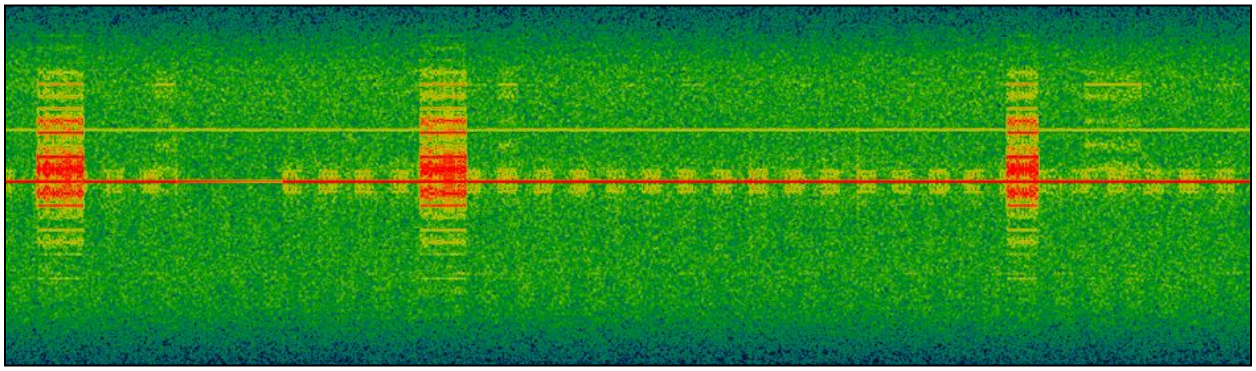


Figure 4: 49er ID - Subway balance inquiry capture (zoomed in)

We tried replaying the captured signal from the card and reader at all places where we captured our signal, but we were not getting anywhere. Therefore, we set up an easier experiment, we carried out a replay attack on a car with the capability of remotely unlocking a car. We also designed flowgraphs for the car key fob; however, we have omitted here to save space. We were successful in this experiment, which confirmed that we were using the hackRF correctly. We have included the working demo in the accompanying video. The following is a screenshot from inspectrum, the tool we used to visualize the captures. Frequencies: 312.1MHz, 314.35MHz.

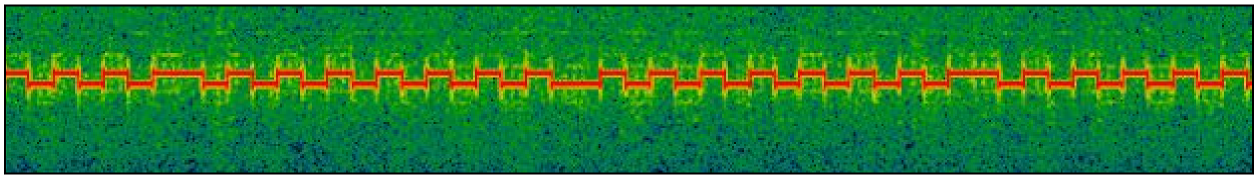


Figure 5: Car key unlock capture (zoomed in)

Technical difficulties and future

The 128-bit AES encryption algorithm used for encryption is still considered highly secure, so brute-forcing the keys stored on the card is not an option currently. The three-pass authentication protocol ensures that the encryption key nor the session key are ever in the air; therefore, the key cannot be sniffed via tools such as the hackRF SDR. Cloning the card is also unfeasible as each file stored on the card is encrypted with a different key. Furthermore, even if we were to obtain just one of the keys stored on the card, each i/o operation on each file requires a different key.

Final achievements

- Gained deeper understanding of RFID/NFC technology
- Understand technologies used by contactless smart cards

- How to use an SDR such as hackRF
- Generating GNUradio flowgraphs
- Visualizing binary capture of a signal

Deviation from original plan

We hit a brick wall after we captured the data as we couldn’t figure out how to process the data. The project could have been much more fruitful if we had the domain knowledge about signal processing. We tried demodulating and decoding the captured signal by importing the data into MATLAB but we were unsuccessful.

Future plans

We talked with some students from Electrical engineering to gain more insight into how the signal processing is done. We identified that the DESFire EV1 uses 100% Amplitude Shift Keying and Modified Miller encoding. The students suggested that decoding the raw signal captured from hackRF might require but not limited to getting a noise sample, filtering the signal, cleaning the signal, demodulating the signal. These steps are currently out of our scope, but we may continue in that direction in the future.

In future we could also take this to next step by studying and implementing the side channel attack to gain better understanding and also would go forward and use the publicly available cryptographic cloner created by the ruhr university researchers and then perform exploitation attacks on card using the information learnt through the cloner and side channel attack.

Team Member Roles

Sagar: Preliminary research, deploying hackRF, creating flow graphs, car replay attack, report.
Parag: Preliminary research, hackRF captures, producing video, reaching out to experts, report.
Sravya: Preliminary research, hackRF captures, car replay attack, analyzing the captures, report.
Himagiri: Preliminary research, learning hackRF/GNURadio, video production, report.

YouTube Link

https://youtu.be/XlmZaW5pHng

Note: Due to the large size of the raw binary captures we have not attached them to the submission; however, we can provide them if they are required..

References

1. A paper on sniffing Near Field communication. https://www.researchgate.net/publication/265976861_Eavesdropping_Near_Field_Communication
2. A paper on sniffing NFC communication https://pdfs.semanticscholar.org/4da6/fad9ecf18f60e68e235364bf4f7729b36ba9.pdf
3. Some random guy used a software called SDR Console to sniff communication between MiFare RFID tags. PS: He also used a AM Loop antenna https://www.youtube.com/watch?v=wPi3PC3F-yw
4. Hacked Mifare Desfire Using side channel attack https://www.emsec.ruhr-uni-bochum.de/media/crypto/veroeffentlichungen/2011/10/10/desfire_2011_extended_1.pdf
5. Demodulation information https://pdfs.semanticscholar.org/4da6/fad9ecf18f60e68e235364bf4f7729b36ba9.pdf