Rolljam: Replication and Extension of Testing

Nick Papadakis
California Polytechnic State University
San Luis Obispo, United States
npapadak@calpoly.edu

Ayan Patel
California Polytechnic State University
San Luis Obispo, United States
apatel60@calpoly.edu

Tanay Gottigundala
California Polytechnic State University
San Luis Obispo, United States
pgottigu@calpoly.edu

ABSTRACT

This paper explores the practicality of the rolljam attack and extends testing to other cars. The Rolljam attack is attack against rolling codes in key fobs used to unlock cars. An adversary can capture a rolling code signal from the key fob while jamming the car's receive window. Then, when the victim tries to unlock the car again, the adversary can send the first captured code to the car while capturing the second code. The adversary now has this rolling code to unlock the car at a later time. We used a Hack RF One and a YARD Stick One to replicate the attack. After adjusting parameters, we tested the attack against several cars including a Honda Civic 2012, Toyota Tacoma 2008, Subaru Crosstek 2013, and a Ford Escape 2016. Out of these cars, we were able to successfully replicate the attack on the Honda Civic 2012.

ACM Reference Format:
Nick Papadakis, Ayan Patel, and Tanay Gottigundala. 2020. Rolljam: Replication and Extension of Testing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A Rolljam attack is an attack on the rolling code system that modern key fobs use to unlock cars. This attack was first presented at Defcon in 2015. It allows an active adversary within radio range of a user trying to unlock their car to obtain a valid unlock code, without the user knowing. The adversary can then use this code later to gain access to the vehicle [11]. This attack is still relevant and important since car key fob systems only get updated when a consumer purchases a new car.

Here are our main contributions to this research:

(1) We were able to replicate the attack on a car make that was not on the list of vulnerable car makes in the initial research. We replicated the attack on a Honda Civic 2012.
(2) We explored the practicality of the rolljam attack using consumer hardware. Existing research in this field does not properly document how to replicate each step of this attack. The HackRF One and the YARD Stick One are readily available to consumers. We described the placement of the hardware relative to the car, the GNU Radio configurations, and the filters we used in detail.

1.1 Rolling Code Key Fobs

Car key fobs use a rolling code system. The car and the fob both have the same pseudo-random number generator algorithm. During the initial setup of the car and fob, they are both seeded with the same initial value. This way, when the fob sends a new code, the car will be able to compute the expected value and compare it against the received value. If the two values match, the car will unlock [11]. This system is resilient to brute force attacks due to the length of the key. For example, Samy Kamkar presented at Defcon that he attacked a NM95HS01/NM95HS02 HiSeCTM High Security Rolling Code Generator which has 2^{48} possible key combinations [6]. This is not a huge key size by today's standards, but when you consider that the adversary must be within the range of car to test their codes, it makes a brute force attack a lot less realistic.

2 BACKGROUND

In this section, we describe the key components and features of our hardware that allow us to perform this attack. To perform this attack, we used a HackRF One and a YARD Stick One, both of which we chose because they are relatively inexpensive and easy to purchase, which we believed would help show the practicality of this attack.

2.1 PRNG

Pseudo-Random Number Generators create a stream of random numbers that is started with a seed. The same algorithm can create different streams with different seeds and should

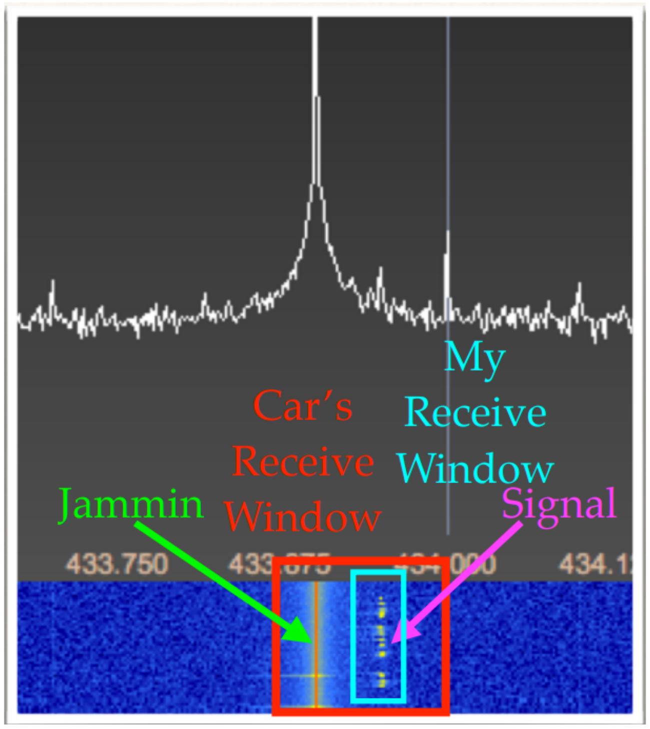


Figure 1: Frequency diagram of a Rolljam attack [11]

create the same stream given the same initial seed. To an eavesdropper than does not know the initial seed, the data produced by the PRNG should appear random. PRNGs are used in key fobs and cars and are synced by initializing to the same seed [8].

The Rolljam attack works by allowing the attacker to save a code to be used later on to unlock the car. Cars have a window to allow codes that are typically within 256 codes from the pseudo-random value set in each device [7]. If you are far from your car and press the button on your key fob then you are de-synchronized with the car. The car will still accept the code up to the next 256 codes. This means if you were to press your key 300 times and then try it with your car, it will not unlock. This can be fixed by re-syncing your key fob to the car using certain patterns with the ignition that may differ based on the manufacturer of the car. Nevertheless, if we were to de-synchronize a key fob on accident, we will be able to re-sync it with the car. Therefore, we do not run the risk of bricking hardware in our personal vehicles that would be prohibitively expensive to replace.

2.2 Rolljam Attack

The attack requires the attacker's hardware to be within range of the victim while they are trying to open the car. A frequency diagram of the attack is depicted in Figure 1. First, the attacker jams at a frequency that is within the car's receive but is not within the attacker's receive window. The victim then presses the key fob to unlock the car, but it does not work because of the jamming attacker. However, the attacker can still capture the code that was sent due to their more narrow receive window. Wanting to get into their car, the victim will then press the key fob again. The attacker continues to jam and captures this second code. The attacker then stops jamming, and replays the first captured code. The car will then unlock, and the victim will continue with their day. Now that the attacker has the second code, they can then replay it later in order to unlock the victim's car [11].

2.3 Software-defined Radio

A SDR is a piece of communication equipment capable of receiving and/or transmitting radio signals, in which some of the physical layer functions are software defined [9]. This allows the SDR to change things like modulation, operating frequency, sample rate, etc. In addition, due to the fact that operations are managed digitally, you can perform advanced analysis and signal processing.

2.3.1 HackRF One. The HackRF is a software defined radio capable of transmitting and receiving from 1 MHz to 6 GHz with a sample rate of up to 20 million samples per second. We selected this for our primary receiver and transmitter [5].

2.3.2 YARD Stick One. The YARD Stick One is a dongle capable of transmission and receiving at frequencies less than 1 GHz with a maximum power output of 50 mW [4]. We selected this to use as our jammer, as we found reports of successful Rolljam attacks being carried out with a YARD Stick. We preferred to use a lower power jammer as to not interfere with legitimate users of the frequency we were attacking.

2.3.3 Gqrx. Gqrx is an open-source software defined radio that can connect to radio dongles and display radio signals and their frequency against time [3]. We used gqrx to visualize radio signals from the HackRF and determine the frequency of key fobs.

2.3.4 GNU Radio. GNU Radio is an open source framework for simulating and deploying radio systems [9]. GNU Radio's GUI allows users to create flow graphs and interact with a SDR, such as setting frequency and sample rate, and applying filters. It was the program we used for the capture and replay of signals.

2.3.5 RfCat. The YARD Stick One comes with RfCat firmware installed. RfCat allows you to control the YARD Stick One using a python shell. The main functionality includes setting

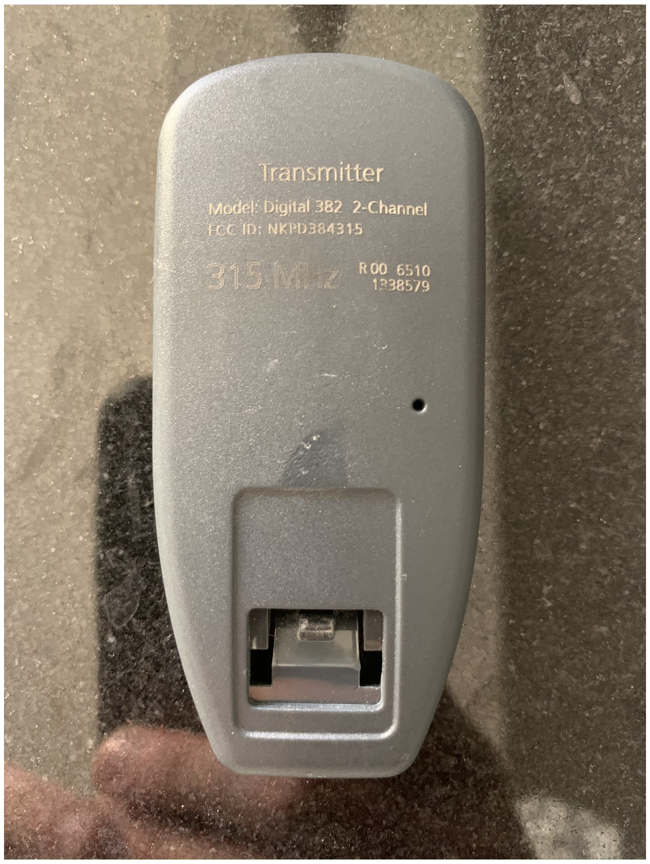


Figure 2: The back of the garage remote we targeted for a replay attack

the power and frequency and transmitting [12]. We used Rf-Cat to control the YARD Stick One and transit our jamming signal.

3 DETERMINING THE FREQUENCY

The first step of this process was to determine the frequency of the garage door remote. To make matters straightforward, the transmission frequency was printed right on the back of the remote, as you can see in figure 2. However, to be certain, since this is a cheap device, we wanted to confirm that it really was transmitting on that frequency first. To do this, we used the frequency scanning tool gqrx. We set the center frequency to 315 MHz, and as you can see by the waterfall graph in the bottom portion of 3, the frequency of the garage remote's transmission was just slightly higher than 315 MHz.

4 CAPTURING THE WAVEFORM

Using GNU Radio, we were able to capture the signal from both a garage door opener and a key fob. We then saved this

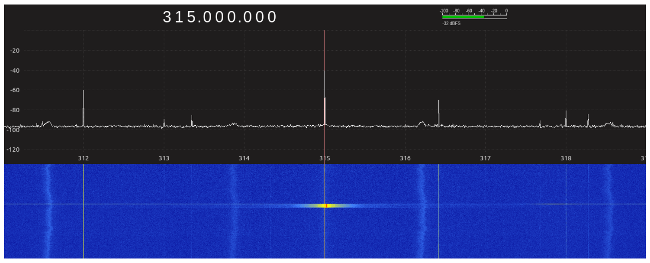
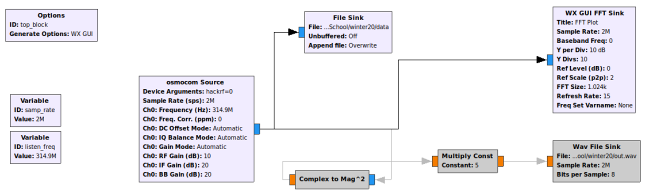


Figure 3: Scanning to find the frequency of the garage door remote



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: listen_freq
Value: 314.9M

osmocom Source
Device Arguments: hackrf=0
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 314.9M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Automatic
Ch0: IQ Balance Mode: Automatic
Ch0: Gain Mode: Automatic
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

File Sink
File: ...School/winter20/data
Unbuffered: Off
Append file: Overwrite

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2M
Baseband Freq: 0
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

Complex to Mag^2

Multiply Const
Constant: 5

Wav File Sink
File: ...ool/winter20/out.wav
Sample Rate: 2M
Bits per Sample: 8

Figure 4: GNU Radio Companion flow graph for capturing the garage remote signal

signal to a file that we later would use to replay the signal. We had to make sure that the listening frequency was at a 0.1 offset from the true frequency of the transmission so that noise caused by the HackRF at its center frequency would not interfere with the transmission waveform. We also had to set the sampling rate to 2M for better accuracy. We set the RF gain to 10, the IF gain to 20 and the BB gain to 20. As shown in figure 4 this was saved into a file and was sent to a graphing sink for visualization.

5 ANALYZING THE WAVEFORM

We used Audacity to visualize the captured waveform to be able to see the modulation in the wave. We were clearly able to see the on (1) and off (0) values that were in the code transmitted. For the replay attack we do not need to decode these, but simply playback these codes from a file. We captured two wave forms from the garage door opener and compared them as shown in figure 5. We were able to see that these two captures were identical and therefore the opener simply sends the same code every time the button is pressed. This means a replay attack can reuse the same captured frequency to open the garage door as many times as you want. Looking at the wave forms from a key fob, we can see that each one is different because rolling codes are used so this code can only be used once.

6 REPLAYING THE WAVEFORM

Using GNU Radio we were able to transmit the frequency that was saved to a file through the HackRF. The flow graph

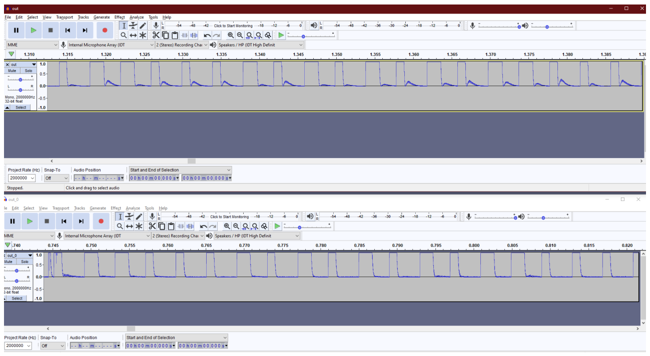
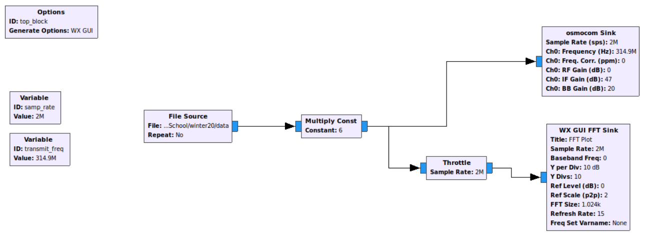


Figure 5: Comparing two captures from the garage door opener


Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: transmit_freq
Value: 314.9M

File Source
File: ...School/winter20/data
Repeat: No

Multiply Const
Constant: 6

osmocom Sink
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 314.9M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 20

Throttle
Sample Rate: 2M

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2M
Baseband Freq: 0
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

Figure 6: GNU Radio Companion flow graph for transmitting the garage remote signal

is shown in figure 6. From the file source we had to add a multiplier to be able to amplify the actual signal and for the transmission to be less susceptible to noise. In addition, we must make sure that our transmitting frequency is the same as the receiving frequency used to capture the signal. To amplify the signal further we increased our RF gain to 47 dB which is the max value for that parameter. We left the IF gain at 0 and the BB gain at 20.

6.1 Garage

The first device that we tested this replay attack on was a garage remote. We decided to attack this one first since we knew that garage remotes generally do not using rolling codes. This allowed us to use the same waveform to open and close the garage. The FCC ID of the garage remote was NKPD384315.

6.2 Car

An important thing to keep in mind here is that a key fob has rolling codes while the garage does not. This means that the wave form we captured for the car can only be used once. The car key we targeted is depicted in figure 7. Determining the frequency of the key was slightly more complex than with the garage remote. On a car key replacement website [1], we were able to determine the FCC ID of they key to likely be

Figure 7: The car key we targeted for a replay attack

MLBHLIK-1T. We were then able to reference online FCC records to determine the frequency of the key to be about 313 MHz [2]. We then confirmed that this was correct using gqrx like we did for the garage remote.

We then slightly modified the send and receive flow graphs that we used for capturing and transmitting the garage door signal. We changed the transmission and receive frequencies to 313.5 MHz, slightly deviated from the actual center frequency of the key fob. Next, we took the key out of range of the vehicle, and captured the code. This way, the car would not discard the current code as a valid code but we could still capture it. We then returned to within range of the car and were able to replay the signal with the HackRF and unlock the car.

7 JAMMING

In the first deliverable, we were able to successfully perform a replay attack on a Honda Civic 2012; however, we did so without jamming, which meant that we had to make sure the key fob was out of range of the car before we captured it. This served as a good initial prototype, but in order for this attack to be practical, we need to be able to use the key fob while in range of the car and successfully jam the signal being sent while simultaneously capturing the code from the key fob so that it can be used at a later time.

7.1 YARD Stick One

We used the YARD Stick One to be able to jam the car's receiver [12]. We setup RfCat to be used with the YARD Stick. We set a frequency at a slight offset of the center key fob frequency but still within the car's receive window. We then set the YARD Stick to transmit garbage data on a loop at its maximum output power. While we transmit noise at this frequency, the car is not able to receive a clear signal from the key fob due to the size of its receive window. All jamming tests were done on private property for short periods of time so as to not interfere with bystanders trying to access their vehicles.

We tested the jamming attack with the yardstick the 2012 Subaru Crosstrek and the 2012 Honda Civic. For the Subaru Crosstrek, the jammer was able to block the victim from being able to lock their car when the user was approximately 30 feet from their car. The attack did require the jammer to be within the line of sight of the user between them and the vehicle to reliably jam the signal.

The Honda Civic jamming was even less feasible for a practical attack. In order for the jammer to be able to reliably jam the victim's signal, the victim had to be over 100ft from the car. The jammer had to be located very close to the car, again in the victim's line of sight of the vehicle. This attack was extremely sensitive to the angle of the jammer antenna as well, requiring that the attacker with the laptop spin until a working angle was found. This suggest that the transmitter for the Honda Civic was much stronger than the Subaru's. This also brings into question of whether a YARD Stick One would be powerful enough of a jammer in order to make this attack practical.

8 FILTERING THE SIGNAL

For the roll jam attack to work we need to capture the key fob signal while we are jamming the car's receiver. This is done by setting a filter on the capture so that only a small width of frequencies are recorded. By doing so, we can filter out the jamming frequency that is slightly offset of the key fob frequency.

8.1 Low-Pass Filter

To be able to do this we use a low-pass filter when we capture the signal. A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. We used a low-pass filter with a cutoff frequency of 40K and a transition width of 50K.

8.2 DC Filter

We also added a DC filter which allows us to filter out the noise the HackRF makes on the set center capture frequency. We added a DC Blocker with a length of 1024. The flow graph for this configuration is depicted in figure 8. A sample key capture using this configuration can be seen in figure 9.

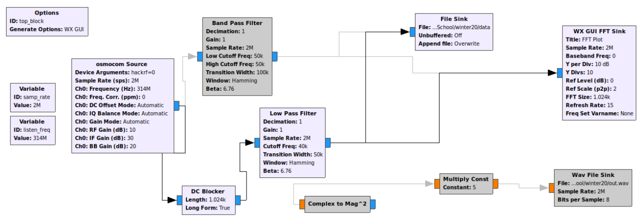


Figure 8: DC Filter and Low-pass Filter added to Receiver Flow Graph

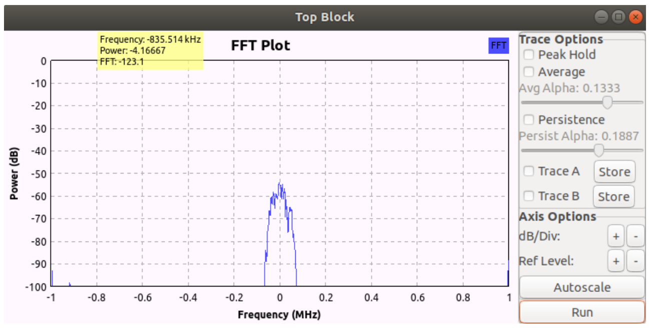


Figure 9: The filtered key fob signal received while the YARD Stick was jamming at 315.1 MHz

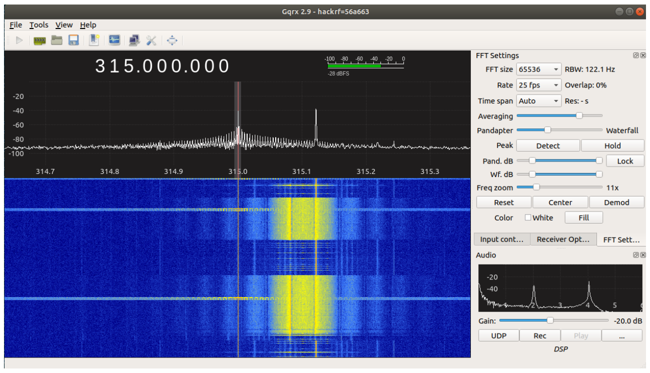


Figure 10: A diagram of the frequencies of receiving a key fob signal while jamming at the same time.

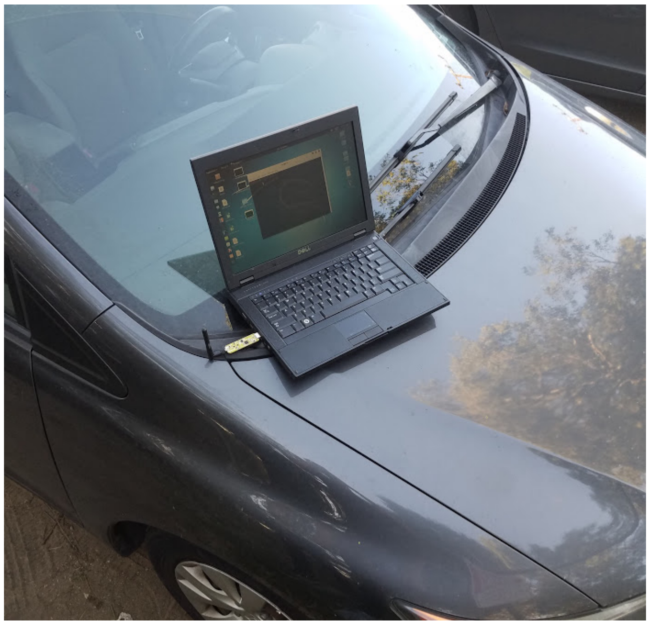


Figure 11: Position of the YARD Stick One relative to the car

9 PUTTING THE ATTACK TOGETHER

We tried to put the attack together and capture the key fob signal while jamming the Honda Civic. The setup included the YARD Stick One transmitting at a slightly offset frequency than the key fob, jamming the signal within the car's receive window. The HackRF was used to receive the key fob signal while filtering out the jamming signal. A sample run of what the frequencies of a simultaneous key signal capture and YARD Stick jam can be see in the gqrx screenshot shown in figure 10. The key fob signal is being transmitted on 315 MHz while the jamming is being transmitted on 315.1 MHz.

10 PARAMETER TUNING

Our main focus for this deliverable involved tweaking the parameters, specifically the DC Filter and Low Pass Filter.

10.1 DC Filter

The DC filter is used to filter out the spike in noise coming from the HackRF device at the listening frequency. Capturing at an offset frequency seemed to work with a replay attack, however, adding a low-pass filter was difficult due to the fact that the cutoff is equivalent distances on either side of the center listening frequency. To be able to keep the listening frequency at the desired frequency, we used a DC filter to remove noise and capture a clear signal from the key fob. As shown in figure 13, we added the filter after the input source and did not change the default parameters.

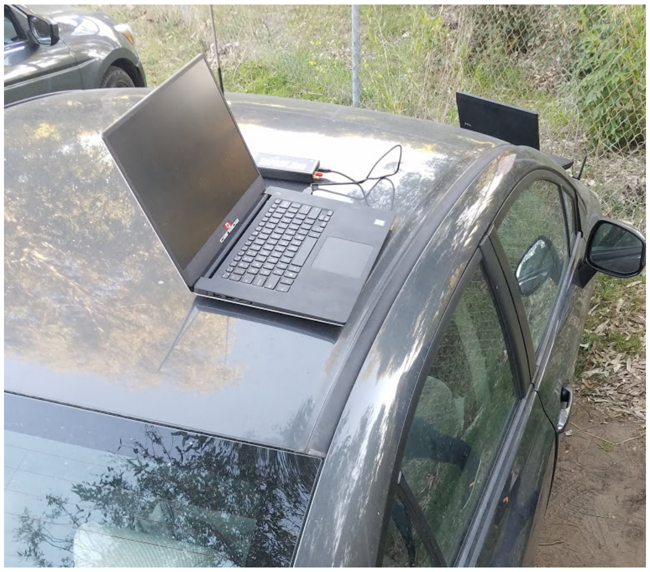
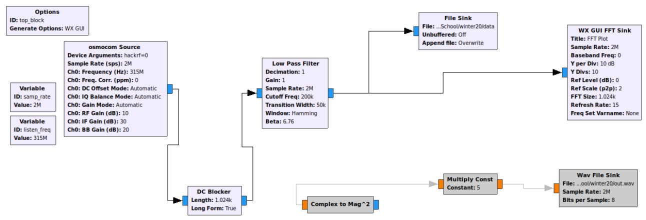


Figure 12: Position of the HackRF One relative to the car



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: listen_freq
Value: 315M

osmocom Source
Device Arguments: hackrf=0
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 315M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Automatic
Ch0: IQ Balance Mode: Automatic
Ch0: Gain Mode: Automatic
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 30
Ch0: BB Gain (dB): 20

DC Blocker
Length: 1.024k
Long Form: True

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2M
Cutoff Freq: 200k
Transition Width: 50k
Window: Hamming
Beta: 6.76

File Sink
File: ...School/winter20/data
Unbuffered: Off
Append file: Overwrite

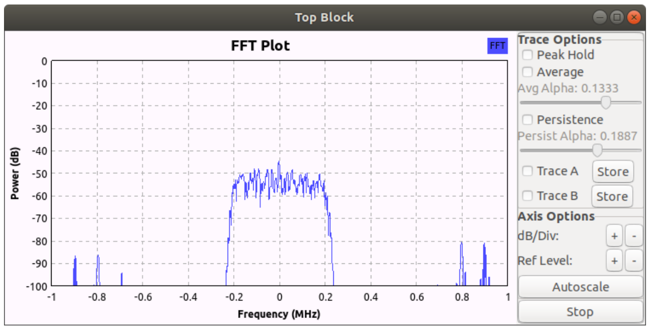
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

Complex to Mag^2

Multiply Const
Constant: 5

Wav File Sink
File: ...ool/winter20/out.wav
Sample Rate: 2M
Bits per Sample: 8

Figure 13: Our new GNURadio Companion flowchart with tuned receiving parameters



Avg Alpha: 0.1333
Persist Alpha: 0.1887

Figure 14: Receiving the Honda's unlock signal in GNURadio Companion with our new parameters

10.2 Low Pass Filter

The Low Pass Filter is used to cutoff the captured signal frequency so that the noise from the jammer is not captured. The Honda Key fob, however, seemed to send multiple signals around the center frequency that were necessary for the codes to work on the car. Therefore, we had to set the low pass filter to capture as much of the key fob signal as possible while still blocking the jammer signal. In addition, this required that we set the jammer frequency far enough offset from the key fob frequency while staying in the receive window of the car.

To be able to tune the cutoff frequency for the low pass filter, we had to capture a key fob signal away from the car, and then transmit the captured signal to the car to see if it was successful. We overcame this walking dilemma by using another car to drive around a block where one end of the block was out of range of the car and the other end passed the car. We captured the signal at the far end and drove around to replay the signal to the car at the other end. This sped up the process of determining the cutoff frequency of the low pass filter.

As show in figure 13, we ended up using a low pass filter with a cutoff frequency of 200K and a transition width of 50K. This seemed to be the optimal filter parameters to be able to capture a key fob signal and jam the car successfully. Figure 14 shows the captured signal from the key fob after the low pass filter was applied.

11 ROLLJAM

We revisited the full Rolljam attack from our previous deliverable in hopes of executing it successfully.

11.1 Multi-Computer Setup

In order to maximize our chances of success, we decided to separate the placement of the jammer and the receiver. This would allow for greater tuning of the placement of each device. With the previous deliverable we had issues with achieving placement that worked well with both devices being connected to the same machine. If we placed the computer in position so that the jammer would work well, the receiver was unable to successfully capture the signal. If we place the computer in a position where it was able to receive the signal, the jammer would not work. Using one machine for each component enabled us to avoid these issues, but does make our attack less practical.

11.2 Placement Tuning

A big part of this deliverable was adjusting the placement of both the HackRF and the YARD Stick. Once we decided to use the multi-computer setup described in the previous section, we attempted to make the attack as practical as possible by having the jammer as close to the receiver as possible. We began by placing the YARD Stick right on top of the car's receiver as shown in Figure 11, which is where it stayed for the remainder of our experiment. We decided not to change the positioning of the YARD Stick since the jammer tended to work much more inconsistently and unpredictably when placed in other locations. Instead, we tried to minimize the distance between the HackRF and the YARD Stick by testing various positions for the HackRF. We began by putting the HackRF right next to the key fob and moved it closer and closer to the car until we arrived at the placement shown in Figure 12.

The placement and power of these devices reduces the practicality of this attack. It is unlikely that an individual wouldn't notice two computers sitting on top of their car when they try to unlock it. The jammer and the receiver for this experiment simply did not have enough power, as described in the background section, to allow us to discreetly jam the signal while simultaneously receiving it. Using a more powerful jammer and receiver than the YARD Stick One and the HackRF One would allow us to perform the attack from a distance.

12 ADDITIONAL CARS

In addition to the Honda Civic 2012, we tried this attack on three additional cars: a Toyota Tacoma 2008, a Subaru Crosstrek 2013, and a Ford Escape 2016. We were unable to successfully replicate this attack on these cars. Each car failed at the initial step, performing an out of range replay attack where the key is taken out of range of the vehicle and its next rolling code is captured and then replayed in range of the car.

Despite Ford being on the list of car makes that are vulnerable to the attack, it likely failed with the Ford Escape due to the fact that this particular key fob was manufactured after the Rolljam attack was discovered. One possible explanation for this not working on the newer model could be that Ford added a timestamp that automatically rolls keys based off of the time. Another explanation for why this attack might not work could be due to the fact that some car manufacturers have started to implement challenge-response schemes [14]. This works by first having the key send a request, the car responding with a challenge, and the key replying to the challenge to unlock the car. Despite these additional security measures, [13] indicates that a similar attack is possible on this car; however, the article does not go into sufficient technical detail to allow us to replicate the attack.

The Tacoma in particular was surprising, as [10] indicated that Toyota's in 2015 were vulnerable to this attack. Additionally, it was the oldest vehicle in the test. The owner of the Tacoma also reported that he has observed some interesting behavior in the past with his key fob. He noted that he has seen the truck's tail lights illuminate when he has gotten near the vehicle with the fob in his pocket, as if he had pressed the key, but had not. We were unable to replicate this behavior ourselves, but it could suggest that the key fob system is employing something more sophisticated than just rolling codes.

13 CONCLUSION AND FUTURE WORK

We successfully got the rolljam attack to work on one car - a Honda Civic 2012. However, we did use two computers to be able to have the antennas at the correct positions. With a more powerful jammer, we speculate we could have used one computer to be able to jam and receive the signal. The attack did not work on any of the other cars we tried. This was due to the fact that some of these cars used newer key fobs and possibly implement the rolling code algorithm with a time parameter or a challenge and response pair. Future work can look into possible reasons why these cars are not susceptible to this attack. This could include scanning out of band frequencies or analyzing the challenge and response signals.

REFERENCES

[1] [n. d.]. https://carkeysexpress.com/store/keys-and-remotes/Honda/Accord/2012/1001517-honda-remote-key-combo-4-button-w-trunk

[2] [n. d.]. https://fccid.io/MLBHLIK-1T

[3] [n. d.]. GQRX 2.9. http://gqrx.dk/

[4] [n. d.]. greatscottgadgets/yardstick. https://github.com/greatscottgadgets/yardstick Library Catalog: github.com.

[5] [n. d.]. HackRF One - Great Scott Gadgets. https://greatscottgadgets.com/hackrf/one/

[6] [n. d.]. NM95HS01/NM95HS02 HiSeCTM High Security Rolling Code Generator. http://www.meditronik.com.pl/doc/plus/nm95hs01.pdf

[7] MARSHALL BRAIN. [n. d.]. How Remote Entry Works. ([n. d.]). https://auto.howstuffworks.com/remote-entry2.htm

[8] Season Cherian. 2019. Jam and Replay Attacks on Vehicular Keyless Entry Systems. (2019). https://s34s0n.github.io/2019/07/18/Jam-and-Replay-Attacks-on-Vehicular-Keyless-Entry-Systems/

[9] Jorge Duarte García. 2016. Software Defined Radio for Wi-Fi Jamming. (05 2016). https://doi.org/10.13140/RG.2.2.23772.90240

[10] Andy Greenberg. [n. d.]. This Hacker's Tiny Device Unlocks Cars And Opens Garages. ([n. d.]). https://www.wired.com/2015/08/hackers-tiny-device-unlocks-cars-opens-garages/

[11] Samy Kamkar. 2015. Drive It Like You Hacked It. https://samy.pl/defcon2015/2015-defcon.pdf

[12] Michael Ossmann. 2014. Software Defined Radio with HackRF. https://greatscottgadgets.com/sdr/1/

[13] Seth Rosenblatt. 2019. This hack could take control of your Ford. https://the-parallax.com/2019/05/03/hacker-ford-key-fob-vulnerability/

[14] Margaret Rouse. [n. d.]. challenge-response authentication. https://searchsecurity.techtarget.com/definition/challenge-response-system