Breaching Vehicle RF Systems Using Replay Attacks

Vlad Slyusar, Alfred Kishek

Department of Computer Science, University of Michigan
CIS544 - Computer Network Security, Fall 2016
Dr. Di Ma

***Abstract*—With the increase of vehicle complexity, vehicle systems are becoming progressively more vulnerable. Vehicle ports are opening to several different wireless communication protocols: Wi-fi, Bluetooth, and a copious amount of vehicle communication networks. We present the implementation of a wireless replay attack on one of the most traditional wireless vehicle systems: *Remote Keyless Entry*. We begin by presenting the standard communication protocol for most keyless entry systems. We further provide a proof-of-concept attack on a 2015 Ford Fusion and a 2011 Toyota Sequoia. Finally, we introduce two preventative methods to further the security of the system and prevent this type of replay attack.**

I. Introduction

The first keyless entry systems appeared on the automotive market in the 1980s. Compared to mechanical keys, a wireless entry was actually perceived as more secure. In fact, European auto insurance companies made remote keyless entry systems a requirement for acceptance [6].

Not long after, remote keyless entry systems, or *RKE* systems, became a target for attack. Historically, most systems would transmit a single key without any encryption or security mechanism in place. An attacker could simply listen to the correct channel and intercept only a single key to *permanently* access a victim's vehicle. Preventative measures, once more, began appearing on the market to prevent this style of replay attack (discussed in section III).

In this paper, we examine a replay attack on the latest RKE security measure (code hopping with KeeLoq encryption). We demonstrate our results on two vehicles: a 2015 Ford Fusion and a 2011 Toyota Sequoia. Furthermore, transmissions from the two different OEMs are decoded for analysis in section VII.

Conclusively, we present two possible countermeasures in section VIII. We assess the feasibility of implementations, including several constraints (such as costs, range, and FCC regulations).

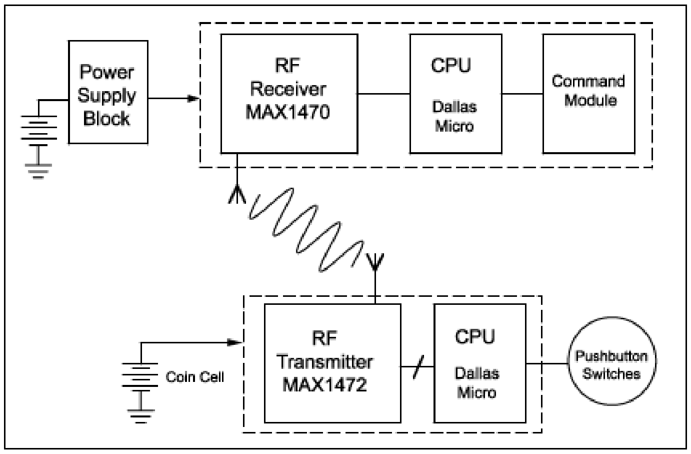


Figure 1. Typical RKE circuit with keyfob transmitter and vehicle receiver [6].

II. Remote Keyless Entry

Remote Keyless Entry Systems were introduced for the obvious advantage of convenience. A standard RKE system consists of two elements: A *keyfob* containing an RF transmitter, and a vehicle containing an RF receiver. A wireless signal is sent to the vehicle by the keyfob, where it's decoded. From there, the vehicle interprets the message and executes the required function (see Figure 1). Typically, RKE systems must meet requirements such as range, battery life, reliability, cost, and even regulatory compliance.

The wireless frequency is 315MHz in the US and Japan, whereas in Europe and Asia the carrier frequency is 433.92MHz. On the signal transmission level, the modulation used is *amplitude-shift keying*, or ASK. The amplitude is modulated between a high and a low level (near zero to save power). This produces a complete *on-off keying* modulation, OOK. For an example of OOK, see figure 2 [3].

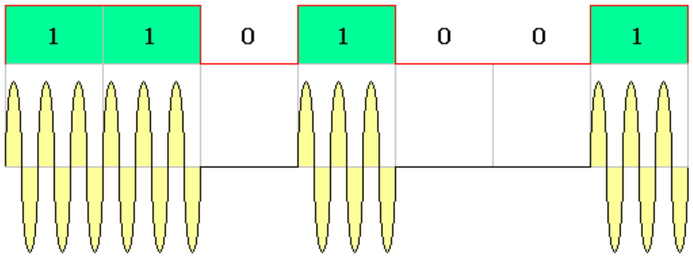


Figure 2. Typical on-off keying transmission scheme [6].

III. Mechanisms to Secure RKE

With severe requirements on RKE systems, it's not surprising that a low power device with a single transmission could be subject to several attacks. The historical implementations of RKE quickly became exposed to *replay attacks*. An attacker could eavesdrop on the transmission frequency, capture the key as it's being pressed, and simply replay the same key to gain access to the victim's vehicle. As a result of replay attacks, several systems began to appear to prevent intrusion.

A. Rolling Code

The first method we will study is *rolling code* encryption (*hopping code*). Rolling code was introduced to prevent replay attacks and exists currently in many systems, including garage door openers.

An ordinary rolling code system will have a cryptographically secure pseudo random number generator (PRNG) in both the transmitter and receiver. In this type of model, both the receiver and transmitter will have a seed value. The transmitter will then send the next code in the sequence. The receiver compares the code to the next 256 codes in the sequence [1]. If one of the codes matches, the receiver will execute the command as requested.

B. KeeLoq

KeeLoq, developed in 1996, is an encryption mechanism with a dedicated hardware block-cipher. This cipher uses a *non-linear feedback shift register* (NLFSR), which is a common component of modern stream ciphers.

Encoders encrypt a zero-padded block with the KeeLoq cipher to produce a 32-bit rolling code. An initial vector is XORed to the least significant bits for encryption and decryption. For full specification on KeeLoq hardware, see patent [2].

Even with such an intricate proprietary hardware encryption method (patent sold for over $10 million dollars and used by over ten known manufacturers), a weakness still exists in the single message transmission protocol making replay attacks *still* possible. Within this paper, we implement the attack on two modern vehicles. The vehicles are manufactured by competing OEMs and each employ a custom implementation of the rolling code mechanism as well as proprietary data transmission scheme.

[1]Window size is configured by the manufacturer.

IV. Experimental Setup and Technology

In order to perform the attack, the methodology behind observing and interacting with wireless RKE signals has to be understood. As mentioned, the communication consists of a simplex (one directional) channel from keyfob to the car centered around 315Mhz (within the US). Several approaches exist to interacting with wireless signals. On the fundamental level, the main requirements are an antenna with proper length and an oscillation circuit. This an approach would present the lowest part cost, however, the radio frequencies were instead analyzed using a more versatile radio solution: *Software Defined Radio*, or *SDR*. This device replaces tasks traditionally implemented dedicated hardware components with a software-based solution. The main benefit of an SDR centers on the wide range of frequencies accessible to the user (through simple software changes). The SDR must still be paired with a proper length antenna to achieve high signal power.

In theory only two separate radios are necessary, however, three SDRs were used for the implementation. At least one radio, capable of transmission, must be used for jamming the receiver and replaying the captured keys. A second radio is necessary to record the actual RF sequence transmitted by the keyfob. For the purpose of the project, an additional radio was used for remotely monitoring the air waves during the entire process. The following devices were used:

- HackRF One SDR with ANT500 antenna
- 2x generic DVB-T usb RTL-SDR dongles

The *HackRF* device provides the capability to not only receive signals, but also transmit in frequencies ranging from 1Mhz to 6Ghz. Many other SDRs or custom radio devices that are able to perform this functionality within a frequency range that includes 315Mhz should also work. The HackRF was used in conjunction with the ANT500, a simple telescopic antenna designed for frequencies between 75Mhz and 1Ghz. The ideal length for antenna expansion is calculated using the following formula for a quarter wave antenna [1]

$$Length(meters) = \frac{1}{4} * \frac{SpeedOfLight * 0.95}{Frequency} \tag{1}$$

For the frequency of interest, 315,000,000Hz, the ideal antenna length equates to 22.6 centimeters, or 8.9 inches. Although the actual antenna does not have to match exactly, a close approximation was maintained to maximize dissipation power and effective range of transmission.

After the necessary hardware is obtained, software support for each SDR must be implemented. GNU Radio, a powerful and free software development toolkit, was selected for implementing the radio-based attack. Furthermore, the GNU Radio Companion (GRC) graphical toolkit addition was employed to simplify the research and generate simple to follow flow diagrams of the system. The complete control set for both jamming and reception radios was implemented within GRC with an output file of captured raw data stored for further analysis. Matlab was used to analyze, demodulate, and interpret the data into hexadecimal strings representing each RKE transmission.

Finally, the most important hardware used within the project was the two test vehicles presently available and their respective remote keyfobs. The following vehicles were tested:

- 2015 Ford Fusion
- 2011 Toyota Sequoia

The general setup consisted of testing each vehicle individually and within a 10 meter range of the radio devices to reduce the necessary power output and limit unintentional interference.

V. Methodology

In this section, we will discuss the methodology used to bypass the rolling key and KeeLoq system. The replay attack presented shown here was demonstrated by Samy Kakamar at Defcon 23. However, no source code, tooling, or reproduction instructions were ever published [5].

As the victim presses their key to unlock their vehicle, the attacker will jam the first message, causing the receiver to not receive the request. Subsequently, the victim innocently transmits another key press. This time, the attacker will intercept the second key press, and send the first (while simultaneously halting the jam). This will, as expected, unlock the vehicle. However, the attacker now has the next key sequence in his buffer. If the attacker continues following this method, for every subsequent message sent by the victim, the attacker will always have the next key in the sequence.

A. Jamming The Receiver

One approach to disable the keyfob reception circuitry within the vehicle is to transmit a high amplitude noise signal within the receiver's active frequency range. The inherent problem with this implementation is the requirement for the attacker to properly detect the keyfob data while preventing the vehicle from doing so. A highly directional antenna may be used with careful alignment towards the vehicle, but such an approach requires specialized hardware and imposes the burden of proper antenna alignment on the attacker. Instead, an antenna with an omnidirectional radiation pattern was selected for the jamming function. While a large amount of energy is wasted with such an approach, the orientation of the final jamming device would have little impact on performance and provide greater simplicity of use.

Besides the greater power requirements, the major problem with omnidirectional jamming is that the attacker's receiver will also be subject to the high amplitude interference. If the injected noise overlaps significantly with the actual data, recovery may be extremely difficult or impossible. The solution revolves around exploiting the wide bandwidth of most receivers.

While the RKE system officially operates on 315Mhz, the specification reflects the center frequency and the actual signal will deviate slightly during transmission. Furthermore, most keyfob circuits employ relatively inexpensive local-oscillators for generating the carrier signal. The stable frequency on which the device will transmit fluctuates with environmental factors, specifically the temperature. As a result, the center frequency of the typical keyfob can experience significant shifts throughout different seasons and climates [6]. To make up for this, most RKE receivers are designed to receive a wide frequency encompassing 315Mhz and allowing for several Mhz of deviation in either direction. Hence the exploit is possible by injecting a narrow-bandwidth, high-amplitude signal into the receiving antenna. The center frequency of the noise signal can be set sufficiently far from actual carrier frequency of the keyfob, but within the reception range of the vehicle. A graphical illustration of the principle is provided in figure 3. Because of the extremely high amplitude noise signal, the actual keyfob transmission gets interpreted as noise and is not processed by the system.

B. Processing a Jammed Transmission

While the capability to completely jam the vehicle's reception is discussed in section V-A, successful implementation of the replay attack requires that the attacker capture the valid key for later re-transmission. Following the presented jamming method, there exists an arbitrary separation between jamming noise and valid

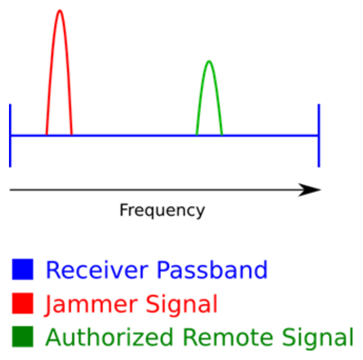


Figure 3. RKE receiver bandwidth in presence of noise and signal [7].

signal within the frequency domain. The separation allows for the application of a band-rejection filter to significantly attenuate a range of frequencies centered around the injected noise. Since the jamming device is under attacker control, the central frequency and bandwidth are known and can be programmed directly into the filter. A common illustration of a bandpass filter is provided in figure 4. The components of the recorded signal corresponding to the jamming frequency can be significantly attenuated, effectively reproducing the initial keyfob recording with the jamming signal filtered out.

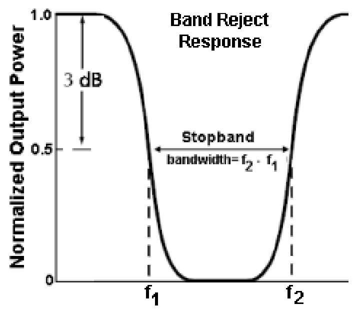


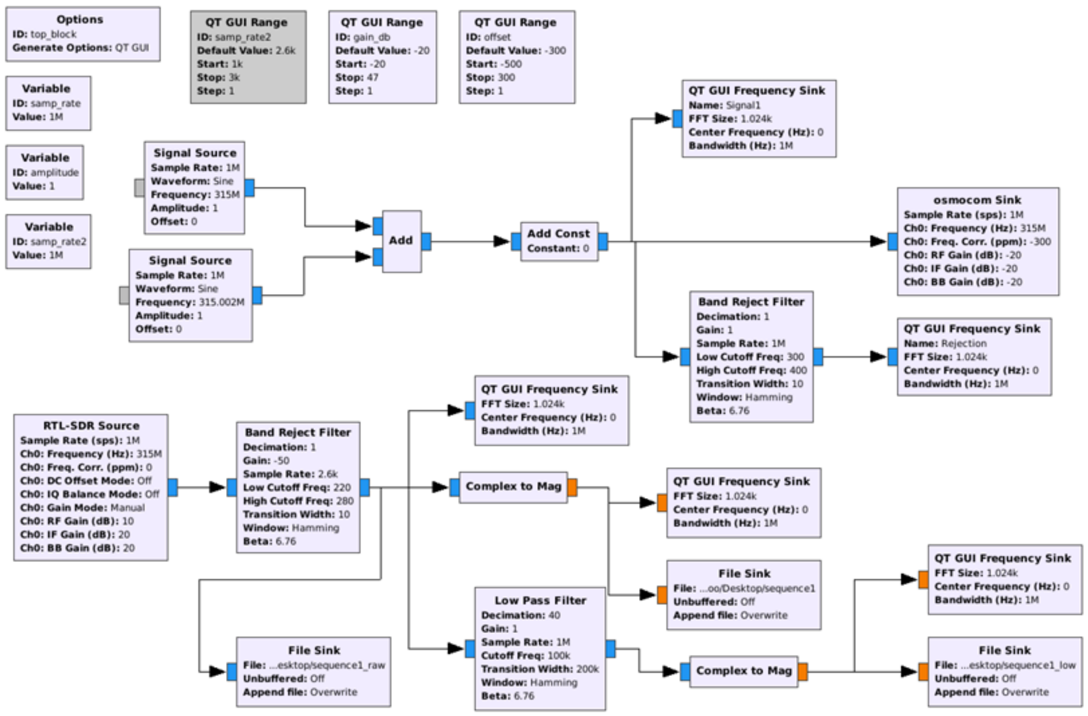
Figure 4. Bandwidth Rejection Filter (Band-Stop) [4].

Next, the filtered signal undergoes a further stage of processing where the individual keypress sequences can be isolated within the temporal domain. Basic digital signal processing (DSP) techniques are applied to eliminate the low activity time spectrum of the recording, leaving only the short transmissions containing each valid message. The SDR used for recording airwaves measures the amplitude and phase of the electromagnetic radiation absorbed by its antenna over short time intervals. For this implementation, the SDR was set to record one million samples every second. The initial measurements, stored as complex values, were converted to their magnitude representation and a time based analysis was performed to determine when signal amplitudes began to rise. The significant rise within a short time represents start of a transmission and the preceding low energy data points can be discarded. This simple method is applied to both the start and end of each transmission producing a short output file corresponding to each user keypress. The outputs can then be further analyzed or simply replayed back to the vehicle through an on-off modulator. Although for the purpose of this experiment the processing was performed in matlab, the implementation demonstrates the feasibility of fully automating the approach and directly combining with SDR software to achieve imperceptible delays for the entire process.

VI. Attack Implementation and Technical Details

In this section we introduce the specific details of implementing the attack. As previously mentioned, two separate radio devices are necessary. The HackRF and a generic RTL-SDR were both utilized within a single control program in Gnu Radio Companion. The system flow diagram for jamming and recording is presented in figure 5. The program breaks down into two separate execution flows, as well as several variable definitions. The top flow within the figure represents the jamming circuitry.

To generate a narrow bandwidth noise source, a perfect sinusoidal signal should be generated oscillating at the corresponding frequency. In reality, it was determined that a narrow signal does not attain the proper amplitude within the frequency domain. In other words, the jamming signal must exhibit a minimal bandwidth for the desired effect on car receiver. The exact details remain a topic for further study, but a possible solution was developed by combining two sinusoids with a short difference in oscillating frequency. As per the two signal sources illustrated, 315Mhz and 315.002Mhz sinusoids were combined together to generate the signal for the jam. The signal was fed through to multiple analysis blocks as well as the HackRF labeled osmocom sink. The sink block is responsible for final processing and directly controlling the execution on the software defined radio. Parameters for amplitude gain must be configured, in this case -20dB eliminates the jamming while a value of up to 47dB produces amplification. The frequency correction coefficient shifts the central frequency of the jamming signal during run-time. This provided an attacker the capability to shift the central frequency of the jammer sufficiently far from the central frequency of the signal, but still within the overall bandwidth of the RKE receiver. In a sample implementation, the keyfob operated at 314.97Mhz while the noise signal

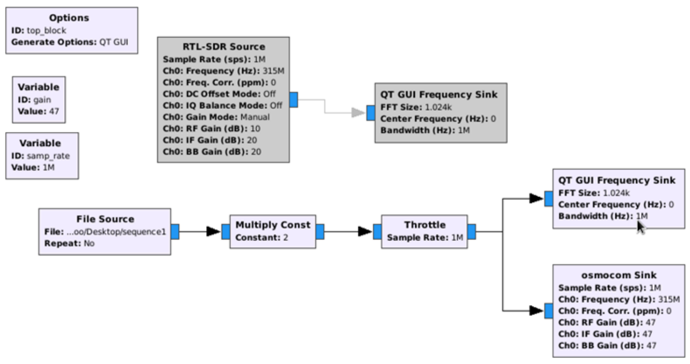


```
Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 1M

Variable
ID: amplitude
Value: 1

Variable
ID: samp_rate2
Value: 1M

QT GUI Range
ID: samp_rate2
Default Value: 2.6k
Start: 1k
Stop: 3k
Step: 1

QT GUI Range
ID: gain_db
Default Value: -20
Start: -20
Stop: 47
Step: 1

QT GUI Range
ID: offset
Default Value: -300
Start: -500
Stop: 300
Step: 1

Signal Source
Sample Rate: 1M
Waveform: Sine
Frequency: 315M
Amplitude: 1
Offset: 0

Signal Source
Sample Rate: 1M
Waveform: Sine
Frequency: 315.002M
Amplitude: 1
Offset: 0

Add

Add Const
Constant: 0

QT GUI Frequency Sink
Name: Signal1
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

osmocom Sink
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 315M
Ch0: Freq. Corr. (ppm): -300
Ch0: RF Gain (dB): -20
Ch0: IF Gain (dB): -20
Ch0: BB Gain (dB): -20

Band Reject Filter
Decimation: 1
Gain: 1
Sample Rate: 1M
Low Cutoff Freq: 300
High Cutoff Freq: 400
Transition Width: 10
Window: Hamming
Beta: 6.76

QT GUI Frequency Sink
Name: Rejection
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

RTL-SDR Source
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 315M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Band Reject Filter
Decimation: 1
Gain: -50
Sample Rate: 2.6k
Low Cutoff Freq: 220
High Cutoff Freq: 280
Transition Width: 10
Window: Hamming
Beta: 6.76

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

Complex to Mag

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

File Sink
File: ...oo/Desktop/sequence1
Unbuffered: Off
Append file: Overwrite

File Sink
File: ...esktop/sequence1_raw
Unbuffered: Off
Append file: Overwrite

Low Pass Filter
Decimation: 40
Gain: 1
Sample Rate: 1M
Cutoff Freq: 100k
Transition Width: 200k
Window: Hamming
Beta: 6.76

Complex to Mag

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

File Sink
File: ...esktop/sequence1_low
Unbuffered: Off
Append file: Overwrite
```

Figure 5. Recording system flowchart. Simultaneously jam using HackRF (top) and record with RTL-SDR (bottom).

was shifted to center around 314.89Mhz. This approach provided 80Khz of separation between central frequencies, preventing overlap and allowing for simpler filter construction to reject the noise.

The bottom portion of figure 5 illustrates the flow graph for simultaneous recording through a separate radio while the jammer operates. The RTS-SDR source block drives the SDR at 1 mega-samples/second centered around 315Mhz. The output is fed directly through a band rejection filter, which attenuates the range of frequencies corresponding to the known jamming signal by 50dB. The output of the filter contains the true keyfob information without the noise components and is further parsed and written to file. The data can be downsampled to require less storage space using the implemented additional low pass filter; however, with the interest of producing nicer visual results for demonstration the magnitude of the original signal was stored to file for analysis. The original output data in complex representation was also stored for proof of concept demonstration of replaying capabilities illustrated in figure 6. The recording was amplified by a multiplication factor to adjust for long distance between keyfob and receiving antenna. The output was then throttled to the selected sampling rate and fed to monitor display as well as the osmocom sink (HackRF One). This action overrides the previous transmission of noise signal and immediately utilizes the same radio to transmit the pre-recorded keyfob data.



```
Options
ID: top_block
Generate Options: QT GUI

Variable
ID: gain
Value: 47

Variable
ID: samp_rate
Value: 1M

RTL-SDR Source
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 315M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

File Source
File: ...oo/Desktop/sequence1
Repeat: No

Multiply Const
Constant: 2

Throttle
Sample Rate: 1M

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

osmocom Sink
Sample Rate (sps): 1M
Ch0: Frequency (Hz): 315M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 47
Ch0: IF Gain (dB): 47
Ch0: BB Gain (dB): 47
```

Figure 6. Replay recorded signal flowchart.

In the real would, an attacker would play back the first key and maintain the next recording in the buffer. Immediately after playback the first program from figure 5 can be activated once again creating a theoretical infinite loop where the attacker always jams, replays, and maintains the next real sequence generated by the keyfob.

VII. Empirical Results and Discussion

In this section, we examine the physical recording of keyfob interactions to study the protocols used and decode the data being sent with each key-press. While the experiment was performed on both test vehicles, this section focuses on the 2011 Toyota Sequoia in a detailed analysis example. In the presented experiment, the lock button was pressed five separate times with each keypress varying in length before button release. The 11.3-second output recording was imported into matlab for analysis, the magnitude representation of each data point is illustrated in figure 7. The figure represents the RF data corresponding to each respective keypress. The gaps in between each of the groups are the time delay to push button again, while the width of each plotted group corresponds to the length of keypress.

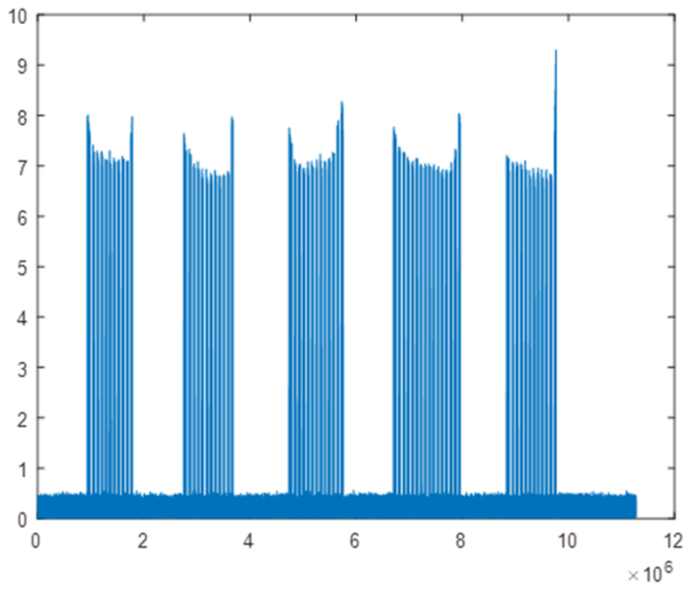


Figure 7. Five long presses of lock key.

The first group of points can be further studied to explore the exact transaction during just the first keypress event. Figure 8 zooms in on just the first cluster of points. As evident, the original recording encompasses a recursive nature. The illustrated data in figure 8 corresponds to just the first keypress and, per visual inspection, the first cluster of points differs from the rest. In fact, the ten groups following the first are evenly spaced apart and hold the same exact data.

In order to explore the specifics of the protocol, we further zoom in on just the first group from figure 8. The image is provided in figure 9 and illustrates that the transaction contains a regular pattern. Such behavior is characteristic of a synchronization pattern rather than valid data. The pulse width of each control bit is used to set the proper amplitude and spacing requirements within the receiver in the car for the current transmission.

Figure 10 further amplifies the synchronization pulse to determine the exact pattern. As evident, the pattern does not exhibit standard on-off behavior but rather transmits a series of 1-0-0 pulses. By characterizing

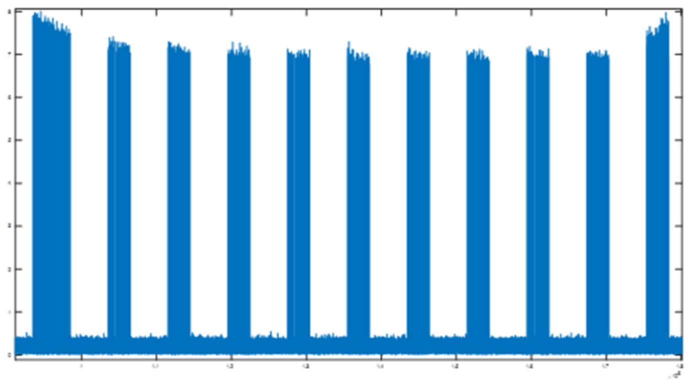


Figure 8. Transmission during a single long keypress.

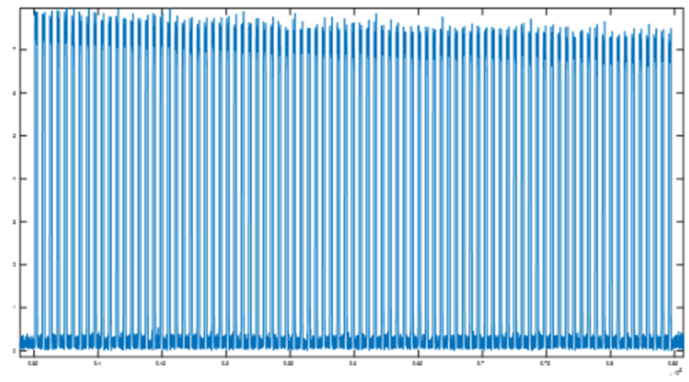


Figure 9. Synchronization, first data segment of each keypress transmission. First pulse from figure 8

the spacing within the sync pulse, we can apply the proper demodulation technique for actual data within the recorded transmission.

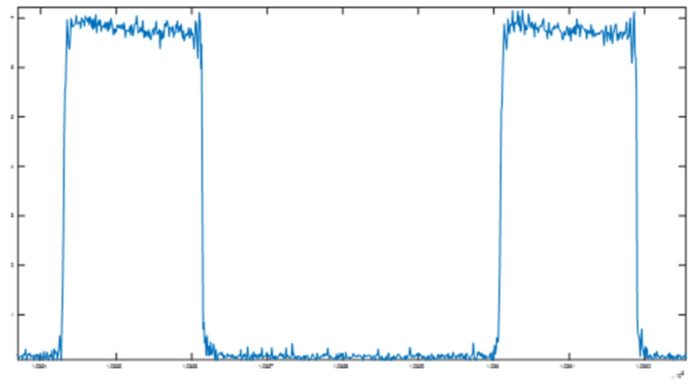


Figure 10. Four sample bits of sync transmission from figure 9.

Once the first data transmission within the first keypress has been studied, ten other pulses remain. From the experiment, it was determined that the information encompassed within each subsequent transmission was repeated without any change, at least not until the keypress was released. These results stand to reason that the key only updates the counter value used for encryption after a keypress is released and simply repeats the message for the duration of the keypress. Figure 11 illustrates a sample of one of these transactions, in this case the first one captured in the sequence. It can be seen that the data further splits into a short synchronization pattern followed by valid data characterized by varying spacing in-between transitions.

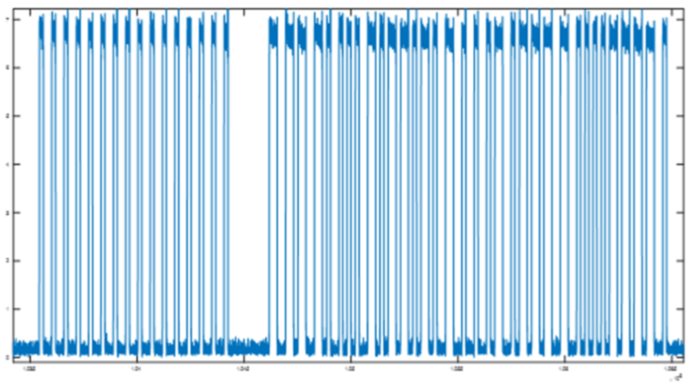


Figure 11. Repeated message contents. Second pulse from figure 8.

Using the spacing determined from characteristic analysis of the synchronization pulse, the data in figure 11 can be properly sampled with representative points relative to the transmission speed. Such an approach eliminates the requirement of hard-coded timings for a specific protocol and should theoretically just adapt to most vendor specific implementations. Re-sampling with characteristic points and discarding the unnecessary data in-between, the original information can be represented with less than 300 times the data points. The re-sampled pulse from the previous transmission is provided in figure 12.

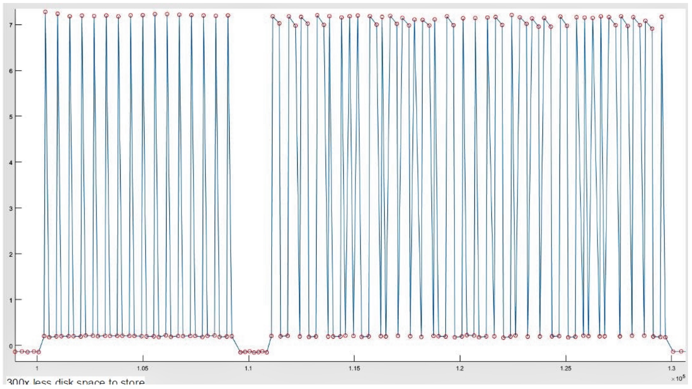


Figure 12. Sampling figure 11 based on sync pulse timing.

The last stage of data processing involves final quantization of the data points. A threshold amplitude value determines the conversion of each point to the binary domain. The total information contained within all five recorded keypresses is shown in figure 13. This represents all valid data contained within the five recorded transactions using only 36.6 thousand binary points as compared to the original 11.3 million amplitudes at the start of the process in figure 7.

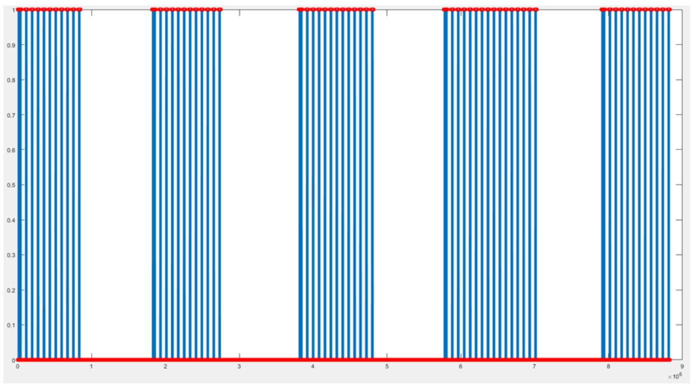
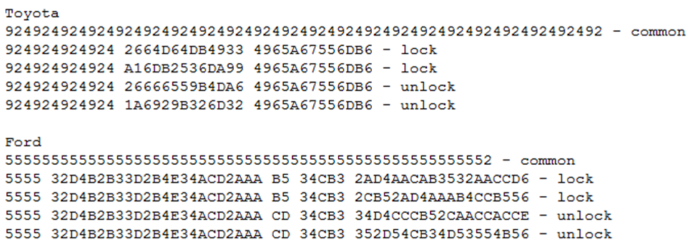


Figure 13. Binary data for all 5 key transmissions from figure 7.

The final phase of the decoding process translates the binary data into hexadecimal representation for analysis. The entire process was repeated for sequences of both lock and unlock keys from both vehicles. Per the analyzed data, figure 14 provides a representative summary of each protocol. It is important to note that the start and stop bits for decoding were chosen arbitrarily with the main goal of comparison between key sequences. The results simply summarize similarities found within recorded data and do not claim to fully disclose proprietary protocol specifications.


```
Toyota
92492492492492492492492492492492492492492492492492492492492492492 - common
924924924924 2664D64DB4933 4965A67556DB6 - lock
924924924924 A16DB2536DA99 4965A67556DB6 - lock
924924924924 26666559B4DA6 4965A67556DB6 - unlock
924924924924 1A6929B326D32 4965A67556DB6 - unlock

Ford
555555555555555555555555555555555555555555555555555552 - common
5555 32D4B2B33D2B4E34ACD2AAA B5 34CB3 2AD4AACAB3532AACCD6 - lock
5555 32D4B2B33D2B4E34ACD2AAA B5 34CB3 2CB52AD4AAAB4CCB556 - lock
5555 32D4B2B33D2B4E34ACD2AAA CD 34CB3 34D4CCCB52CAACCACCE - unlock
5555 32D4B2B33D2B4E34ACD2AAA CD 34CB3 352D54CB34D53554B56 - unlock
```
Figure 14. RKS message contents and protocol comparison.

In the case of the Toyota, the synchronization pattern consisted of a repetition of 1-0-0 leading to the 924 repetition of hex values. On the contrary, the Ford vehicle used a continuous 1-0 pulse creating the sequence of repetitive fives in hexadecimal. While both protocols only update the encryption after a full press and release of the key, the signal behavior during continual transmission was found to differ. Unlike the presented Toyota, the RKS system within the Ford sends the full synchronization pattern before each identical transmission. In other words, in the Toyota protocol the common hex pattern is first sent followed by repetitive transmission of the data packet until the key is released. The Ford protocol includes its hex pattern before each data packet while still maintaining the repetitive nature of the system until the button is released.

Breaking down the content of each message, the shorter synchronization pulse is evident in both manufacturers for all individual commands. The Toyota composes the rest of the message with an encrypted portion and what seems to be the keyfob serial, with each section parsed as 52 bits. The encrypted portion changes on every complete keypress independent of command. We postulate that the command code is fully stored within the encrypted portion and thus the exact button press cannot be simply determined from a captured transmission. In the case of the Ford vehicle, the serial directly follows the short synchronization pattern and the message further breaks down into several distinguishable chunks. Per the analysis, it seems that the next byte contains the plaintext button code followed by another 20 bits of common information. The final block holds 76 bits of encrypted information that changes on each keypress. Further research is necessary into the specific protocol, but preliminary results show a potential for an attacker to utilize the plaintext button code as an additional input vector into the decision process of a real attack.

VIII. Countermeasures

Adding hardware is often difficult when it comes to the production of millions of vehicles. The smallest costs could result multi-millions in lost profits. With that in mind, we present two methods to potentially prevent the replay attack: one cheaper and one more secure.

A. Timer-Based Method

The first method is a timer-based solution. This solution requires two timers on the receiver and the transmitter. The timers are synced during the rolling code synchronization period.

This solution requires that a time is sent from transmitter to the receiver. The receiver then decrypts the message, and compares the vehicle's time to the transmitter's time. If the transmitter's time is valid within an acceptable drift between the receiver, then execute the payload.

This timer-based method prevents the presented replay attack. The victim will send the first key, the attacker will jam the key, the victim will send the next subsequent key, the attacker sends the first. This first key *will* unlock the car. However, the second key that the attacker has in the buffer is no longer valid for later use (due to the time differential).

This method requires the addition of a real time clock on the transmitter and receiver, as well as additional software development for clock synchronization.

B. Modified TCP Handshaking for RF Communication

The second method utilizes an existing 3-way TCP handshake protocol. We modified this protocol slightly to allow for a rolling code encryption.

The handshaking method requires a transmitter and a receiver on both the vehicle and the keyfob. Due to battery constraints, the receiver on the keyfob has to be a low power receiver, and only active after a button is pressed. And due to the range constraints, this receiver must also allow for a medium range transmission message.

We begin by transmitting a packet containing an encrypted sequence number $s1$ and rolling code from the keyfob to the receiver. The receiver will then *acknowledge* the message, sending back the the incremented original sequence number $s1 + 1$ and another sequence number $s2$. Finally, the transmitter will acknowledge the receivers sequence number with $s2 + 1$, and the original sequence number $s1 + 1$. A rolling-code increment will co-occur with each transmission.

This modified TCP handshake will also prevent a replay attack. If the attacker intercepts the any sequence, he will not be able to increment the sequence number without knowledge of the encryption algorithm, thus restricting authentication.

Next-generation systems are specified to include receivers on the keyfobs. The intent is to notify users of important events (such as, low fuel or low tire pressures). This system could theoretically also be used for handling the additional security measures as discussed above.

IX. Conclusion and Future Work

In this paper, we demonstrated a replay attack on the code hopping transmission protocol. This attack exemplifies the importance of a secure protocol. Without a trusted transmission scheme, even some of the most intricate encryption (KeeLoq) can be bypassed without any knowledge of its inner workings. Furthermore, we recorded and analyzed the signals of two manufacturers and recorded several differences in implementation. Lastly, we proposed mechanisms to protect against this style of attack in the future.

X. Contribution Factor

The success of the project hinged on combining the work and expertise of both contributing parties. The research, design, implementation, and data analysis of the project relied on a combination of background knowledge within different fields contributed by each partner. With a computer science background, the high level protocol research, analysis, and software structure expertise was provided by Alfred. Having a background in electrical engineering, Vlad contributed knowledge of RF communication theory. While combined knowledge created the necessary foundation for the project, the project required attaining a significant amount of new knowledge, through both research and testing. As nearly all time spent on the project included both partners, this work was equally distributed. The overall contribution quantifies as follows:

- Alfred Kishek - 50%
- Vlad Slyusar - 50%

References

[1] Andrew Albert. Fm 23-10 chptr 7 communications - field expedient antennas.

[2] Kuehn Gideon J. Bruwer Fredrick J., Smit Willem. Microchips and remote control devices comprising same, 1996. Patent US5517187.

[3] Alfred Centauri. Determine type of modulation, November 2013.

[4] Larry Davis. Electronic engineering glossary terms, February 2012.

[5] Samy Kamkar. Drive it like you hacked it: New attacks and tools to wirelessly steal cars, August 2015.

[6] Inc Maxim Integrated Products. Requirements of remote keyless entry (rke) systems, 2005.

[7] Spencer Whyte. Jam intercept and replay attack against rolling code key fob entry systems using rtl-sdr, March 2014.