Attenuation of Relay Attacks on Passive Keyless Entry

By: Jackson Yuan (yuanja@sheridancollege.ca)

Cristian Di Bartolomeo (dibartoc@sheridancollege.ca)

Luca Di Bartolomeo (dibartlu@sheridancollege.ca)

Julian Lee (leejuli@sheridancollege.ca)

Renee Tyra Evangelista (evangere@sheridancollege.ca)

Faculty of Applied Sciences and Technology

Bachelor of Information Sciences (Cyber Security)

Sheridan College, 1430 Trafalgar Rd., Oakville ON, Canada

August 14, 2023

Academic Advisors:

Ali Hassan (ali.hassan1@sheridancollege.ca)

Pedram Habibi (pedram.habibi@sheridancollege.ca)

George Mikhailov (george.mikhailov@sheridancollege.ca)

Table of Contents

Abstract 3
Problem Statement 4
Literature Review 5
Hypothesis, Planning, and Risk Assessment 9
Solution #1: Distance-Bounding 10
Solution #2: Immobility Detection 11
Associated Risks 11
Implementation and Testing 12
Prototype Creation and Coding Process 13
Testing Methodology 17
GUI Development 19
Observations and Conclusions 20
Areas of Further Study 21
Bibliography and Acknowledgements 23
Appendices 26
Appendix A. Items Used in Hardware Component 26
Appendix B. “car_door_sim.py” Full Code 26
Appendix C. “gui_car.py” Full Code 28
Appendix D. “keyfob.py” Full Code 29
Appendix E. Presentation and Explanation Video 30

Abstract

Majority of modern vehicles widely adopt the Passive Keyless Entry System (PKES) to enhance user convenience and allow drivers to unlock and start their cars effortlessly. However, this convenience introduces vulnerabilities with attackers exploiting the PKES to gain unauthorized access without the authorized key fob. As this technology becomes ubiquitous in modern vehicles, these security concerns amplify. Our capstone project aims to develop a proof-of-concept security solution for the PKES that ensures protection without interfering with its core functionality. Our developed solution effectively strengthens the security of the PKES signal using minimal features and easily available supplies complemented by supporting code. Importantly, this approach offers an easily replicable and cost-effective solution for manufacturers.

Problem Statement

Passive Keyless Entry System (PKES) vehicles were introduced in the early 1980s, and as the years went by its system has become more widespread and available, on its way to becoming the standard system in modern vehicles. In 2010, during its rise in implementation, three students demonstrated a glaring vulnerability in the system by utilizing a relay attack method to copy the message of the car's key fob and utilizing that message to open and start the vehicle, all while the legitimate key fob is not physically near the vehicle while also not alerting either the vehicle or the owner of the vehicle of the attack occurring or its subsequent success [1].

While PKES vehicles had built-in protections to hijacking, such as Radio-Frequency Identification (RFID) being implemented within the vehicle's key fob, this is easily bypassed thanks to the use of relay attacks with [2] detailing the vulnerabilities found in the simplified four-way handshake designed for communication between the key fob and the vehicle. While relay attacks are the most well-known attack surface in PKE hijacking, there are several diverse types of attacks that can be employed as well, such as rollback and roll jam attacks [3] which also utilize the other vulnerabilities in the system.

As the occurrences of attacks rise, some of the proposed simple solutions provided to the public include the use of items such as faraday-style bags to store their key fob in, adding physical barriers to prevent theft of the vehicle, or installing tracking devices to the vehicles that may be affected, but not only are these solutions a blockade to the ease-of-use PKE has allowed in the usage of vehicles, it forces the end user to be responsible for their security when it should be directly implemented at the point-of-sale. All modern cars come equipped with PKES technology, and those that cannot install aftermarket hardware and supplies to become PKES-compatible, but it still retains the same security issues outlined [4].

In early 2023, the ADAC, Europe's largest automobile association, conducted research showing that out of the 567 tested automobiles that utilized PKES, only 29 were not vulnerable to a relay attack [5]. With these vulnerabilities present in all modern vehicles and relay attacks being the most common and well-known attack vector that is still not sufficiently protected against and other attack surfaces that are less used and equally effective, it only shows to emphasize the need for a built-in security implementation designed to prevent the hijacking communication between the key fob and vehicle. Many vehicles function off the same base of keyless entry system technology, thus guaranteeing that attackers will always be able to manipulate their attack regardless of each model's own safety mechanism. Additionally, there is no standardized security protocol for manufacturers to follow regarding the implementation of PKES, so each company may deploy varying levels of security in their key fob, leading to some vehicles or brands being more susceptible to attacks than others. There will be a time when all vehicles will use PKES, introducing more people to potential attacks and broadening the attack surfaces and methods as seen in [6]-[7] due to the insecure nature of the technology, making it imperative to properly secure these systems to ensure the safety of consumers.

Given the current state of automobile security, rather than adding security measures to the vehicle as a secondary measure of protection after the initial breach, there is a clear need for improved security not on the vehicle, but rather the key fob that controls the vehicle's accesses as another layer of preventative security instead.

Literature Review

Since the implementation of PKES in vehicles has become mainstream, there have been several observations, studies, and proposed solutions that have been put out detailing potential countermeasures, designed to minimize or even outright prevent relay attacks against PKES vehicles. However, all these solutions are theoretical, and none have been properly implemented or gone past basic testing, which leaves little idea on how effective some of these solutions truly are when put into practice.

One of the more popular theories is distance-bounding, proposed by [1], [8], [9] as their primary choice. Francillion et al. in particular go into this, detailing that distance-bounding functions by implementing an upper-bound to the physical distance between communicating devices, and has been widely used proximity-based authentication and access control mechanisms. However, this requires accurate measurement of the distance between devices to properly implement an upper-bound, and the distance is “proportional to the time of flight of exchanged messages between the key and the car”, and any delay even in the milliseconds would be capable of causing large issues in the communication, with the risks increasing in case of an attack mid-communication, and thus would not work as a stand-alone solution as it comes with too many potentials for error and added time calculations [1]. Additionally, Yang et al. [8] examined existing distance-bounding protocols and found issues such as extra memory requirements for the communication challenge-and-response, and memory space being wasted, which had the potential to cause synchronization loss between the key and the car.

Upon further research into the feasibility of distance-bounding, we found that it was prone to being attacked and exploited as a security mechanism. Cremers et al [10] details three pre-existing attacks known as “distance fraud”, “mafia fraud”, and “terrorist fraud”, and outline a new attack they refer to as “distance hijacking”. Distance hijacking is defined as an attack where the attacker manages to convince the verifier that they are within the upper-bound by using the answer calculated by the legitimate prover, as all it would require would be to replace the honest prover’s signatures with the attackers. Additionally, Chotia et al. [11] analyze how some distance-bounding attacks function and conduct their own tests for each of the listed attacks and see how the attackers were successful and what they exploited. As it stands, while distance-bounding is the go-to solution to prevent relay attacks, it is also riddled with vulnerabilities and cannot be used as a standalone security mechanism as it can be easily bypassed.

Valko [12] presents a multitude of potential countermeasures against relay attacks such as coordinate tracing to position the key fob and compare it to the vehicle, GPS location verification using the vehicle’s on-board computer, and spatial proximity calculations using wi-fi access points, and goes into detail about the feasibility of the solution’s efficacy, its ease of implementation, and if any potential vulnerabilities exist in the solution. The predominant solution forwarded by Valko would be immobility detection, which is a security mechanism that has been used to enhance existing security measures and prevent unauthorized access by triggering an alert or protective reaction if a device or user has been immobile for a specified period. Some situations immobility detection has been used in include automatic workstation locking, re-authentication requirements, and logging users out of websites with sensitive data. While there is already a level of immobility implemented in PKES vehicles, it only exists as a way to stop the vehicle from moving and leaving if the key is not inside the vehicle, done through a process of authentication the proximity of the transponder inside the key fob to the transponder of the vehicle [13], Valko would detail a physical implementation of PKES and performs testing by implementing the detection mechanism within the key fob instead of the vehicle, showcasing its feasibility and difference in performance and functionality.

Additionally, there is also research that proves immobility-detection, especially the implementation in vehicles so far, can be exploited and has been reverse-engineered [13]. Though the testing and exploitation has been specifically noted to be effective on DST80 -based immobilizers, the same principles and concepts used to reverse-engineer this can be applied to other immobilizers, and since some vehicle manufacturers share the same parent company, they may share the same mechanisms and vulnerabilities, so an attacker could exploit that shared connection to reverse-engineer one brand and be able to attack two. Some of the attacks detailed by [13] are downgrade attacks, which involve reducing the 80-bit key length to 40 bits, effectively reducing the computational complexity by almost half. Another attack is denial-of-service (DoS) which could outright prevent the communication between key fob and vehicle to occur. This shows that while the built-in immobility detection in vehicles does work, it only works under specific circumstances, can be circumvented entirely, and would only slow an attacker down.

Another theoretical solution listed by Park and Hong [14] details utilizing modern technology referred to as “BackProx”. They describe it as a secure proximity detection system that protects the system from relay attacks by taking communication signals aimed towards the vehicle and passing that through a verifier, which then decides if it would pass or fail the signal in relation to the proximity detection. Their solution requires the implementation of an external backscatter tag within the vehicle to act as another method to act as evidence for the proximity distance. The process would then be doing comparisons with the distances between the backscatter tag, the verifier, and the signal requester. Thanks to the backscatter tag acting as another checkpoint for verification, this solution successfully makes it more difficult for relay attacks to occur as they require passing by multiple verifications. However, this solution relies on the backscatter tag and the actual BackProx hardware, two external devices, to be installed inside the vehicle itself, so it would still leave the key fob vulnerable to being exploited at the source.

Furthermore, Joo et al. [15] propose a solution that uses radio frequency (RF) fingerprinting called “Hold the Door!” (HODOR). RF fingerprinting is a technique used to identify wireless devices based on their unique radio frequency characteristics and is often used to enhance network security and improve device authentication and has been used in situations such as device identification and tracking, access control, and rogue device detection. HODOR would use RF fingerprinting to extract the fingerprints of RF devices and their associated signals and use the extracted data to differentiate between legitimate signal requests against the malicious impersonations. To accompany their solution, they also built a prototype that compared peak frequency, signal-to-noise ratio, kurtosis, and spectral brightness which allowed them to effectively test their solution. HODOR is shown to be effective, but has insufficient results to be a practical application as it is, such as extreme temperatures being a factor that could influence and affect the accuracy of the data, or high traffic locations like parking garages and busy streets affecting the comparison data which necessitated increasing the comparison thresholds, which then caused a significant number of false positive results. As it also relies on pre-existing security mechanisms, it should be further researched and tested and adapted to any changes in security before it can be deployed.

Finally, a different solution theorized by Choi et al. [16] was to use a two-factor authentication approach by adding ambient sound detection to the key and vehicle. Sound detection is normally used when audio signals and patterns can be identified and used to respond to security threats. Though not as common as other security methods, it is normally used in physical security detection, honeypot detection to monitor and alert analysts, and IoT (Internet of Things) security. The solution was to record ambient noises separately, and the vehicle would then measure the similarity of the recorded sounds, then calculate the proximity of the key fob based on the results of the ambient sound. They built a small prototype to show its functionality: small microphones would record then compare a short sample of the ambient noise captured to determine if the key and vehicle were within communication distance of each other. They also note that the risks associated with this solution would be environmental factors, such as the potential for noise pollution and interference. While the communication and subsequent noise comparison should not be impacted by the noise, if there are any issues with the initial recording that would be used as the baseline for future verifications, then it could cause issues such as false positives. The implementation also implements a beacon sound the vehicle would emit, and by the researcher's admission would be considered too annoying and obstructing to people within the vicinity, and future iterations of the beacon sound would need to be tuned to a level imperceptible to humans, but still within acceptable calculation speeds. However, they also understand that with this solution, they would be introducing new potential attacks to bypass this security mechanism, such as "out-of-range" and "record-and-playback" attacks, where the attackers would exploit the noise detection mechanism.

Hypothesis, Planning, and Risk Assessment

Relay attacks, while not the only method of attack used, are the most common attack vector used against PKES systems. Relay attacks intercept the communication signal between devices and make them believe they are communicating with each other when the communication has been intercepted and hijacked, allowing for fraudulent transmission of data. To prevent this vulnerability from being exploited, we will need to add additional protective measures that would ensure that the communication cannot be intercepted, while simultaneously ensuring that the built-in encryption deployed by manufacturers is not tampered with and keeping the fast communication time between devices.

Thus, the objective of this project is to implement a method to authenticate the source of the communication signal, protecting the signal and the authentication process between the key fob and the vehicle without interfering with the other implemented security features. To achieve that goal, this project will implement a two-part layered solution utilizing distance-bounding protection, and immobility detection.

Solution #1: Distance-Bounding

Distance-bounding technology is a cybersecurity concept that is designed to prevent attacks focused on physical proximity and distance between two target devices that engage in a communication process. In the realm of cybersecurity, it is often used as an additional layer of protection in authentication, authorization, and access control decisions to ensure that the communication being transmitted is from legitimate devices.

In this project, we will implement distance-bounding as one of the layers of security. Distance-bounding functions by performing round-trip time measurements, measuring the time it takes for a signal to travel between communicating devices, and using the round-trip time measurement it will perform an accurate calculation of the distance of the two devices. Then, when the round-trip time and the distance has been calculated, we can implement a function that provides an upper-bound to the communication time between the devices based on the distance, and later compare that upper-bound with the round-trip time for future communication requests.

By implementing distance-bounding, we ensure that the key fob's functionality of sending signals whenever a button is pressed is not compromised. Once the upper-bound from the normal maximum communication distance is configured, any attempts to increase the intercepting relay attack's distance will not work due to the comparison with the upper-bound calculation, because if the round-trip time exceeds the upper-bound, the device is programmed to reject the communication request.

This was chosen primarily for its ability to prevent long-distance relay attacks, but distance-bounding does come with the issue that it is easy to bypass the upper-bound limit by simply getting within the normal communication distance of the key fob and the vehicle. This can happen in occurrences such as a key fob being placed by the front door of a home close to the target vehicle. To protect against this vulnerability, we will use the second layer of protection, immobility detection.

Solution #2: Immobility Detection

Immobility detection refers to a security mechanism's capability to identify instances where a device has become immobile for an extended time. The purpose of immobility detection is to primarily prevent unauthorized access and session hijacking by detecting suspicious behavior when a device should be or should not be in motion, depending on the circumstances. The way it functions is that a baseline is created to identify normal behavior to determine what is considered normal and abnormal activity. Afterwards, it will trigger an event to make the appropriate actions to safeguard the device.

In this implementation, immobility detection will primarily serve as a method to help protect against the vulnerabilities left behind in distance-bounding by determining when the key fob is immobile, using the lack of movement as the baseline, and when that baseline is satisfied, call the event trigger to act accordingly. The event to be used is a sleep timer, which will put the key fob to sleep and prevent any signals from being received and being sent out to and from the key fob.

This solution was chosen because a person is very likely to be in motion when they are out of the house, such as at a park or at work, and will likely keep their key fob on them, and the distance is more than enough to ensure that the distance-bounding protection will activate to protect relay attacks, while the immobility detection is used in situations such as the key being left immobile at home. When the key is left positioned near the vehicle, immobility detection will activate and put the communicator to sleep. Thus, this ensures there is no signal to capture even when the attacker is close to the devices to bypass the distance-bounding solution.

Associated Risks

The project is not without its risks. Successful implementation will take time and constant modification to best understand how to protect all types of key fobs and automotives for all manufacturers without interference with the communication signal, and the hardware used for the prototype may be incompatible with some types of proprietary hardware, leading to standardization issues. There is also the need to consider the possibility of false positives where legitimate users and communication are denied access, which could cause inconveniences, and the possibility of false negatives where the device fails to properly activate its security mechanisms. Furthermore, there may be signal interferences due to the additional hardware used to implement the security mechanisms code, leading to inaccurate distance measurements or erroneous immobility detection. Finally, as technological improvements are released, both the distance-bounding and the immobility detection hardware and code may require frequent updates to address any emerging threats or changes to the functionality of the code and its libraries to maintain accuracy.

Implementation and Testing

The solution's methodology is to incorporate a combination of the two outlined solutions – distance–bounding and immobility detection – based on the preliminary analysis in the literature review. As car manufacturers will have proprietary hardware and software deployed in their PKES key fobs, we opted not to use a brand name vehicle and key fob for initial implementation and testing designs. Instead, we chose to build our own simulated vehicle and key fob using easily attainable hardware, and use coding and software to implement the outlined security features in the prototype, as well as building the simulated vehicle. This gave us further flexibility in our creation and testing stages, while ensuring we had minimal signal interference from outside data, and we would not interfere with any other signals in the process.

There will be four test cases utilized to provide evidence that in situations where if a single mechanism is implemented the car would be unlockable, whereas with the addition of the second mechanism, the car remains locked despite meeting the conditions of one mechanism to unlock the door. This is to check that the implemented security mechanisms will function separately and see the vulnerabilities within the separate security mechanisms when the other is disabled.

For a list of the specific hardware parts, components, and brands used in the creation of the proof-of-concept, see Appendix A.

Prototype Creation and Coding Process

The project made use of Raspberry Pi's, transceivers, an accelerometer, and a Flipper Zero as part of the signal broadcasting tests. The Flipper Zero plays a key part in that it was used as part of the initial testing of mimicking a replay attack and gathering info on the communication signal. It would once again be used as an attacker to test the implemented solutions. The other pieces of hardware would be used to create the prototype key fob which will have distance-bounding and immobility-detection security coded into it.

Python was chosen as the coding language as it had a multitude of libraries that were suited to our needs to implement distance-bounding and immobility detection and helped us save time by using pre-existing defined functions in the library rather than making our own from scratch. A list of the Python libraries used are:

- **adafruit_adxl34x**
    - The accelerometer library contains code with useful functions allowing communication between the raspberry pi and the accelerometer.
    - This library is used to track movement and speed.
- **RFM69**
    - The Transmitter library contains code with useful functions allowing communication between the raspberry pi and the accelerometer.
    - This library provides a high-level interface that abstracts away the low-level details of the RFM69HCW module. It supports configuring the RFM69HCW module's settings, which was used to send unlock packets and perform distance bounding between the two transmitters.

To connect the accelerometer to the Raspberry Pi, the GPIO pins on the Raspberry Pi are used. The first step to this is to solder pins onto the modules as you will see in Figure #1 below, this process is done for both the accelerometer and transmitter.

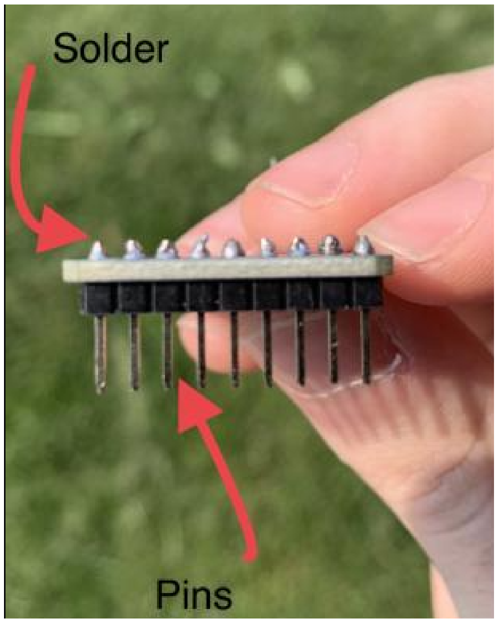


Figure #1: Raspberry Pi Pin Connections

The next step is to connect the pins on the Raspberry Pi to the accelerometer in the following order:

Raspberry Pi	**Accelerometer**
Ground (Pin 6)	GND
3v3 Power (Pin 1)	VCC
GPIO 2 - I2C1 SDA (Pin 3)	SDA
GPIO 3 - I2C1 SCL (Pin 5)	SCL

Table #1: Pin Connections for Accelerometer

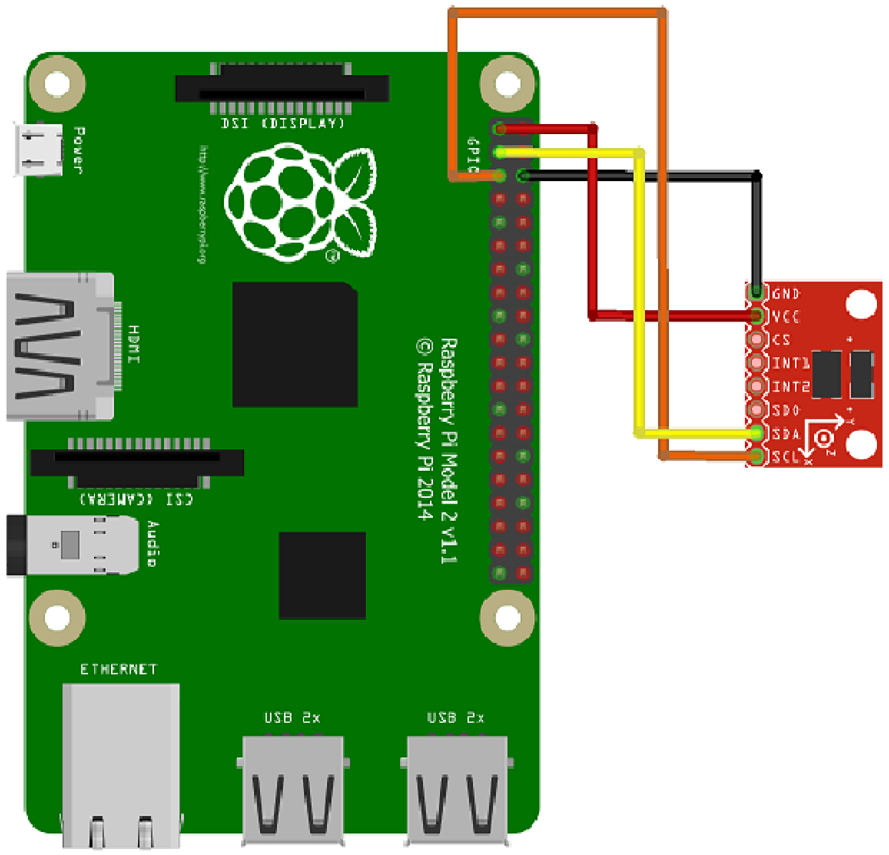


Figure #2: Top-down view of the connections from Table #1

Once the accelerometer was connected, we then connected the transmitter to the Raspberry Pi via the same process but with different pin connections:

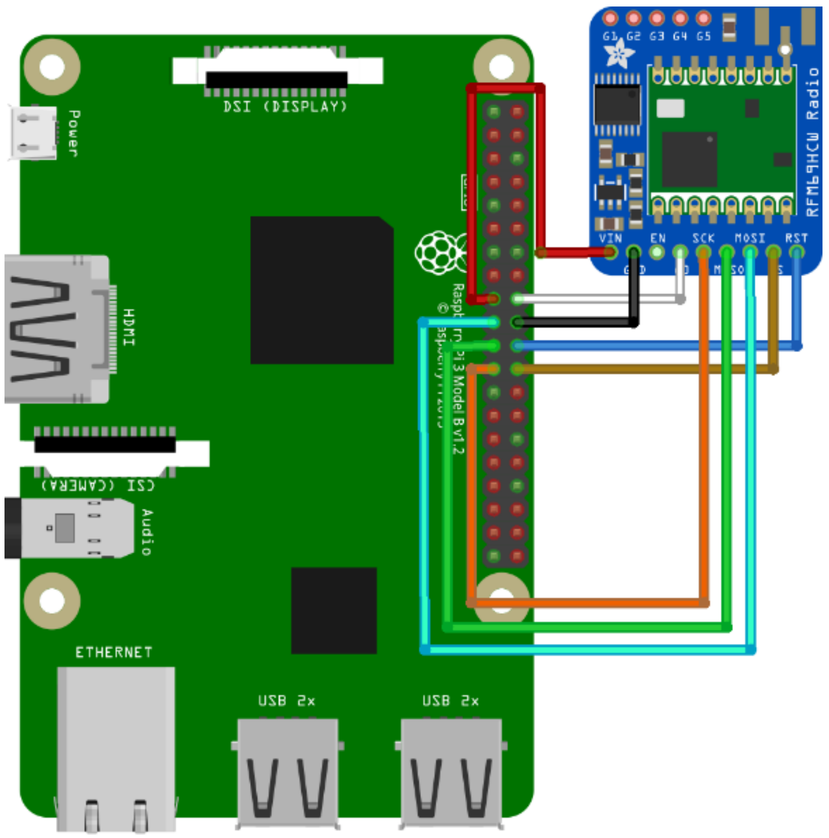


Figure #3: Top-down view of connections from Table #2

Raspberry Pi	**Accelerometer**
3v3 Power (Pin 17)	VIN
Ground (Pin 20)	GND
GPIO 24 (Pin 18)	G0
GPIO 11 - SPI0 SCLK (Pin 23)	SCK
GPIO 9 - SPI0 MISO (Pin 21)	MISO
GPIO 10 SPI0 MOSI (Pin 19)	MOSI
GPIO 8 SPI0 CE1 (Pin 24)	CS
GPIO 25	RST

Table #2: Pin Connections for Transmitter

Testing Methodology

It is important to test that each of our solutions was capable of functioning separately and that they were working when combined together, while also ensuring that the core functionality of the key fob was not changed or disrupted.

To ensure our solutions could work separately and when combined. This was achieved via four specific test cases:

Test Cases			
Test #	**Test Name**	**Description**	**Desired Outcome**
1	KeyOutRangeNoMov	The key fob will be out of range and no movement is detected.	Car remains locked (key fob in sleep mode)
2	KeyInRangeMov	The key will establish connection and user is moving.	Car will unlock (key fob in awake mode. Distance bounding will constantly be checking range of car to key fob)
3	KeyInRangeNoMov	The key will establish connection, but the user is not moving.	Car will remain locked (key fob will be in alert state)
4	KeyOutRangeMov	Key fob out of range but w/ movement	Car remains locked (key fob in sleep mode)

Table #3: Test Cases Developed and Used

Test Case 1: KeyOutRangeNoMov

In the "KeyOutRangeNoMov" test case, we investigate the implementation of immobility detection and distance-bounding in the passive keyless entry system (PKES). The experiment involves placing the key fob out of range from the car and ensuring no movement is detected. The desired result is to observe that the car remains securely locked, indicating that the key fob has entered a sleep mode due to its distance from the vehicle. This test verifies the effectiveness of the system in preventing unauthorized access when the key fob is stationary and beyond the communication range.

Test Case 2: KeyInRangeMov

In the "KeyInRangeMov" test case, we examine the functionality of immobility detection and distance-bounding in PKES under the condition where the key fob is within range of the car and the user is in motion. During the experiment, the key fob establishes a connection with the vehicle, and the user simulates movement. The desired outcome is to observe that the car unlocks promptly, indicating that the key fob is in an awake mode and the distance bounding mechanism continuously checks the range between the car and the key fob to ensure smooth and secure access when the user is in motion within the authorized range. This test validates the system's capability to detect user movement and grant access only when appropriate proximity conditions are met, thereby enhancing the overall security and user experience of the keyless entry system.

Test Case 3: KeyInRangeNoMov

In the "KeyInRangeNoMov" test case, we evaluate the immobility detection and distance-bounding features in PKES when the key fob establishes a connection with the car, but the user remains stationary. Throughout the experiment, the key fob is within range of the vehicle, but no movement is detected. The desired outcome is to observe that the car remains securely locked, signifying that the key fob enters an alert state due to the absence of user movement. This state ensures enhanced security measures by keeping the car inaccessible until authorized movement is detected, validating the system's ability to prevent repeater attack entry when the user remains stationary within the designated range, such as in the case of leaving your key fob by your door when you enter your home.

Test Case 4: KeyOutRangeMov

In the "KeyOutRangeMov" test case, we investigate the effectiveness of immobility detection and distance-bounding in PKES when the key fob is taken out of the vehicle's range, but the user attempts to simulate movement. Throughout the experiment, the key fob is deliberately placed out of range from the car, and user movement is simulated. The desired outcome is to observe that the car remains securely locked, indicating that the key fob remains in sleep mode despite the attempted movement. This result confirms the system's ability to recognize the absence of authorized proximity and disregard movement attempts when the key fob is not within the prescribed range, such as in the case of parking your car running errands with your key fob in your pocket or purse, thereby ensuring robust security measures and preventing unauthorized access.

GUI Development

The graphical user interface (GUI) developed in Python provides a visual representation of the state of the car door (locked or unlocked) and responds to packets received from a Raspberry Pi to simulate real-time changes in the door's state. The GUI was developed using the Tkinter library, a standard Python interface to the Tk GUI toolkit and the Python Imaging Library (PIL); a powerful library for opening, manipulating, and saving many different image file formats. The development process involved several steps:

1. **GUI Setup**: The Tkinter library was utilized to create the main application window. The dimensions of the window were set to comfortably accommodate the GIF animations used to represent the car door's states.

2. **Image Loading:** The PIL library was employed to load the GIF animations and the initial image. The GIF animations visually represent the locking and unlocking of the car door, while the initial image signifies the locked state of the car door at the start of the simulation.

3. **Image Label Creation:** A Tkinter Label widget was used to display the current image. The label was packed with padding to center it within the application window.

4. **Packet Response:** The GUI was designed to respond to packets received from a Raspberry Pi. Upon receipt of a packet indicating a change in the car door's state, a function is called to toggle the lock state and update the displayed image.

5. **GIF Animation:** A custom function was created to animate the GIF images. This function uses the seek method from the PIL library to iterate through the frames of the GIF. The function is recursively scheduled to run every 100 milliseconds, creating the animation effect.

6. **Pop-up Message Display:** A pop-up message is displayed whenever a packet is received that changes the state of the car door. The message is shown in a new top-level window at the center of the screen and automatically disappears after 2 seconds.

Observations and Conclusions

This project's prototype was designed to prove that the combination of immobility detection and distance-bounding in passive keyless entry systems offers substantial benefits that significantly enhance security and user experience. By incorporating immobility detection, the system can discern between stationary key fobs and potential security threats, ensuring that the car remains inaccessible when the key fob is not in motion. Simultaneously, the distance-bounding feature establishes continuous communication between the key fob and the vehicle, enabling real-time monitoring of the proximity. This cooperative approach guarantees that access is granted only when the key fob is within the authorized range *and* in an awake state, theoretically thwarting unauthorized entry attempts such as relay attacks, distance-hijacking, as well as other threats that are effective against each mechanism independently. Therefore, in this project we have demonstrated that the constructive collaboration between these features instills confidence in the system's ability to prevent unauthorized access while also providing seamless and convenient access for authorized users, making it a robust and reliable solution for passive keyless entry systems of the future.

Areas of Further Study

Further exploration in the realm of immobility detection and distance-bounding within passive keyless entry systems offers promising opportunities, particularly when considering advancements in artificial intelligence (AI). On the software front, the integration of AI-driven anomaly detection algorithms could enhance the accuracy of immobility detection, effectively discerning benign activity from potential attacks. Leveraging AI techniques to analyze intricate patterns and user behavior could enable the system to adapt and respond intelligently to emerging threats, and even tailor its detection to individual user profiles [17].

In tandem with software innovations, a crucial facet is the development of secure and cost-efficient hardware components, otherwise risking not incorporating these solutions if they cannot be manufactured simply and cheaply. Focusing on accelerometer integration within key fobs, while optimizing manufacturing processes, could improve tamper resistance and affordability. Hopefully in the future researchers and engineers can create a balance between heightened security and viable cost structures, ensuring that innovations translate effectively to real-world implementation and mass-adoption.

This is where we rely on the principles of Moore's Law, driving exponential growth in computational capabilities. This will be integral to enhancing distance-bounding and immobility detection. As hardware evolves, more complex cryptographic methods and communication protocols can be integrated without compromising efficiency. The future could relate to intricate encryption mechanisms, extended communication ranges, and optimized energy consumption in key fob devices. This research could elevate the holistic security baseline of passive keyless entry systems that could be leveraged for other devices such as garage doors (and any other passive entry doors) and medical devices such as pacemakers.

While we were unable to dive into exploiting our own solution due to time constraints, we hope that in the future, other researchers and white hat car hackers will come up with their own novel vulnerability testing approaches see if our passive keyless entry solution will stand up to the common attacks that are being used on distance bounding and immobility detection separately.

Finally, the next logical step is to partner with existing car manufacturers and integrate our solution with their proprietary technologies, transitioning from our proof-of-concept demonstration to physical implementation and rigorous in-house testing, leading the way for eventual market release.

Bibliography and Acknowledgements

[1] A.Francillon, B. Danev and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars", *NDSS*. 2011. doi: 10.3929/ETHZ-A-006708714

[2] J. Ashworth, J. Staggs, and S. Shenoi, "Radio Frequency Identification and tracking of vehicles and drivers by exploiting Keyless Entry Systems," *International Journal of Critical Infrastructure Protection*, vol. 40, p. 100587, 2023

[3] Csikor, L., Lim, H. W., Wong, J. W., Ramesh, S., Parameswarath, R. P., & Chan, M. C. "RollBack: A New Time-Agnostic Replay Attack Against the Automotive Remote Keyless Entry Systems", ArXiv (Cornell University). 2022. [Online] Available: https://doi.org/10.48550/arxiv.2210.11923

[4] M. Wallaker, "What Is Keyless Entry and Which Cars Have It?," *MUO*, Mar. 05, 2022. https://www.makeuseof.com/what-is-keyless-entry-which-cars/. Accessed July 25, 2023.

[5] "Keyless theft continues: Even new cars are easy to crack," ADAC, 28 February, 2023. [Online]. Available: https://assets.adac.de/image/upload/v1677579204/ADAC-eV/KOR/Text/PDF/31639_ptp7nn.pdf. Accessed: July 25, 2023.

[6] A. Greenberg, "Hackers Remotely Kill a Jeep on the Highway—With Me in It," WIRED article, https://bit.ly/3AJLhjn, 2015. Accessed: April 06, 2023.

[7] "FREE-FALL: HACKING TESLA FROM WIRELESS TO CAN BUS," Presentation at BlackHat, https://bit.ly/3FZyRGx, 2017.

[8] T. Yang, L. Kong, W. Xin, J. Hu and Z. Chen, "Resisting relay attacks on vehicular Passive Keyless Entry and start systems," *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, 2012, pp. 2232-2236, doi: 10.1109/FSKD.2012.6234155

[9] K. Joo, W. Choi, and D. H. Lee, “Hold the Door! Fingerprinting Your Car Key to Prevent Keyless Entry Car Theft,” arXiv:2003.13251 [cs, eess], Mar. 2020, Available: https://arxiv.org/abs/2003.13251

[10] C. Cremers, K. B. Rasmussen, B. Schmidt and S. Capkun, "Distance Hijacking Attacks on Distance Bounding Protocols," *2012 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2012, pp. 113-127, doi: 10.1109/SP.2012.17.

[11] T. Chotia, J. de Ruiter, B. Smyth, “Modelling and Analysis of a Hierarchy of Distance Bounding Attacks,” *USENIX Security Symposium*, pp. 1563–1580, Aug. 2018, doi: 10.5555/3277203.3277320, [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-chothia.pdf

[12] A. Valko, "Relay attack resistant passive keyless entry: securing PKES systems with immobility detection," *KTH Royal Institute of Technology*, 2020, doi: 10.13140/RG.2.2.14835.45600

[13] L. Wouters, J. Van den Herrewegen, F. D. Garcia, D. Oswald, B. Gierlichs, and B. Preneel, “Dismantling DST80-based Immobiliser Systems”, *TCHES*, vol. 2020, no. 2, pp. 99–127, Mar. 2020.

[14] H. Park and J. Hong, “BackProx: Secure Backscatter-Assisted Proximity Detection for Passive Keyless Entry and Start Systems,” *Sensors*, vol. 23, no. 4, p. 2330, Feb. 2023, doi: https://doi.org/10.3390/s23042330.

[15] K. Joo, W. Choi, and D. H. Lee, “Hold the Door! Fingerprinting Your Car Key to Prevent Keyless Entry Car Theft,” arXiv:2003.13251 [cs, eess], Mar. 2020, Available: https://arxiv.org/abs/2003.13251

[16] W. Choi, M. Seo, and D. H. Lee, “Sound-Proximity: 2-Factor Authentication against Relay Attack on Passive Keyless Entry and Start System,” Journal of Advanced Transportation, vol. 2018, pp. 1–13, 2018, doi: https://doi.org/10.1155/2018/1935974.

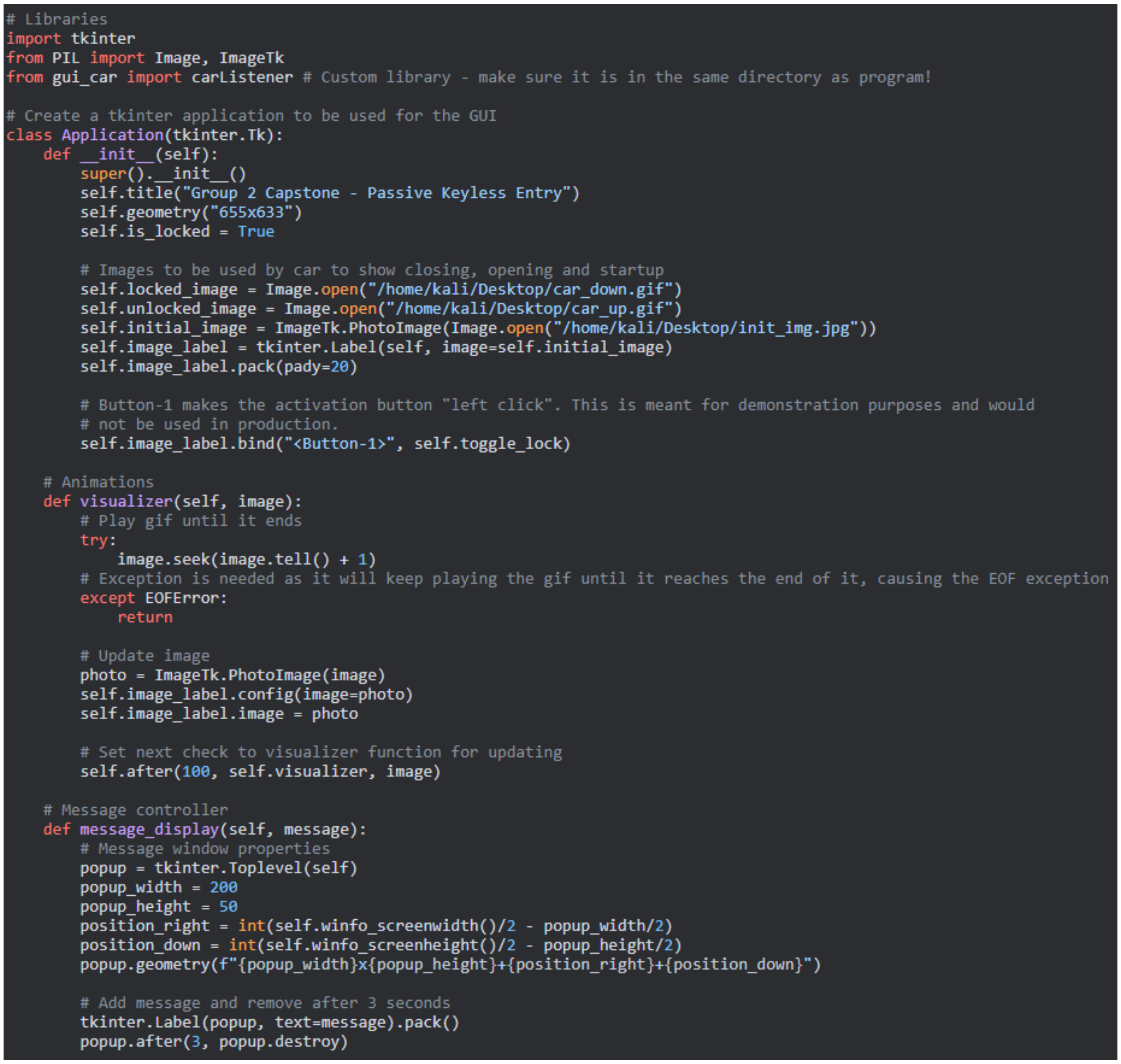
[17] S. Rizvi, J. Imler, L. Ritchey, and M. Tokar, “Securing PKES: Integrating Artificial Intelligence Into PKES Forcefield For Anomaly Detection,” *IEEE Xplore*, Jul. 01, 2019. https://ieeexplore.ieee.org/abstract/document/9307677.

Appendices

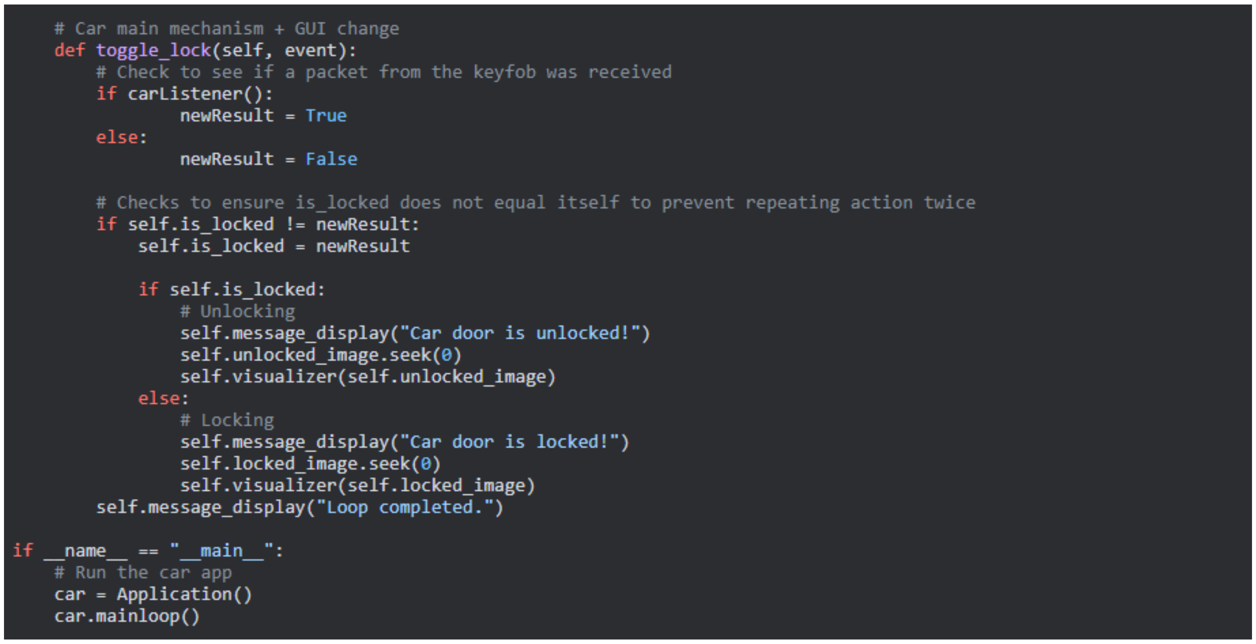
Appendix A. Items Used in Hardware Component

- Raspberry Pi
- Adafruit RFM69HCW Transceiver Radio Breakout - 915 MHz
- Geekstory 915MHz LoRa Antenna IPX IPEX 1.13 UF.L Antenna

Appendix B. "car_door_sim.py" Full Code

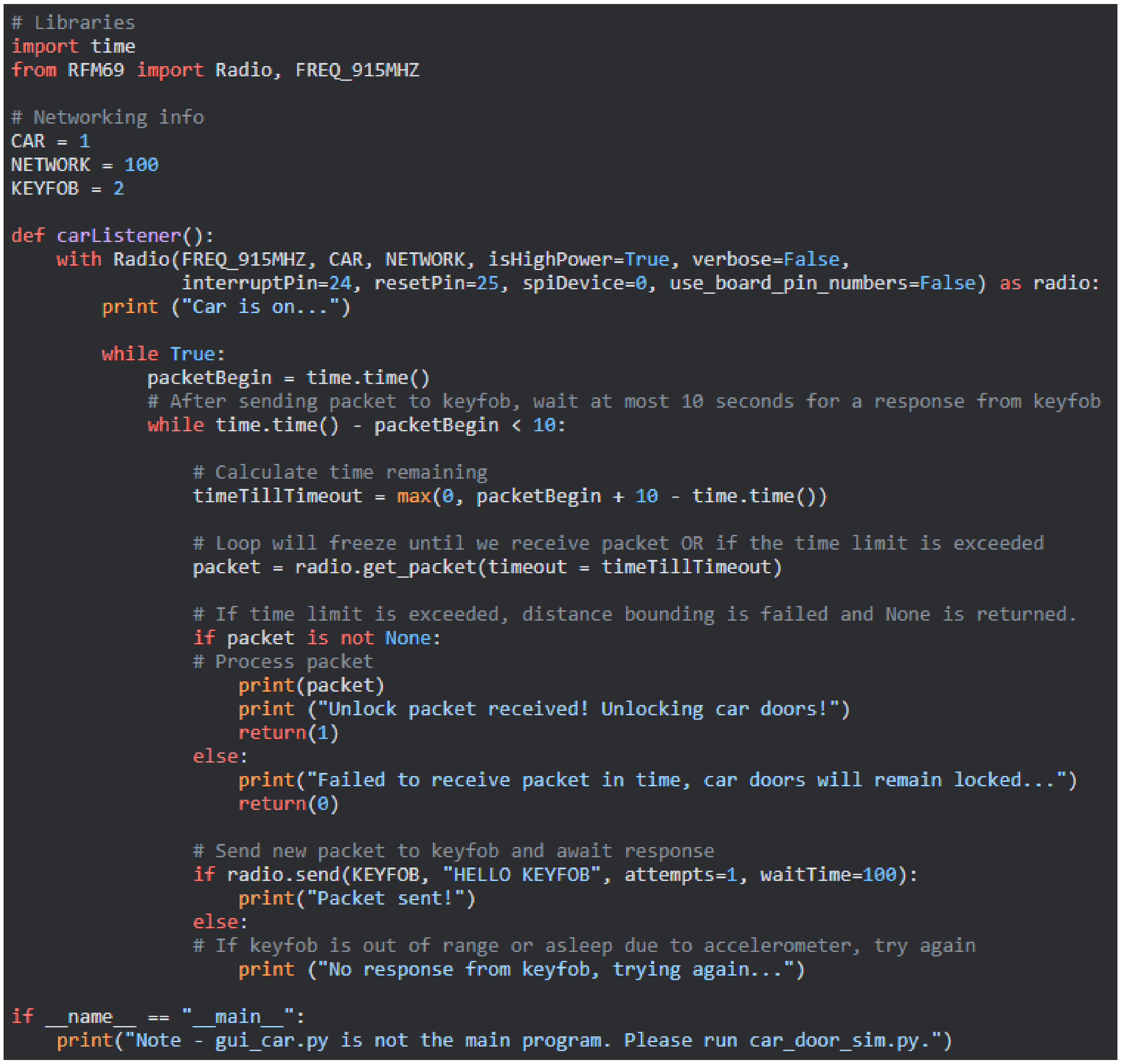


```python
# Libraries
import tkinter
from PIL import Image, ImageTk
from gui_car import carListener # Custom library - make sure it is in the same directory as program!

# Create a tkinter application to be used for the GUI
class Application(tkinter.Tk):
    def __init__(self):
        super().__init__()
        self.title("Group 2 Capstone - Passive Keyless Entry")
        self.geometry("655x633")
        self.is_locked = True

        # Images to be used by car to show closing, opening and startup
        self.locked_image = Image.open("/home/kali/Desktop/car_down.gif")
        self.unlocked_image = Image.open("/home/kali/Desktop/car_up.gif")
        self.initial_image = ImageTk.PhotoImage(Image.open("/home/kali/Desktop/init_img.jpg"))
        self.image_label = tkinter.Label(self, image=self.initial_image)
        self.image_label.pack(pady=20)

        # Button-1 makes the activation button "left click". This is meant for demonstration purposes and would
        # not be used in production.
        self.image_label.bind("<Button-1>", self.toggle_lock)

    # Animations
    def visualizer(self, image):
        # Play gif until it ends
        try:
            image.seek(image.tell() + 1)
        # Exception is needed as it will keep playing the gif until it reaches the end of it, causing the EOF exception
        except EOFError:
            return

        # Update image
        photo = ImageTk.PhotoImage(image)
        self.image_label.config(image=photo)
        self.image_label.image = photo

        # Set next check to visualizer function for updating
        self.after(100, self.visualizer, image)

    # Message controller
    def message_display(self, message):
        # Message window properties
        popup = tkinter.Toplevel(self)
        popup_width = 200
        popup_height = 50
        position_right = int(self.winfo_screenwidth()/2 - popup_width/2)
        position_down = int(self.winfo_screenheight()/2 - popup_height/2)
        popup.geometry(f"{popup_width}x{popup_height}+{position_right}+{position_down}")

        # Add message and remove after 3 seconds
        tkinter.Label(popup, text=message).pack()
        popup.after(3, popup.destroy)
```



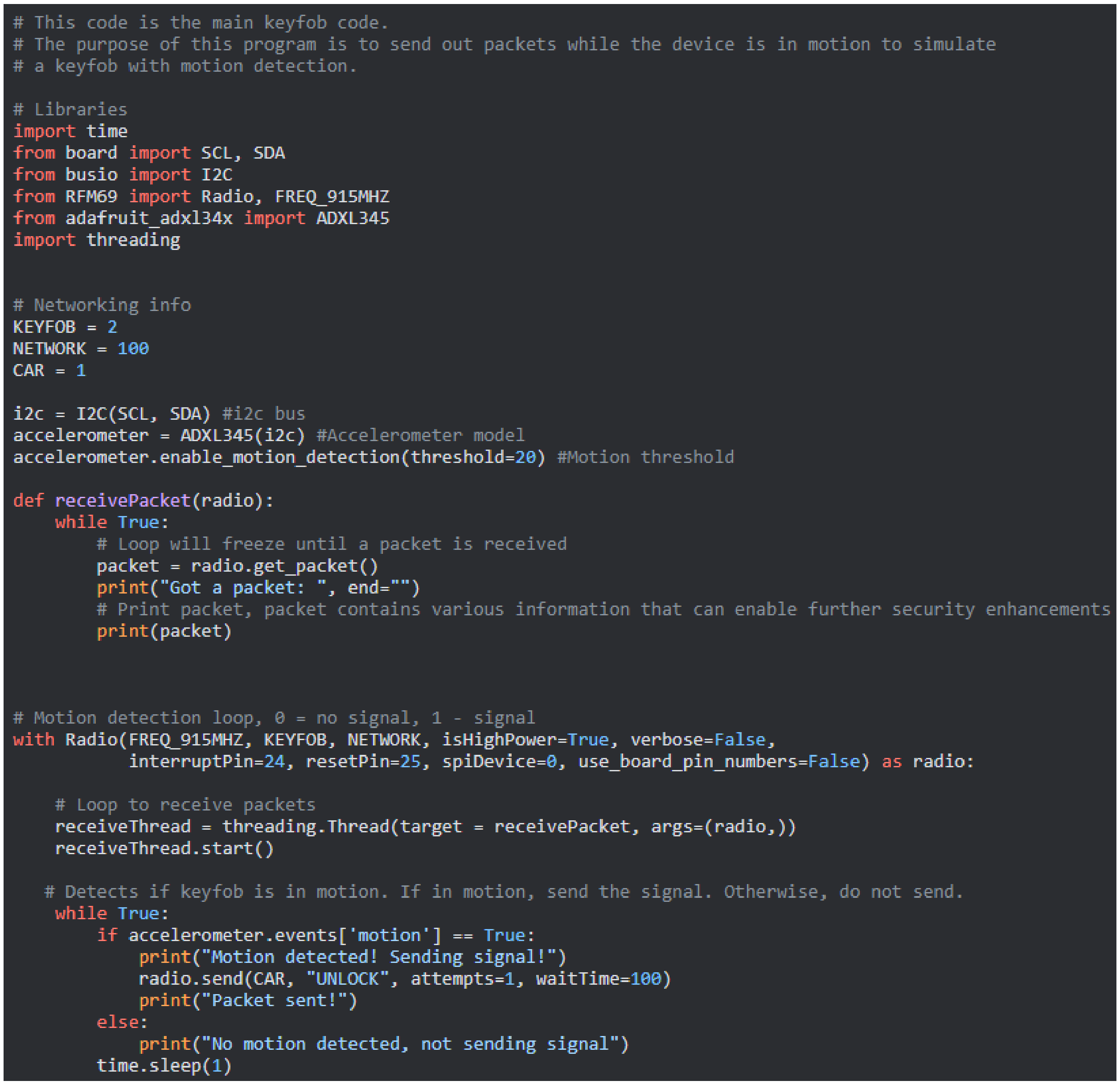
```python
    # Car main mechanism + GUI change
    def toggle_lock(self, event):
        # Check to see if a packet from the keyfob was received
        if carListener():
                newResult = True
        else:
                newResult = False

        # Checks to ensure is_locked does not equal itself to prevent repeating action twice
        if self.is_locked != newResult:
            self.is_locked = newResult

            if self.is_locked:
                # Unlocking
                self.message_display("Car door is unlocked!")
                self.unlocked_image.seek(0)
                self.visualizer(self.unlocked_image)
            else:
                # Locking
                self.message_display("Car door is locked!")
                self.locked_image.seek(0)
                self.visualizer(self.locked_image)
        self.message_display("Loop completed.")

if __name__ == "__main__":
    # Run the car app
    car = Application()
    car.mainloop()
```

Appendix C. “gui_car.py” Full Code



```python
# Libraries
import time
from RFM69 import Radio, FREQ_915MHZ

# Networking info
CAR = 1
NETWORK = 100
KEYFOB = 2

def carListener():
    with Radio(FREQ_915MHZ, CAR, NETWORK, isHighPower=True, verbose=False,
               interruptPin=24, resetPin=25, spiDevice=0, use_board_pin_numbers=False) as radio:
        print ("Car is on...")

        while True:
            packetBegin = time.time()
            # After sending packet to keyfob, wait at most 10 seconds for a response from keyfob
            while time.time() - packetBegin < 10:

                # Calculate time remaining
                timeTillTimeout = max(0, packetBegin + 10 - time.time())

                # Loop will freeze until we receive packet OR if the time limit is exceeded
                packet = radio.get_packet(timeout = timeTillTimeout)

                # If time limit is exceeded, distance bounding is failed and None is returned.
                if packet is not None:
                # Process packet
                    print(packet)
                    print ("Unlock packet received! Unlocking car doors!")
                    return(1)
                else:
                    print("Failed to receive packet in time, car doors will remain locked...")
                    return(0)

                # Send new packet to keyfob and await response
                if radio.send(KEYFOB, "HELLO KEYFOB", attempts=1, waitTime=100):
                    print("Packet sent!")
                else:
                # If keyfob is out of range or asleep due to accelerometer, try again
                    print ("No response from keyfob, trying again...")

if __name__ == "__main__":
    print("Note - gui_car.py is not the main program. Please run car_door_sim.py.")
```

Appendix D. “keyfob.py” Full Code



```
# This code is the main keyfob code.
# The purpose of this program is to send out packets while the device is in motion to simulate
# a keyfob with motion detection.

# Libraries
import time
from board import SCL, SDA
from busio import I2C
from RFM69 import Radio, FREQ_915MHZ
from adafruit_adxl34x import ADXL345
import threading


# Networking info
KEYFOB = 2
NETWORK = 100
CAR = 1

i2c = I2C(SCL, SDA) #i2c bus
accelerometer = ADXL345(i2c) #Accelerometer model
accelerometer.enable_motion_detection(threshold=20) #Motion threshold

def receivePacket(radio):
    while True:
        # Loop will freeze until a packet is received
        packet = radio.get_packet()
        print("Got a packet: ", end="")
        # Print packet, packet contains various information that can enable further security enhancements
        print(packet)



# Motion detection loop, 0 = no signal, 1 - signal
with Radio(FREQ_915MHZ, KEYFOB, NETWORK, isHighPower=True, verbose=False,
           interruptPin=24, resetPin=25, spiDevice=0, use_board_pin_numbers=False) as radio:

    # Loop to receive packets
    receiveThread = threading.Thread(target = receivePacket, args=(radio,))
    receiveThread.start()

   # Detects if keyfob is in motion. If in motion, send the signal. Otherwise, do not send.
    while True:
        if accelerometer.events['motion'] == True:
            print("Motion detected! Sending signal!")
            radio.send(CAR, "UNLOCK", attempts=1, waitTime=100)
            print("Packet sent!")
        else:
            print("No motion detected, not sending signal")
        time.sleep(1)
```

Appendix E. Presentation and Explanation Video

Attenuation of Relay Attacks on PKES - Capstone Final Presentation