Phase #1 – Secure RF Rolling Codes

CSE321 – Real-time Embedded Operating Systems

Prepared by: Kenneth Pang, kpang
University at Buffalo
School of Engineering and Applied Sciences

September 16, 2025

Table of Contents

Problem Statement

Real-World Scenario

In today’s world, hardware communication security has many weaknesses ranging from replay attacks, wiretapping, eavesdropping, keyloggers, and even physical entry. In the automotive industry, they build key remotes to communicate with the car to unlock and lock itself. However, one must ask how do they keep it secure from an attacker? A common and simple way to gain access as an attacker is to create a replay attack. The solution that was used for decades was called the Rolling code, also known as the Hopping codes.

First, we must define a replay attack to understand the problem a little more. Let's say a malicious attacker with intent wants to gain access to something of yours (like a car). An unencrypted remote car key could send an unencrypted signal when you press the unlock button to your car. The attacker could then record this signal from your key and mimic the signal for the next time, so they would be able to unlock your car system. This is a replay attack, essentially recording an unencrypted or weak encrypted message, so the signal can be replayed to mimic the user, and to gain unrightful access.

In the automotive industry, early security engineers used the method of rolling codes and encryption algorithms to keep a semi-perfect system to encrypt messages to send to a car. The car would then be able to pick up those codes to determine if the person was an authorized user of the system. This method has vulnerabilities that could still be exploited but is an early implementation to secure embedded systems and implores others to understand and pursue more knowledge within the security domain for embedded systems.

The rolling codes algorithm basically includes two internal counters set in both the key remote and the car system. When the key remote command is clicked, the key remote increments a semi-randomized code, the code gets encrypted using an encryption algorithm like AES. The command is appended to the encrypted message and sent using an RF transmitter. The car system is supposed to pick up those signals using an RF receiver which is then decrypted and checks if the code is valid and within range of the acceptable codes. If the code is valid, the car system performs the specified command and increments the counter to be synced with the remote key. However, if the code is invalid, the car should not perform the command.

Project Goal

The goal of this project is to simulate the rolling codes with encryption algorithms and create a secure hardware system onto an Arduino board. Unfortunately, we will not be creating an entire car system and making it secure, so we will be replacing each command with a smaller scaled idea. For the unlocking and locking doors, we will be using an LED, with an on as locked state and off for an unlocked state. The sounding alarm will be a buzzer that produces a noise to alert the car owner. The final command of the trunk automatically opening will be replaced with a servo motor that will try to open a tiny cardboard box door, or a tiny 3D printed door.

We will have two separate systems, one that will do the transmitting of radiofrequency signals. We want the transmitter to be a soft real-time system, so timing is not very important in this system. There are four commands, and a user should be able to press a button to transmit the type of command with an encrypted key that will be handled by Arduino. The transmitter will take a signal and transmit the signal over radiofrequency. We will also probably use an LED to denote that a signal was transmitted by the Arduino.

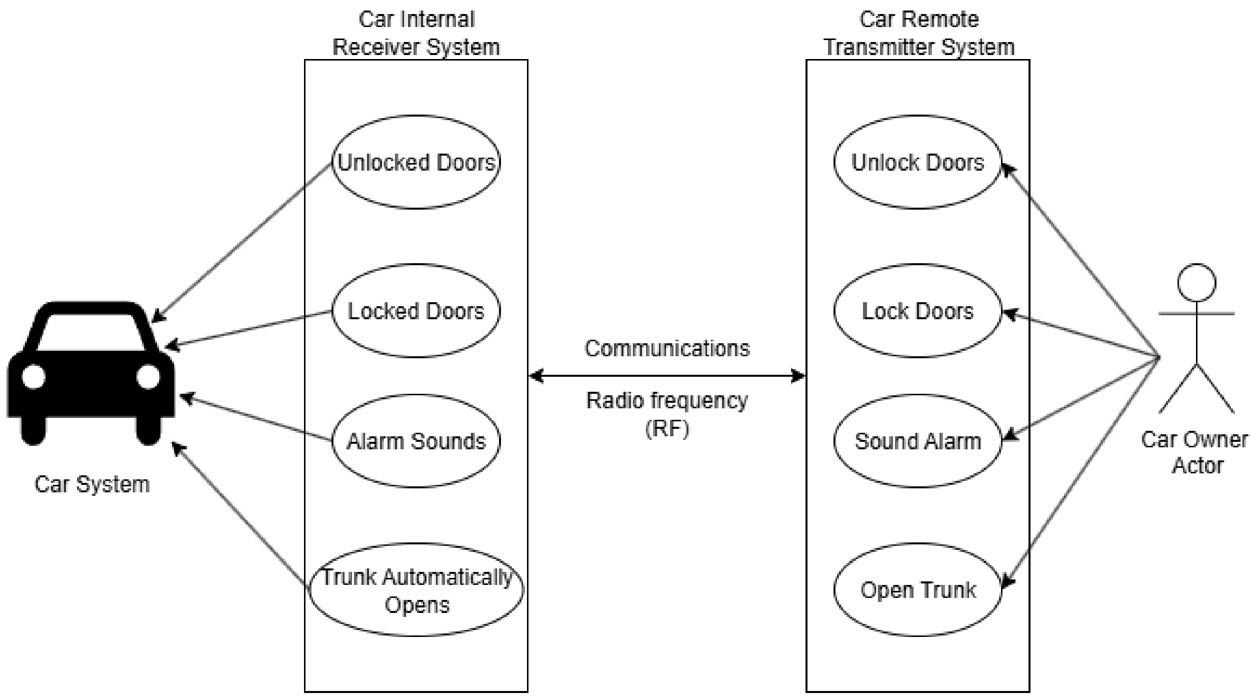
The other system will be the one that receives the radiofrequency signal. This receiver system will be more of a hard real-time system, timing is very important because if the transmitted signal is not captured in time or not captured at all, then our counter code will not be in sync and have significant loss of function within our secure system. This will lead to our systems to be in different counter codes, so our transmitter signal will no longer work with our receiver. This receiver system will decrypt the signal and perform the operation that it was given by the transmitter.

We want both of our systems to be reliable and working 24/7, so signals can be transmitted and received when the user is requesting commands on our embedded system. Our buttons should register within 100ms onto the Arduino, and our transmitter and receiver should be sending signals within 500ms to 2s. The transmitter Arduino should be able to compute the encrypted key with command of the next signal after our transmitter sends a signal. The receiver Arduino should be able to decrypt the key and compare it to determine if the key is valid. The LED should be reliable and display the correct state for unlocked and locked. Our buzzer should make a “beep” or noise for 300ms and stop for 200ms for a total of 5 times. Lastly, our servo to open the door should be able to swing 90° so that our tiny door is accessible to anyone needing to manually open it.

Use Case Diagram (UML)

Normal Usage

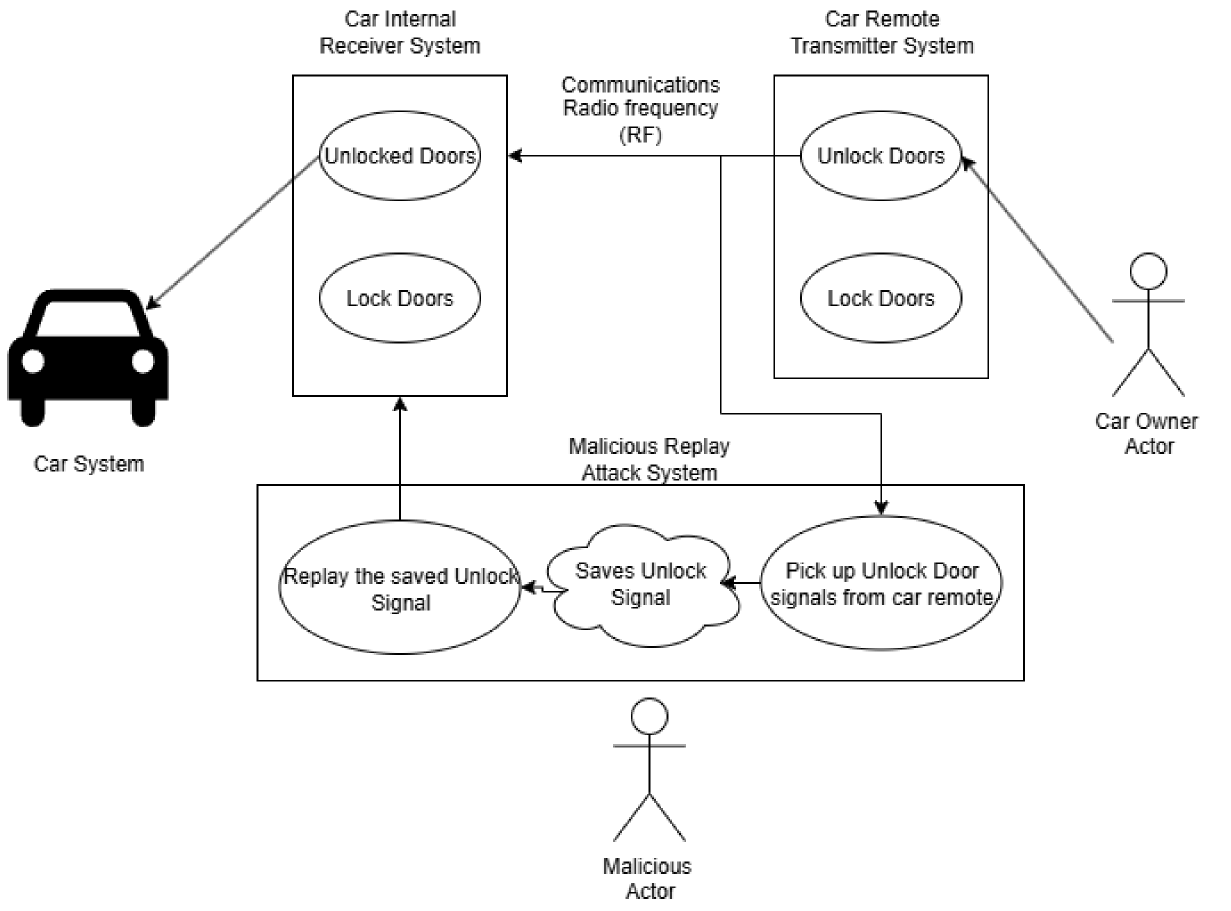
In a normal case, we would see an actor, or the car owner who will own a tiny car remote embedded system to control their car system. Usually, this involves 3-4 features that allow the user to control and gain access to the car. The car should only be accessible to the car owner, since it is their property and left untampered by someone else.



Security Threat

In a security threat case, we would see two actors, one which is the original car owner, and the other who will be the malicious attacker actor. We removed some commands to make this easier to picture. In this case, we would see the normal car owner play an unencrypted signal to unlock the car. However, the malicious attacker would have their own system to pick up the door signal, save it, and then replay the signal when it is appropriate. This would mean that both the car owner and the malicious attacker have access to the car. The malicious attacker could then steal from the owner or do unrightful actions against the car owner.

Our goal specifically is to stop this replay attack with an encrypted key and verification of the key on the car internal system. This encrypted key will increment with a pseudo random generator, so even if the signal is picked up by the malicious attacker, the encrypted key would have changed on both the transmitter and receiver side.

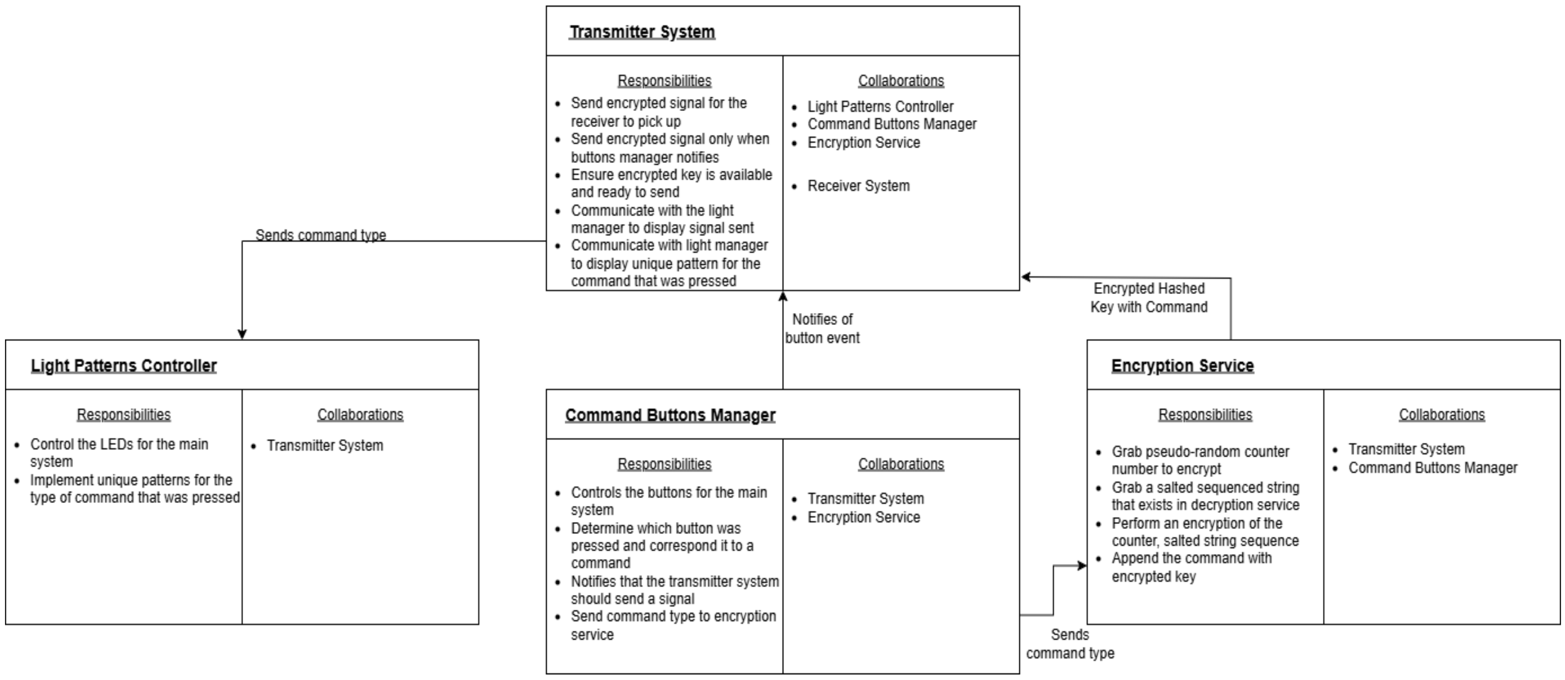

Car Internal Receiver System
Unlocked Doors
Lock Doors
Communications Radio frequency (RF)
Car Remote Transmitter System
Unlock Doors
Lock Doors
Car System
Car Owner Actor
Malicious Replay Attack System
Replay the saved Unlock Signal
Saves Unlock Signal
Pick up Unlock Door signals from car remote
Malicious Actor

Modules of Project (CRC)

We have two modules of this project, and we have classified them into Class, Responsibilities, and Collaboration cards. These systems are just an idea of what we will be creating and how each hardware interfaces with each other, what their responsible of keeping track, what actions will they perform to keep our system secure, etc.

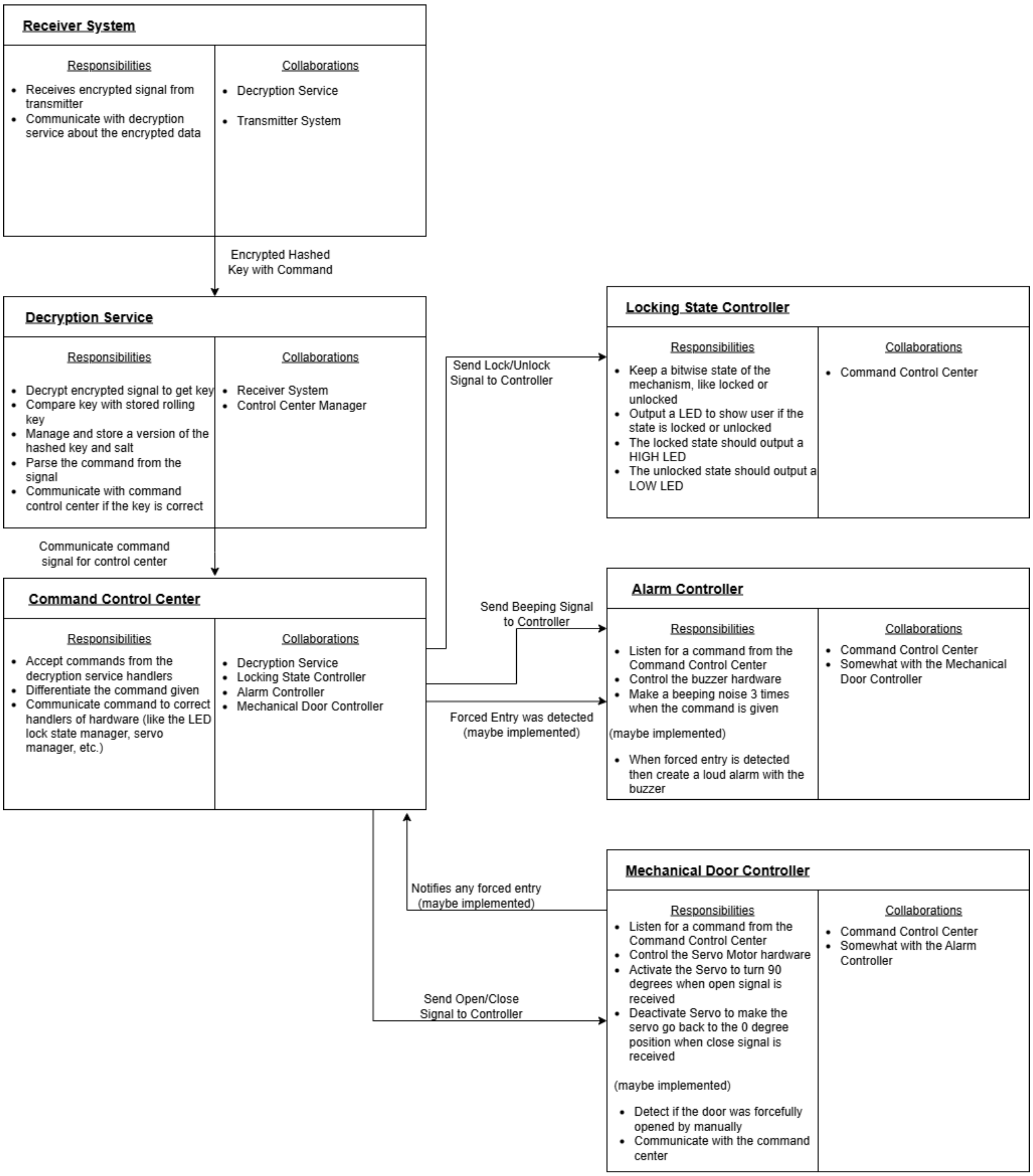
Key Remote System

Here, we will be implementing the transmitter side of our project. First, we will have buttons on our remote, this will be all under the class "Command Buttons Manager," they will determine which button was pressed, the command associated with the button and send the command to the "Encryption Service." The service handles most of the encryption algorithm, rolling code counter, salted string (stored in both systems), and then sending the encrypted signal to the main "Transmitter System". The buttons will notify the transmitter system to prepare to send a message through radiofrequency, and when the encrypted signal is given by the encryption service, the signal should be sent. Finally, after the signal is sent from the transmitter, we want the "Light Patterns Controller" to control an LED that most likely blink in a pattern to show that the signal is sent to the user.


Transmitter System
Responsibilities
Send encrypted signal for the receiver to pick up
Send encrypted signal only when buttons manager notifies
Ensure encrypted key is available and ready to send
Communicate with the light manager to display signal sent
Communicate with light manager to display unique pattern for the command that was pressed
Collaborations
Light Patterns Controller
Command Buttons Manager
Encryption Service
Receiver System
Sends command type
Notifies of button event
Encrypted Hashed Key with Command
Light Patterns Controller
Responsibilities
Control the LEDs for the main system
Implement unique patterns for the type of command that was pressed
Collaborations
Transmitter System
Command Buttons Manager
Responsibilities
Controls the buttons for the main system
Determine which button was pressed and correspond it to a command
Notifies that the transmitter system should send a signal
Send command type to encryption service
Collaborations
Transmitter System
Encryption Service
Sends command type
Encryption Service
Responsibilities
Grab pseudo-random counter number to encrypt
Grab a salted sequenced string that exists in decryption service
Perform an encryption of the counter, salted string sequence
Append the command with encrypted key
Collaborations
Transmitter System
Command Buttons Manager

Signal Receiver System

Here, we will be implementing the receiver side of our project. First, the “Receiver System” should constantly be trying to read RF signals. Then, the “Decryption Service” will take the RF signal that gets picked up, then decrypt the signal into a key and a command. The key needs to match with our rolling code value to become a match, therefore our signal would be validated. The command will then be forwarded to the “Command Control Center,” to then send further instructions for that controller that will manage the hardware which includes: the LED, buzzer, and servo motor.



Receiver System

Responsibilities	Collaborations
• Receives encrypted signal from transmitter • Communicate with decryption service about the encrypted data	• Decryption Service • Transmitter System

Decryption Service

Responsibilities	Collaborations
• Decrypt encrypted signal to get key • Compare key with stored rolling key • Manage and store a version of the hashed key and salt • Parse the command from the signal • Communicate with command control center if the key is correct	• Receiver System • Control Center Manager

Command Control Center

Responsibilities	Collaborations
• Accept commands from the decryption service handlers • Differentiate the command given • Communicate command to correct handlers of hardware (like the LED lock state manager, servo manager, etc.)	• Decryption Service • Locking State Controller • Alarm Controller • Mechanical Door Controller

Locking State Controller

Responsibilities	Collaborations
• Keep a bitwise state of the mechanism, like locked or unlocked • Output a LED to show user if the state is locked or unlocked • The locked state should output a HIGH LED • The unlocked state should output a LOW LED	• Command Control Center

Alarm Controller

Responsibilities	Collaborations
• Listen for a command from the Command Control Center • Control the buzzer hardware • Make a beeping noise 3 times when the command is given (maybe implemented) • When forced entry is detected then create a loud alarm with the buzzer	• Command Control Center • Somewhat with the Mechanical Door Controller

Mechanical Door Controller

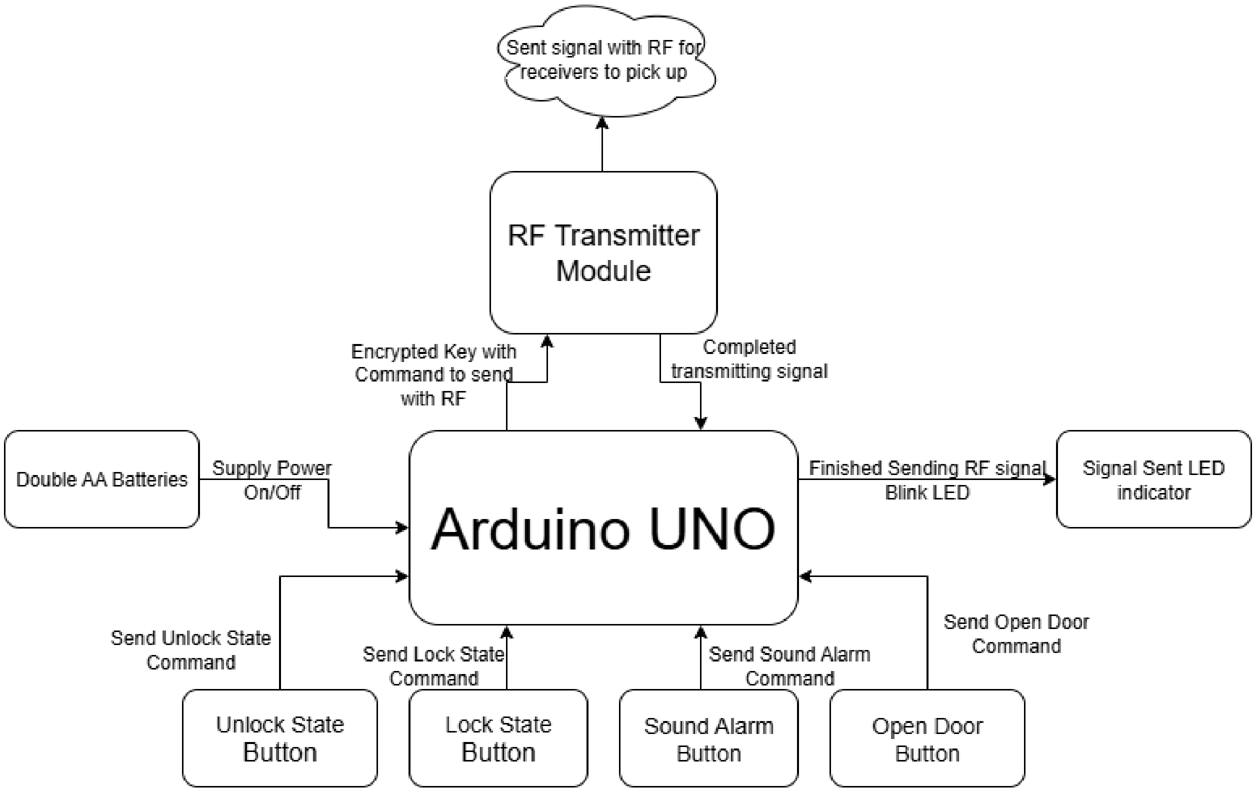
Responsibilities	Collaborations
• Listen for a command from the Command Control Center • Control the Servo Motor hardware • Activate the Servo to turn 90 degrees when open signal is received • Deactivate Servo to make the servo go back to the 0 degree position when close signal is received (maybe implemented) • Detect if the door was forcefully opened by manually • Communicate with the command center	• Command Control Center • Somewhat with the Alarm Controller

Encrypted Hashed Key with Command
Communicate command signal for control center
Send Lock/Unlock Signal to Controller
Send Beeping Signal to Controller
Forced Entry was detected (maybe implemented)
Notifies any forced entry (maybe implemented)
Send Open/Close Signal to Controller

Architecture Block Diagram

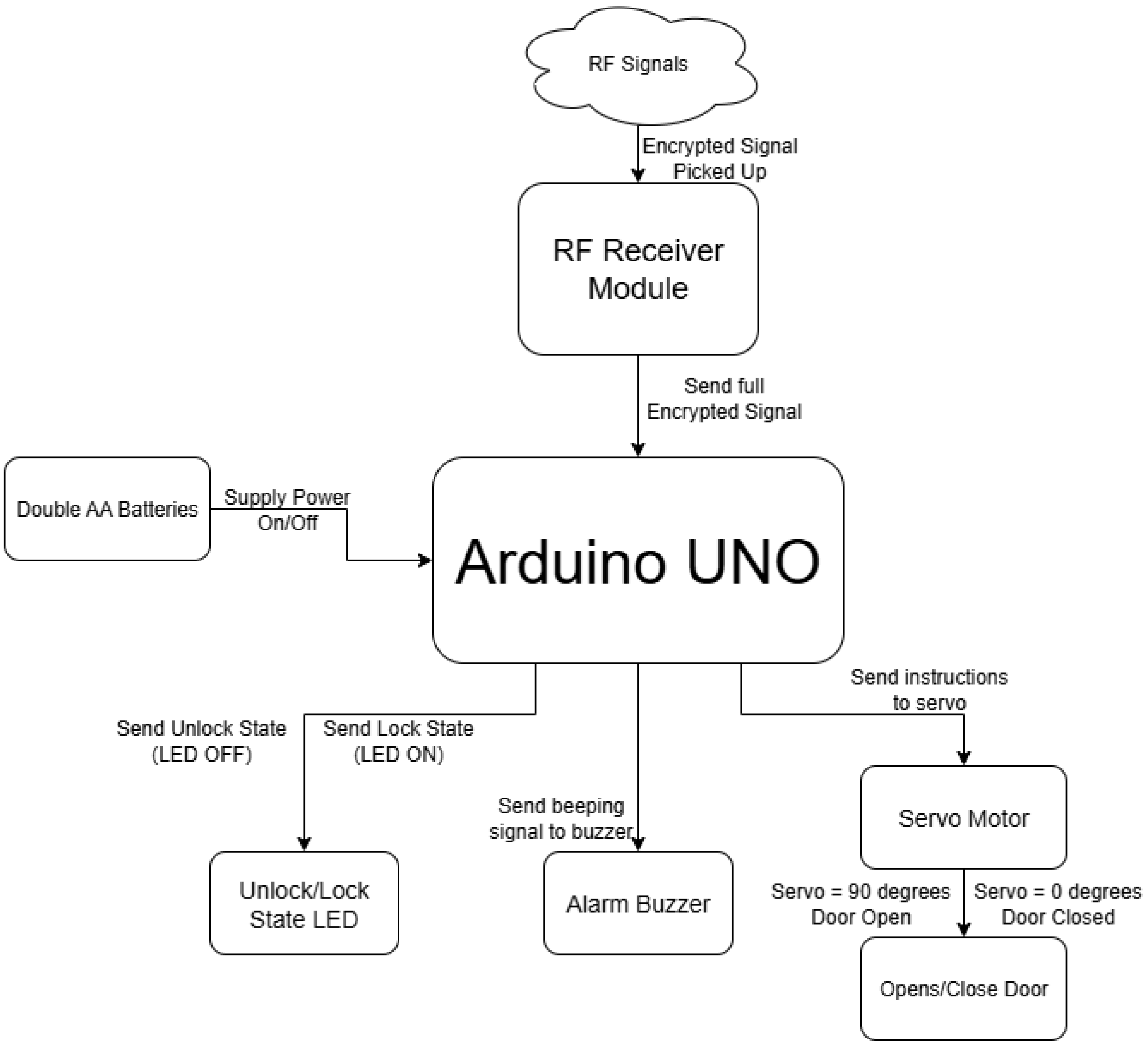
Transmitter Architecture

The Arduino will be our brains to the project, handling things such as encryption, checking our button presses, and sending signals and commands to the right hardware.



Receiver Architecture

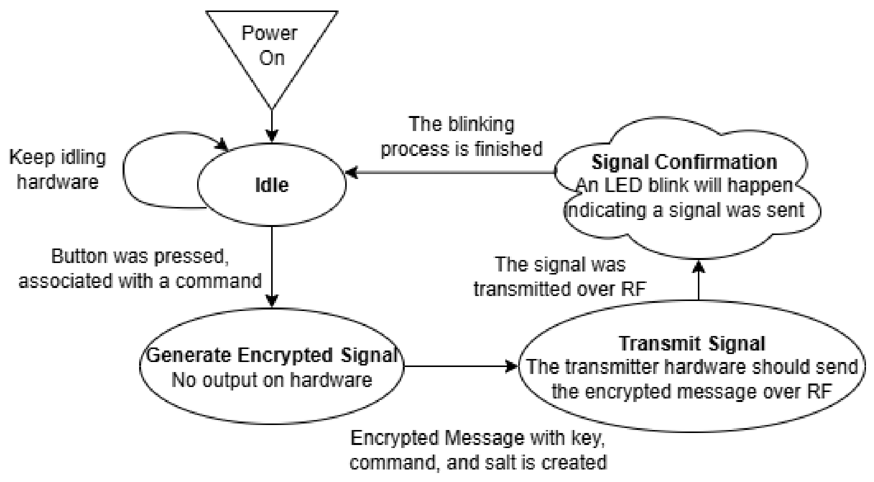
Similarly, the Arduino is our brains of the project, handling things like decryption, more state management, checking which command was given and if the keys match with expected keys. With the command, the Arduino should also send instructions to each of the individual components like LED, buzzer, and servo motor to perform the correct steps.

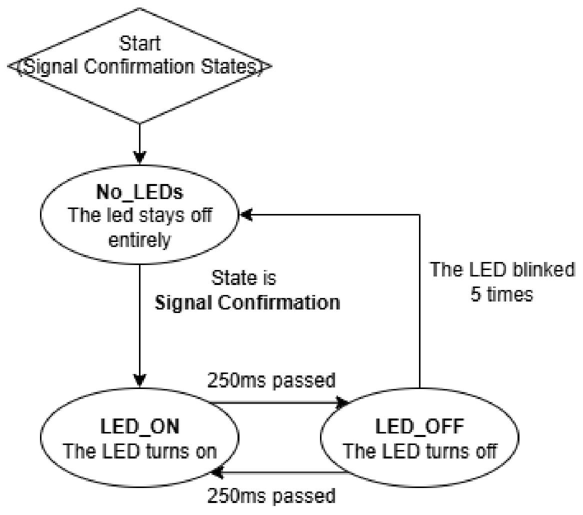

RF Signals
Encrypted Signal Picked Up
RF Receiver Module
Send full Encrypted Signal
Double AA Batteries
Supply Power On/Off
Arduino UNO
Send instructions to servo
Send Unlock State (LED OFF)
Send Lock State (LED ON)
Send beeping signal to buzzer
Servo Motor
Unlock/Lock State LED
Alarm Buzzer
Servo = 90 degrees Door Open
Servo = 0 degrees Door Closed
Opens/Close Door

Architecture Flowchart Diagram

Transmitter Flowchart

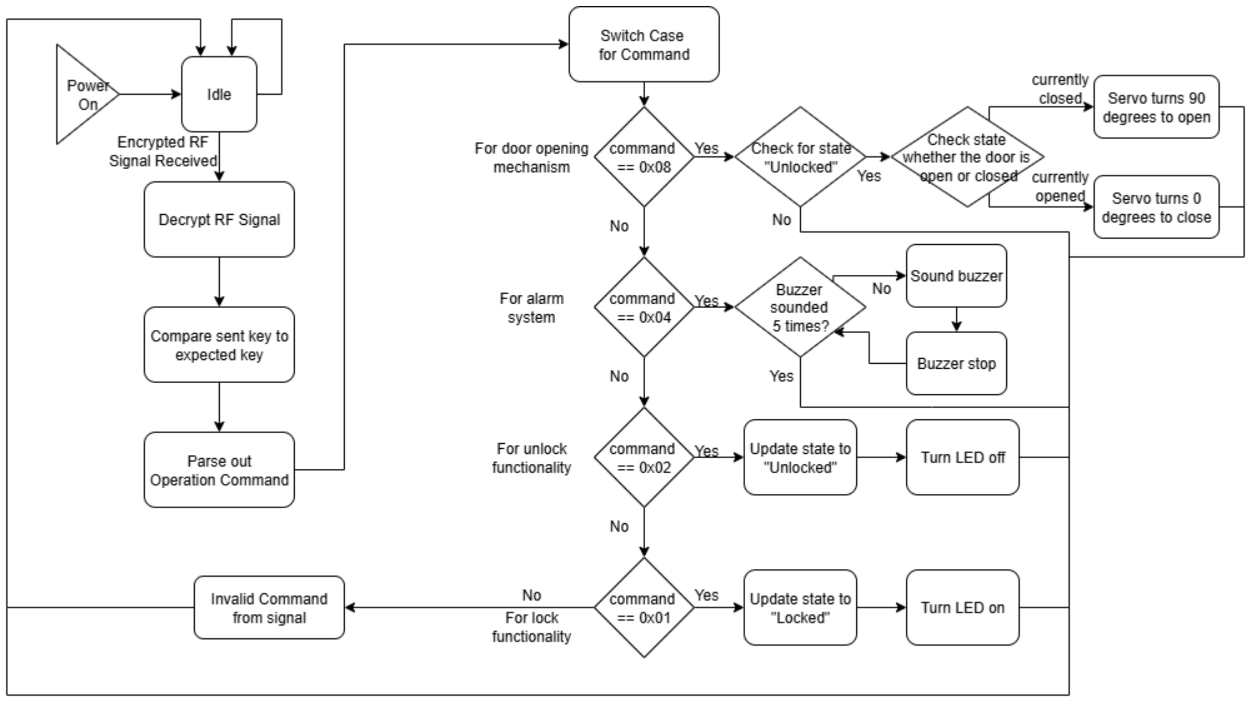
Our design for our transmitter is relatively simple. When the transmitter Arduino is turned on, the hardware will idle until a button is pressed (like a real car key remote). After a button is pressed, the Arduino will append the operation/command to the encrypted key. Then the RF module will transmit the encrypted message over RF for any receivers to accept. After the message was sent, there should be a confirmation that the encrypted signal is sent, which will most likely be a LED blink.


Power On
Keep idling hardware
Idle
The blinking process is finished
Signal Confirmation
An LED blink will happen indicating a signal was sent
Button was pressed, associated with a command
The signal was transmitted over RF
Generate Encrypted Signal
No output on hardware
Transmit Signal
The transmitter hardware should send the encrypted message over RF
Encrypted Message with key, command, and salt is created



Receiver Flowchart

Our design for our receiver is more complex than the transmitter. When the receiver Arduino is turned on, the hardware will idle until an RF signal is received. The RF receiver will take in an RF signal and send it to the Arduino. The Arduino should take the RF signal and decrypt the message, in which the message will be broken down into key and command. The key will first be compared to the expected key to determine if the signal was sent from the user (and not from a malicious attacker). When the key is accepted, the command will be passed through a switch statement. Lastly, the operation from the command will be performed on hardware, either the servo, buzzer or LED.



Component List

Item	Quantity	Unit Price	Found	Have
Arduino R3	2	$27.60	Arduino Website	Owned
Arduino Nano 33 BLE Sense	1	$39.70	Amazon	Preferred for TinyML, but No
433MHz RF Transmitter Module	2	$8.79	HiLetgo, Amazon	Borrowed from CSE450 Lab
433MHz RF Receiver Module	2	$8.79	HiLetgo, Amazon	Borrowed from CSE450 Lab
Breadboard	2	$6.99	DigiKey	Owned
Wires	100	$7.99	WWZMDiB, Amazon	Owned
Buzzer	2	$6.88	Gikfun, Amazon	Owned
LEDs	8	$9.99	ALLECIN, Amazon	Owned
Tactile Push Button	8	$8.68	Gikfun, Amazon	Borrowed from CSE450 Lab
Resistors (varies Ω)	25	$5.39	Luminology Pro, Amazon	Owned
Liquid Crystal Displays	2	$8.39	DigiKey	Borrowed from CSE450 Lab
Servo Motor	1	$3.62	DigiKey	Owned
Double A Batteries	6	$18.64	Duracell, Amazon	Owned
Cardboard	1	$0.98	Walmart	Owned
3D Printer PLA Material	1	$20.69	Amazon	Preferred for 3D printed door but No

Final Notes

The real-time components of this project are mostly in the RF signal receiver; the receiver is critical as the system can fail if there is a bit or issue in collecting the RF signal. If the receiver misses the signal or fails to collect the message from the signal, then the transmitter will be onto the next key code while the receiver is still behind. We will probably build in error detection by syncing codes and allowing room between the two systems. Both the transmitter and receiver are embedded systems, and so the percentages are most likely to be 70% embedded with a 30% real-time component.

The main motivation of this project is to learn, understand, and implement communication between two systems that are going to be secure from attackers. This method is often used in vehicle and garage door systems and allows for remote control of another system. This project can also be used to conduct research like flaws within the encrypted system, from replay attacks to brute force a correct key generation.

Some of the features of our project might change, we wanted to add a component that detects if the key is near, then the door will automatically open. We also felt like the unlocking and locking component was repetitive and did not need two dedicated buttons, so we might implement a different feature if we have the time. The LED for locking and unlocking is also kind of simple, so we might upgrade the feature if a new idea comes up. What we have outlined above is the minimal requirements of our secure embedded project and any additional features will just be experimenting and learning new software techniques and hardware modules.