Key Immobilizer System

https://github.com/mokshayagna/immobilizer

1. Introduction

Modern vehicles rely heavily on electronic control units (ECUs) to ensure safety, reliability, and security. One of the most critical security mechanisms in an automobile is the **Engine Immobilizer System**, which prevents unauthorized engine start. This internship project focuses on understanding, designing, and implementing a **Key Immobilizer system** using a secure cryptographic challenge–response mechanism.

The project simulates a real-world automotive security scenario where a **Transponder ECU (Key)** and an **Engine ECU** authenticate each other before allowing the engine to start. The authentication is completed within strict real-time constraints and mirrors industry-grade embedded security workflows.

This project is intentionally designed to expose interns to **embedded systems, networking, cryptography, Linux-based development, and debugging tools**, making it an excellent bridge between academic concepts and real-world engineering practice.

---

2. Project Objectives

The primary objectives of this project are:

- To understand how automotive immobilizer systems work at a system and protocol level
- To implement a secure **challenge–response authentication flow** using AES-128
- To gain hands-on experience with embedded programming and Linux-based systems

- To learn automotive communication concepts such as CAN protocol
- To develop debugging, testing, and system-integration skills used in industry

By the end of this project, interns should be able to explain and demonstrate how a secure key-based authentication system prevents unauthorized vehicle access.

---

3. System Architecture Overview

The project consists of two logical components:

3.1 Transponder ECU (Key Side)

The Transponder ECU represents the vehicle key or key fob. Its responsibility is to:

- Wake up when the ignition is turned ON
- Receive an authentication request from the Engine ECU
- Receive a random challenge (nonce)
- Encrypt the challenge using a shared AES-128 secret key
- Send the encrypted response back to the Engine ECU
- Wait for authentication success or timeout

The transponder never exposes the secret key, ensuring system security.

3.2 Engine ECU (Vehicle Side)

The Engine ECU represents the vehicle's control unit responsible for engine authorization. Its responsibilities include:

- Detecting ignition ON
- Generating a cryptographically strong random challenge
- Sending the challenge to the transponder
- Receiving the encrypted response
- Encrypting the same challenge locally using its stored secret key
- Comparing results and deciding whether to enable or block the engine

4. Complete Authentication Flow

The authentication process follows a strict sequence:

1. Ignition is turned ON
2. Engine ECU generates a random challenge (nonce)
3. Challenge is sent to the Transponder ECU
4. Transponder encrypts the challenge using AES-128
5. Encrypted response is sent back to Engine ECU
6. Engine ECU encrypts the same challenge using its stored key
7. Results are compared
8. If valid, engine start is enabled; otherwise, the engine remains immobilized

All steps must complete within **2 seconds**, reflecting real automotive timing constraints.

5. Technology Stack and Tools

This project uses a carefully selected set of tools and technologies commonly found in embedded and automotive environments.

5.1 Programming and Operating System

- **C Programming Language**: Used for implementing core logic, cryptographic flow, CAN communication, and socket handling
- **Linux**: Development and execution environment, simulating ECU behavior on embedded Linux platforms

5.2 Embedded Platform

- **Raspberry Pi Single Board Computer**: Acts as a development and prototyping platform for ECU simulation

5.3 Communication and Networking

- **CAN Protocol**: Simulates in-vehicle communication between ECUs
- **CAN Analyzer / CANalyzer Tools**: Used to observe, analyze, and debug CAN frames
- **Network Sockets (TCP/UDP)**: Used to simulate ECU-to-ECU communication where physical CAN hardware is not available

5.4 Debugging and Build Tools

- **gdb (GNU Debugger)**: Used for step-by-step debugging, breakpoints, and runtime inspection
- **Makefile**: Used to automate builds, manage dependencies, and standardize compilation

5.5 Analysis and Monitoring Tools

- **Wireshark**: Used to capture and analyze network traffic, helping interns understand data flow and protocol behavior

---

6. Development Workflow

The typical development workflow for this project includes:

1. Designing the authentication flow and message formats
2. Implementing Engine ECU and Transponder ECU logic in C
3. Setting up communication using CAN or network sockets
4. Building the project using Makefiles
5. Debugging runtime behavior using gdb
6. Monitoring traffic using Wireshark or CAN analysis tools
7. Validating timing constraints and authentication correctness

This workflow closely mirrors professional embedded software development practices.

---

7. Security Considerations

The project introduces interns to practical security concepts, including:

- Symmetric cryptography using AES-128
- Use of random nonces to prevent replay attacks
- Avoiding transmission of secret keys over communication channels
- Timeout handling and failure scenarios

While advanced topics such as HSMs and side-channel protection are out of scope, the project establishes a strong foundation in secure system design.

8. Learning Outcomes

Upon successful completion of this project, interns will:

- Understand real-world automotive immobilizer systems
- Gain confidence in C programming for embedded applications
- Learn Linux-based development and debugging
- Understand CAN protocol fundamentals
- Use professional tools such as gdb, Wireshark, and CAN analyzers
- Develop system-level thinking across hardware, software, and security domains

9. Conclusion

This Key Immobilizer project is designed as an industry-aligned internship assignment that balances **theoretical understanding and hands-on implementation**. It exposes interns to critical concepts in embedded systems, automotive security, and real-time communication while remaining approachable and educational.

The experience gained through this project prepares interns for advanced roles in embedded software, automotive systems, and security-focused engineering domains.