DIGITAL SYSTEM DESIGN FINAL PROJECT REPORT

DEPARTMENT OF ELECTRICAL ENGINEERING

UNIVERSITAS INDONESIA

ROLLING CAR KEY: State Machine Logic for Secure Car Access using Synchronized Rolling Codes

GROUP 1:

Nadira Fayyaza Aisy	**2406368933**
Nayla Pramesti Adhina	**2406368901**
Hanif Mulia Nugroho	**2406369066**

PREFACE

We are thankful to the Almighty God who has granted us inexhaustible grace and blessings that have made us accomplish this final project report. The report is being prepared in the context of completing the final project of the Digital System Design course in the academic year 2025/2026, which is called ROLLING CAR KEY: State Machine Logic of Secure Car Access with the rolling codes synchronized. The approach adopted in this project is VHDL (VHSIC Hardware Description Language), through which a hardware-based digital system can be designed and implemented to provide us with secure access control of vehicles.

The project will develop and optimize a keyless entry system that combines a remote key device with a car-side authentication unit, using the SPI (serial peripheral interface) protocol to facilitate reliable data transmission. The system uses an OTP (One-Time Password) mechanism and counter synchronization to provide strong security and prevent unauthorized access.

We would also like to acknowledge that the lab assistants, Mr. M. Nadzhif Fikri, helped with the implementation and testing process and were very patient. The project is an outcome of strong teamwork, and every step of the work has provided more insights into how theoretical concepts can be applied to the field of computer engineering, namely, in the area of digital systems and safe data transport. We won't deny that this report might still be limited and deficient. Hence, positive comments and recommendations are most welcome to be reviewed and further enhanced

Depok, December 07, 2025

Group 1

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The issue of vehicle security is highly acute in the modern world because the key-based system is becoming more and more susceptible to theft and unauthorized access. Current car models have electronic access controls based on wireless communication between the remote key and the control unit installed on the car. Nevertheless, most of the current systems implement fixed or foreseeable authentication systems, which could be hacked, repeated, or duplicated, which is rather dangerous.

In order to deal with these weaknesses, authentication systems like One Time Passwords (OTP) and dynamic counter-based systems have been implemented in other secure communications. These techniques make every authentication unique and time-sensitive. The introduction of these security functions in an embedded system is to be done in a co-design of both hardware and software, an effective communication protocol, and a strong synchronization of the transmitter to the receiver.

This project aims at the design and implementation of a secure vehicle entry system with the use of VHDL to model the hardware behavior of the key device (transmitter) and the car unit (receiver). It uses the SPI protocol to transmit the data reliably with low latency and has an OTP generator based on a secret key and a running counter. The design focuses on authenticating on the fly, handling keys in a secure manner, and having a smooth interaction between the key and the vehicle control system.

With the creation of this system as a hardware-based project, we will be investigating the practical issues of the design of the secure embedded system, such as the state machine control, data integrity, protocol management, synchronization, and the limitations of the FPGA or ASIC implementation. This project doesn't just implement any theoretical learning of digital design and communication protocols; it also proves the need to observe the issue of security in contemporary electronic systems.

1.2 PROJECT DESCRIPTION

The design of this project is a safe remote keyless entry system in VHDL, comprising a key transmitter and a car receiver that are in communication through SPI. The system practices a rolling code system with the help of a one-time password (OTP) to avoid replay attacks. The pressing of each button will add a counter and, together with a shared secret key, produce a unique OTP hash.

Three packets are sent sequentially with the help of a finite state machine, namely: User ID, Counter, and OTP, which are exchanged by the key. These packets are sent to the Car unit, which accesses the secret key through an internal database with the User ID and then recalculates the expected OTP. Authentication only works when the OTP received matches the locally generated OTP, and the counter received falls within a valid synchronization window before the stored counter.

Key Features that were included in this project are as follows:

- Rolling Code Security: This type of security uses a non-linear algorithm (XOR, rotate bits, constants) to make each transmission have a unique OTP.
- Communication: SPI Master (Key) and SPI Slave (Car). Custom implementation of SPI Master and SPI Slave to ensure reliable data transfer.
- Synchronization Window: The receiver tolerates counters in a certain range to deal with desynchronization to deter replay attacks.
- Database Management: The Car has a database of authorized UserID, Secret Key, and last valid Counter values.
- Finite State Machine (FSM): Coordinates the complex control flow of a sequence of packet transmission and validation code.

A successful authentication opens the door and fills the counter-database of the car; an unsuccessful attempt raises an alarm. The system is completely coded in synthesizable VHDL, which shows a realistic hardware-based approach to automotive security, synchronization, and embedded communication.

1.3 OBJECTIVES

The objectives of this project are as follows:

1. To establish a rolling code authentication scheme based on the One-Time Password (OTP) that is calculated by using a counter and a secret key, which is effective in preventing replay attacks.
2. To establish reliable master-slave communication between the key and car unit using the Serial Peripheral Interface (SPI) protocol.
3. To develop and integrate a synchronized counter system that increments with each transmission and validates counter freshness within a defined window on the receiver side.
4. To create modular VHDL components, including finite state machines (FSMs), an OTP generator, SPI controllers, and a secure database for key and counter storage.
5. To ensure the complete system is synthesizable and capable of real-time authentication, with clear outputs for door lock/unlock status and alarm activation.

1.4 ROLES AND RESPONSIBILITIES

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
Car_Database.vhd, Key_Database.vhd, Final Report document	● Database creation and integration for both Car and Key. ● Documentation and report creation. ● PowerPoint presentation creation.	Nayla
Car.vhd, Car_FSM.vhd, SPI_Slave, SPI_Master, System_tb	● Initiator of the Final Project Idea ● Creation of the main program,	Hanif

	especially the Car System, which includes top-level and FSM, and Serial Communication. ● System testbench that combines both the key and the car system to be tested ● Discussion and design of the application concept	
Key.vhd, Key_FSM.vhd, OTP_Generator, Button_Debouncer, Counter_Inc, README	● Key Systems that include top-level, FSM, button debouncers, counter increment, and OTP Generator creation and integration. ● Documentation and report creation. ● PowerPoint presentation creation.	Nadira

Table 1. Roles and Responsibilities

CHAPTER 2

IMPLEMENTATION

2.1 EQUIPMENT

The tools that are going to be used in this project are as follows:

- Visual Studio Code
- ModelSim & Quartus Prime
- Github

2.2 IMPLEMENTATION

In working on this Digital System Design Final Project, we applied basic concepts such as:

1. **Dataflow Style Programming in VHDL**

   Dataflow Style is implemented through concurrent signal assignments that directly wire internal signals to outputs, such as SCK<=sck_int in the SPI master and Tx_Active<=spi_is_busy in the key unit. It also appears in combinational processes like the packet selector MUX, where outputs update immediately when inputs change, emphasizing parallel signal propagation over sequential execution.

2. **Behavioral Style Programming in VHDL**

   In this project, behavioral styles were mainly implemented through process statements with sensitivity lists that describe sequentially and are driven by clock behavior. It's used extensively in all finite state machines, such as key_fsm, car_fsm for state transitions, in modules like counter_inc and spi_master for sequential counting and shifting, and in the otp generator for algorithmic computation. This enables a procedural description of functionality using if-then-else, case, and synchronous updates on clock edges, and focuses on what the system does rather than its structural wiring.

3. **Testbench**

Testbench, in this project, was handled in a single VHD. file. Our testbench uses structural instantiation of both key and car units, creates a clock process with controlled timing, and implements stimulus procedures such as press_unlock_button, lock_car to simulate user interactions. It monitors outputs such as car_door, car_siren, and key_tx_active to track authentication success through unlock_count. Although in our case we don't use asserts and reports, this testbench still validates our SPI communication system and state machine synchronization across multiple unlock attempts.

4. **Structural Programming**

Although it was done hybridly (behavioral and structural combinations), with behavioral implemented in packet selection, signal assignment & control logic, and constant configurations, structural programming was also comprehensively implemented through component instantiation for flexibility and ease in modifying or debugging to check its functionality. The system is also built by connecting pre-designed modules (component declarations) using a port map. In key.vhd six components are instantiated: button_debouncer, counter_inc, otp_generator, key_fsm, SPI_master, and internal signals are wired to form the complete system itself. Similarly done in a car.vhd, instantiations made: SPI_slave, Car_Database, OTP_Generator, and Car_FSM.

The testbench also uses structural programming by instantiating both key and car top-level entities, connecting their SPI ports directly to simulate full system communication. This approach allows for independent development, testing, and reuse of each functional block. Arrays are also implemented in this project, primarily in the Car_Database and Key_Database, where they are used to create an efficient table structure for storing and retrieving authentication credentials.

5. **Looping Construct**

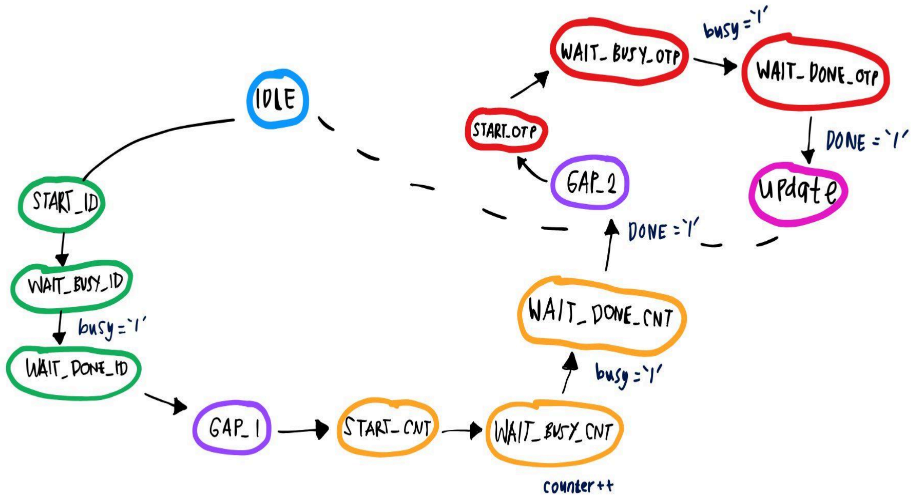
This module is implemented in the Car_Database using a for loop to perform a sequential search operation through the database array. It iterates through all 4 database entries (0 to 3 for loops) to execute two functions, the first one being to search for a matching user ID during the authentication lookup, and the second is to update the corresponding counter value after successful verification. The choice made for a for-loop over something like while or for-generate is due to a fixed number of iterations aligning with the known size of the database, ensuring it was gonna be able to be synthesized and readable.

6. Finite State Machine

Finite State Machines (FSMs) are actually central to our project's control logic, implemented in both Key_FSM and Car_FSM using a behavioral style with explicit state and next-state logic.

1.) Key_FSM (transmitter)

This FSM has a purpose to control the sequential transmission of three authentication packets (user ID to counter to OTP) via SPI. There are 13 states in total:



The FSM starts in IDLE and transitions to START_ID when the Button_Press is received. The FSM subsequently transmits the User_ID, the Counter, followed by the OTP (Rolling Code), and states such as WAIT_DONE_ID, WAIT_DONE_CNT, and

WAIT_DONE_OTP are used to synchronize the SPI after a packet.

Importantly, the WAIT_BUSY_CNT state provides the obligatory operation of the increased value of the counter state, Counter++, prepping the system for the next transaction. Last, the UPDATE state indicates success of the transaction with the car, and it'll permanently record the counter in Key_DB if successful, & goes back to IDLE.

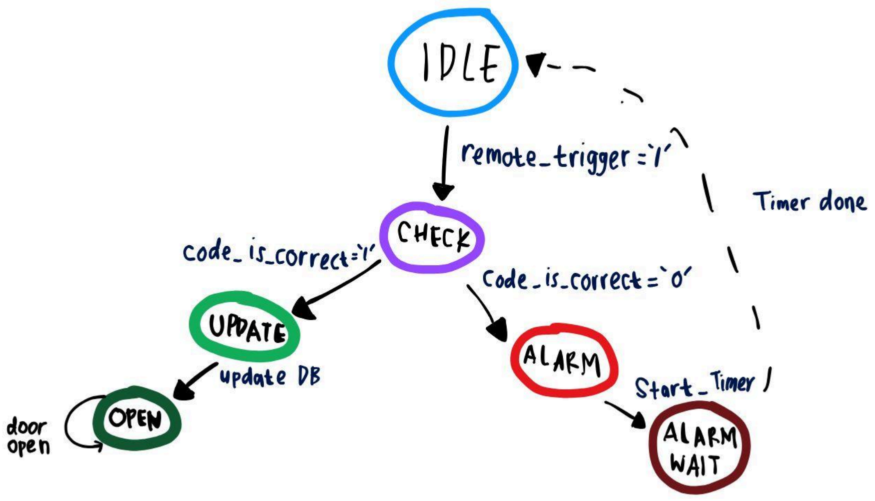
The following table is a more detailed state functions:

State	Trigger	Next State	Outputs	Type
ST_IDLE	Button_Press='1'	ST_START_ID	SPI_Start='0' Packet_Select="00" Inc_Counter='0'	Moore
ST_START_ID	Always triggering	ST_WAIT_BUSY_ID	SPI_Start='1', Packet_Select="00"	Moore
ST_WAIT_BUSY_ID	SPI_Busy='1'	ST_WAIT_DONE_ID	Packet_Select="00"	Mealy
ST_WAIT_DONE_ID	SPI_Busy='0'	ST_GAP_1	Packet_Select="00"	Mealy
ST_GAP_1	Always triggering	ST_START_CNT	Packet_Select="01"	Moore
ST_START_CNT	Always triggering	ST_WAIT_BUSY_CNT	SPI_Start='1', Packet_Select="01"	Moore
ST_WAIT_BUSY_CNT	SPI_Busy='1'	ST_WAIT_DONE_CNT	Packet_Select="01"	Mealy
ST_WAIT_DONE_CNT	SPI_Busy='0'	ST_GAP_2	Packet_Select="01"	Mealy

ST_GAP_2	Always triggering	ST_START_OTP	Packet_Select="10"	Moore
ST_START_OTP	Always triggering	ST_WAIT_BUSY_OTP	SPI_Start='1', Packet_Select="10"	Moore
ST_WAIT_BUSY_OTP	SPI_Busy='1'	ST_WAIT_DONE_OTP	Packet_Select="10"	Mealy
ST_WAIT_DONE_OTP	SPI_Busy='0'	ST_UPDATE	Packet_Select="10"	Mealy
ST_UPDATE	Always triggering	ST_IDLE	Inc_Counter='1' (1-cycle pulse)	Moore

2.) Car_FSM (Receiver Security)

This state machine has a purpose to manage the car's response to authentication attempts (whether it's unlock or alarm). It has 5 states as visualized below:



IDLE transitions to CHECK upon receiving the initial trigger. In CHECK, the FSM quickly performs the critical validation: receiving all three packets (ID, Counter, OTP) via SPI, looking up the User ID, calculating the Expected OTP, and comparing the two OTP values.

Success Path (Code_Is_Correct = '1'): The FSM passes to UPDATE (permanently incrementing the Counter C in the database), and then to OPEN (setting Door_Open = '1'), and again to IDLE.

Failure Path (Code_Is_Correct = '0): When ALARM (activating Alarm_Siren = '1') is entered, ALARM_WAIT (forcing a lockout timer to prevent brute-force attacks) is enforced, then at last the FSM is returned to IDLE.

The following table is a more detailed state function:

State	**Trigger**	**Next State**	**Outputs**	**Type**
ST_IDLE	Remote_Trigger='1'	ST_CHECK	Door_Open='0', Alarm_Siren='0', Check_Enable='0', Update_DB='0'	Moore
ST_CHECK	Code_Is_Correct='1'	ST_UPDATE Code_Is_Correct='0'	ST_ALARM	Check_Enable ='1' Mealy
ST_UPDATE	Always	ST_OPEN	Update_DB='1'	Moore
ST_OPEN	Always	ST_OPEN (hold)	Door_Open='1'	Moore
ST_ALAR	Always	ST_ALARM	Alarm_Siren='1'	Mooreq

M		(hold)		

CHAPTER 3

TESTING AND ANALYSIS

3.1 TESTING

Among other things, to debug the output of the Secure Vehicle Entry System as well as ensure that all the security features of the system are safe, extensive testing based on a planned VHDL testbench was done. To replicate the coordinated communication and cryptographic activities between the Key device (SPI Master) and the Car unit (SPI Slave), a testbench was required. The simulation involved 5 discrete cycles to be executed one after another in order to test the integrity and recovery of the system, and it was found that the system can handle normal as well as security challenges. The testing platform was strictly concerned with the robustness of the rolling-code scheme and its sensitivity to various cases of attacks, to warrant the Counter to be stateful and the Database update to be integrally sound.

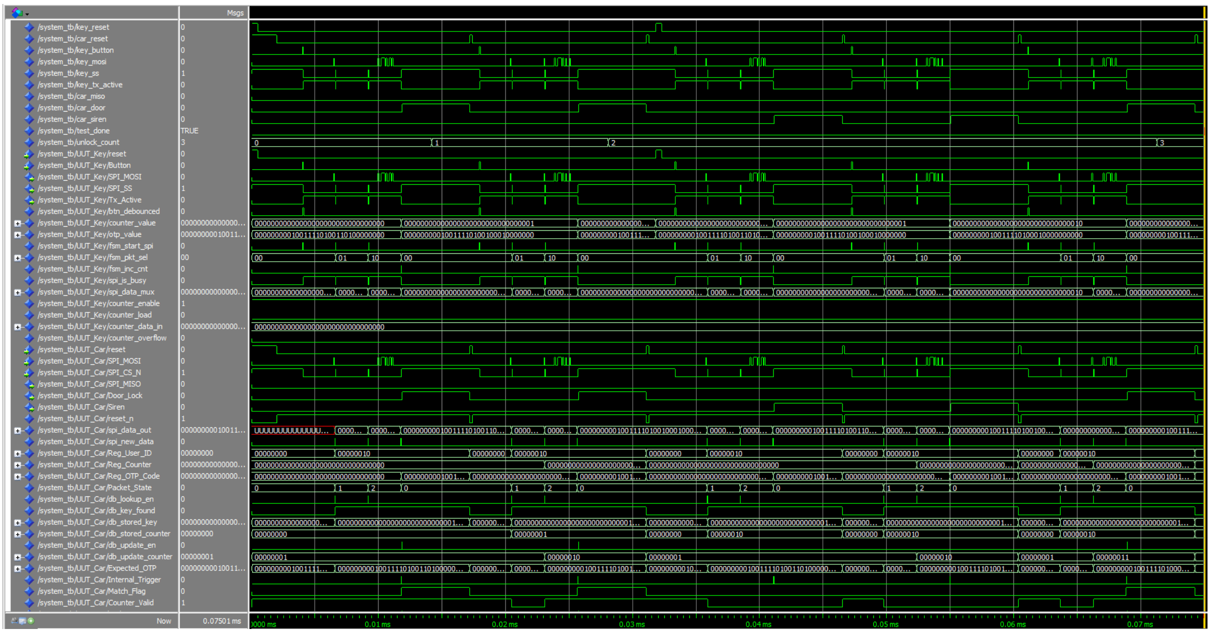
The simulation featured 5 discrete cycles, which were performed one at a time in order to test both system correctness and recovery, which proved that the system would be capable of managing both normal and security challenges:

- Cycle 1 and 2 (Success): The authentication is performed successfully in a sequence Counter Freshness and Database Integrity Door\Lock = '1'.
- Cycle 3 (Failure): Replay Attack Simulation, Alarm\Siren = '1' Rejection of Stale Counter.
- Cycle 4 (Failure): Synchronization Failure (First Recovery Attempt), Alarm\Siren = '1' (Second failure), Security Tolerance Window.
- Cycle 5 (Success): Full Recovery, Door\Lock = 1 and Counter Update Usability after Alarm.

The testbench used particular VHDL functions to simulate button presses and key resets, thus providing proper control of the Event Counter value of each scenario.

3.2 RESULT

The following are the simulation results from a testbench created using ModelSIM.



```
                                          Msgs
/system_tb/key_reset                      0
/system_tb/car_reset                      0
/system_tb/key_button                     0
/system_tb/key_mosi                       0
/system_tb/key_ss                         1
/system_tb/key_tx_active                  0
/system_tb/car_miso                       0
/system_tb/car_door                       0
/system_tb/car_siren                      0
/system_tb/test_done                      TRUE
/system_tb/unlock_count                   3
/system_tb/UUT_Key/reset                  0
/system_tb/UUT_Key/Button                 0
/system_tb/UUT_Key/SPI_MOSI               0
/system_tb/UUT_Key/SPI_SS                 1
/system_tb/UUT_Key/Tx_Active              0
/system_tb/UUT_Key/btn_debounced          0
/system_tb/UUT_Key/counter_value          00000000000000...
/system_tb/UUT_Key/otp_value              0000000010011...
/system_tb/UUT_Key/fsm_start_spi          0
/system_tb/UUT_Key/fsm_pkt_sel            00
/system_tb/UUT_Key/fsm_inc_cnt            0
/system_tb/UUT_Key/spi_is_busy            0
/system_tb/UUT_Key/spi_data_mux           00000000000000...
/system_tb/UUT_Key/counter_enable         1
/system_tb/UUT_Key/counter_load           0
/system_tb/UUT_Key/counter_data_in        00000000000000...
/system_tb/UUT_Key/counter_overflow       0
/system_tb/UUT_Car/reset                  0
/system_tb/UUT_Car/SPI_MOSI               0
/system_tb/UUT_Car/SPI_CS_N               1
/system_tb/UUT_Car/SPI_MISO               0
/system_tb/UUT_Car/Door_Lock              0
/system_tb/UUT_Car/Siren                  0
/system_tb/UUT_Car/reset_n                1
/system_tb/UUT_Car/spi_data_out           0000000010011...
/system_tb/UUT_Car/spi_new_data           0
/system_tb/UUT_Car/Reg_User_ID            00000000
/system_tb/UUT_Car/Reg_Counter            00000000000000...
/system_tb/UUT_Car/Reg_OTP_Code           00000000000000...
/system_tb/UUT_Car/Packet_State           0
/system_tb/UUT_Car/db_lookup_en           0
/system_tb/UUT_Car/db_key_found           0
/system_tb/UUT_Car/db_stored_key          00000000000000...
/system_tb/UUT_Car/db_stored_counter      00000000
/system_tb/UUT_Car/db_update_en           0
/system_tb/UUT_Car/db_update_counter      00000001
/system_tb/UUT_Car/Expected_OTP           0000000010011...
/system_tb/UUT_Car/Internal_Trigger       0
/system_tb/UUT_Car/Match_Flag             0
/system_tb/UUT_Car/Counter_Valid          1
                                    Now   0.07501 ms
```

Fig 2. Testing Result

3.3 ANALYSIS

The safe vehicle entry system is efficient in meeting all the project objectives due to the analysis that depicts the high degree of security validation and recovery systems. The security test of the system is completely deterministic and aligned with the security objectives of the project.

Sequential Success Validation

1. Key Transmission Sequence (Master Side)

   The waveform confirms the Key_FSM follows the correct three-packet sequence required by the protocol:

   - Trigger and Counter Increment: The system is initiated by a key event (implied by debounced button signals), causing the System.DUT.Key_Counter to increment (e.g., 0→ 1→2 in the later part of the trace). This fresh counter is the first input to the OTP generator.

- OTP Calculation: The OTP_Generator calculates the unique OTP_Result using the new counter and the Secret_Key *before* transmission begins, fulfilling the rolling code requirement.
- SPI Transmission: The system executes three distinct transmission phases (indicated by System.DUT.SPI_busy activity) to send:
    - Packet 1: User ID (for database lookup).
    - Packet 2: Current Counter Value (for synchronization check).
    - Packet 3: Calculated OTP (for final authentication)

2. Car Authentication Sequence (Slave Side)

First, the system checks the database. When System.DUT.db_key_found goes high; it means the user ID matches, and the system pulls up the right credentials. Next, it looks at the counter. If System.Counter_Valid goes high, the system knows the counter from Packet 2 is fresh and sits inside the allowed sync window,w or basically, it's not too old or too far ahead.

Now comes the main part of the security check, when the System.Match_Flag goes high (this all happens super fast, around 0.04 ms), and the received OTP matches what the system expected. That's the green light. With the match, the state machine kicks into gear. System.System_State moves through its steps, first ST_CHECK, then ST_UPDATE, and finally ST_OPEN. At the end, the door actually unlocks: System.Door_Open goes high, and System.Door_Lock drops low. And after a successful login, the system updates the database, saving the new counter value. This way, nobody can replay the same code and sneak in again.

Security Failure Validation

1. Prevention of Replay Attack

   The waveform shows how the system defends itself against big security threats, mainly replay attacks and synchronization problems.

   The most crucial defense is the Synchronization Window and Database Update:

   - Counter Freshness: The System.Counter_Valid check ensures that a re-sent (replayed) old counter value would fail the test, specifically the condition $Recv \leq Stored$. If an attacker captures and re-sends an old packet, the System.Counter_Valid signal will remain low, preventing the System.Match_Flag from ever going high, thereby forcing the Car_FSM into the failure sequence (ST_ALARM).
   - Post-Success Update: By updating the stored counter in the Car_Database immediately after a successful unlock (ST_UPDATE state), the system invalidates the currently used OTP and counter for any future attempt, making the captured transaction useless for an attacker.
2. Synchronization Handling

   The system is designed to tolerate benign desynchronization using the upper limit of the window (example: $Recv \leq Stored + 10$):

   - Tolerance: The $Recv \leq\ UpperLimit$ check allows a few missed button presses (and thus missed counter increments) to still result in a System.Counter_Valid high signal. This ensures that the legitimate user is not locked out due to minor transmission failures. A successful authentication visible in the waveform implies that any desynchronization that occurred remained within the +10 tolerance.

   The analyzed waveform confirms that the RKE system is both functionally correct in its operational sequence and secure against replay attacks through its counter and validation logic.

CHAPTER 4

CONCLUSION

Secure Vehicle Entry System Implementation and design with the help of VHDL has been successfully done, and in all aspects, the objectives of the project were achieved and the main goals were to develop a strong hardware-based authentication system. This project will tackle the vital vulnerability of replay attacks in contemporary automotive security as they will be replacing the fixed-code systems with a dynamic rolling code architecture.

Achievement of Security Objectives The main success of this project will be the implementation of the One-Time Password (OTP) scheme supported by a synchronized counter system. As it is shown in the analysis of the simulation, the system successfully authenticates the legitimate users and prevents unsuccessful attempts of access by unauthorized ones. Logic that is implemented in the CarFSM and CarDatabase verify that any packet that has been captured or replayed is automatically invalidated by the freshness of a check on the counter. Moreover, the addition of a synchronization window comes in as a key functionality, where the system can afford small desynchronizations (to a specified threshold) without freezing the user, to allow compromising maximum security with reasonable ease of use.

Stability of Communication and Control The addition of the Serial Peripheral Interface (SPI) offered a dependable and effective communication platform between the Key (Master) and the Car (Slave). The separation of the transmission into three different packets, namely: User ID, Counter, and OTP, that the Key_FSM managed, guaranteed the integrity of the data in the transmission channel. The Finite State Machines were also instrumental in the coordination of the complex timing requirements, and a smooth transition between idle, verification, and actuation states (Door Open/Alarm,) depending on the real-time inputs.

System Synthesis and Verification. The system was thoroughly checked by applying the ModelSim testbench, which tested the system functionality under different conditions. The results of the waveforms showed that:

- Successful Authentication: This Authentication is achieved only when the created OTP matches the expected hash, and the counter is within the valid window.

- Database Management: This system will properly update the internal database upon successful entry to ensure that the same code is not reused in the future.
- Threat Mitigation: The mitigation, which is to evoke the alarm state when stale counters are detected or invalid OTPs are detected, is effective as shown by the security logic.

To sum up, in this project, it has been shown that VHDL is an effective tool when it comes to creating complex and secure embedded systems. It emphasizes the fact that the optimal architecture for modern vehicle security is the co-design of cryptographic algorithms (OTP), state-machine logic, and trusted communication protocols. The resultant system is a modular, scalable, and synthesizable solution that can combat the dangers of wireless keyless entry quite successfully.

REFERENCES

[1] “Module 9 - Microprogra... | Digilab UI,” Digilabdte.com, 2025. https://learn.digilabdte.com/books/digital-sistem-design-psddsg/chapter/module-9-microprogramming

[2] GeeksforGeeks, “What is Serial Peripheral Interface (SPI)?,” GeeksforGeeks, Mar. 13, 2024. https://www.geeksforgeeks.org/electronics-engineering/what-is-serial-peripheral-interface-spi/

[3] “Bypassing Rolling Code Systems – AndrewNohawk.” https://www.andrewmohawk.com/2016/02/05/bypassing-rolling-code-systems/

[4] mohamedamine99, “GitHub - mohamedamine99/Digital-Lock-with-VHDL-state-machine,” GitHub, 2025. https://github.com/mohamedamine99/Digital-Lock-with-VHDL-state-machine

[5] harryli0088, “GitHub - harryli0088/rolling-code,” GitHub, 2025. https://github.com/harryli0088/rolling-code

[6] robert-mcdermott, “GitHub - robert-mcdermott/rolling-code-auth: A basic implementation of a rolling code authentication system.,” GitHub, 2025. https://github.com/robert-mcdermott/rolling-code-auth

[7] kyori19, “GitHub - kyori19/verilog-otp: VerilogHDL implementation of One-Time Password Algorithm (HOTP),” GitHub, 2025. https://github.com/kyori19/verilog-otp

[8] A. Hari, K. Sai, Madhavi Macharapu, and E. S. Babu, “Secure Rolling Code Generation for Remote Keyless Entry Systems Using AES-CTR, Encryption with ChaCha20,” pp. 1–6, Dec. 2023, doi: https://doi.org/10.1109/smartgencon60755.2023.10442146.

APPENDICES

Appendix A: Project Schematic

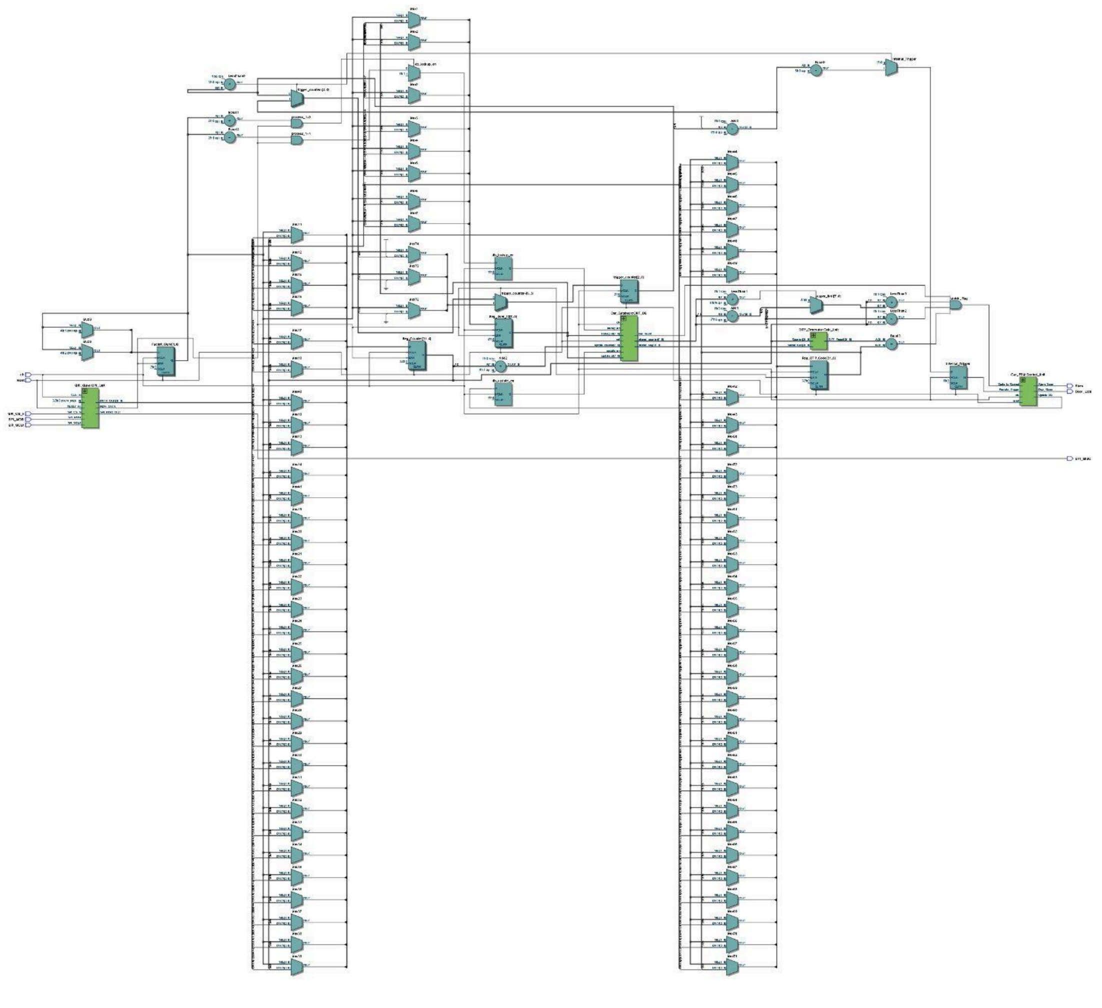


Fig. Car system

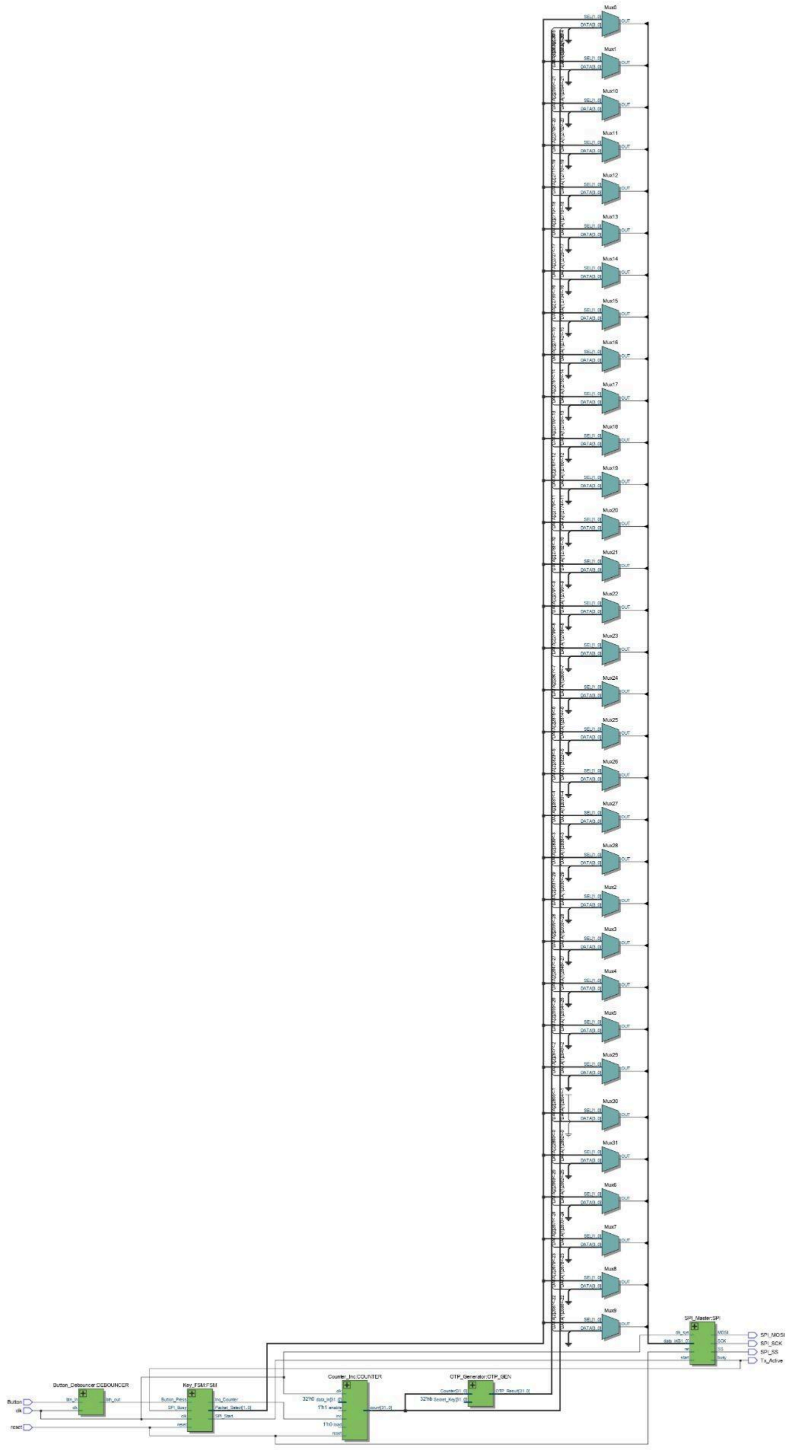
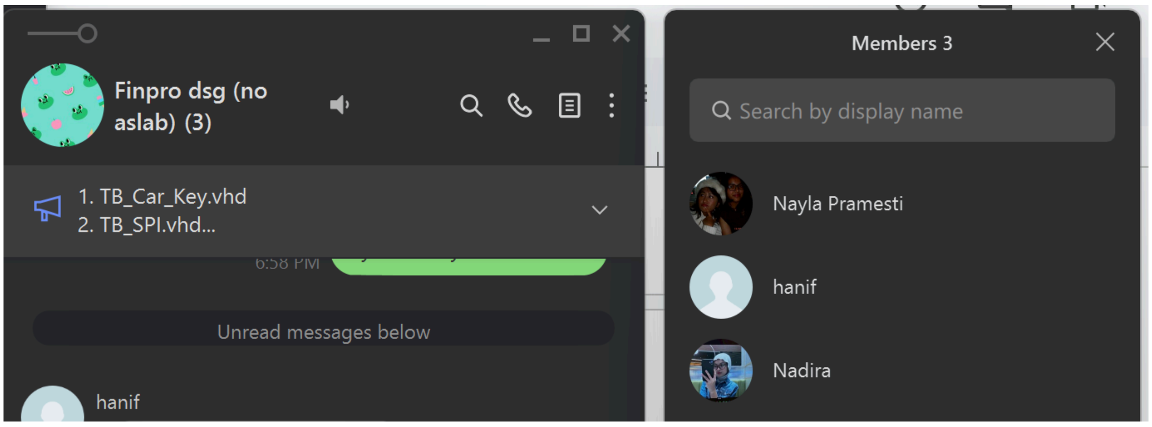


Fig. Key system

Appendix B: Documentation



Finpro dsg (no aslab) (3)

1. TB_Car_Key.vhd
2. TB_SPI.vhd...

6:58 PM

Unread messages below

hanif

Members 3

Search by display name

Nayla Pramesti

hanif

Nadira