ISTANBUL TECHNICAL UNIVERSITY ★ ELECTRICAL AND ELECTRONICS ENGINEERING

DESIGNING AND IMPLEMENTING SECURE AUTOMOTİVE NETWORK FOR AUTONOMOUS CARS

B.Sc. THESIS

Ömer Demirci
M. Enes SOLTEKİN

Department of Electronics and Communication Engineering

Electronic and Communication Engineering

DEC 2020

ISTANBUL TECHNICAL UNIVERSITY ★ ELECTRICAL AND ELECTRONICS ENGINEERING

DESIGNING AND IMPLEMENTING SECURE AUTOMOTİVE NETWORK FOR AUTONOMOUS CARS

B.Sc. THESIS

Ömer Demirci
(040160235)
M. Enes SOLTEKİN
(040150021)

Department of Electronics and Communication Engineering

Electronic and Communication Engineering

Thesis Advisor: Assoc. Prof. Dr. Sıddıka Berna Örs YALÇIN

DEC 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ ELEKTRİK-ELEKTRONİK FAKÜLTESİ

OTONOM ARAÇLAR İÇİN GÜVENLİ HABERLEŞME AĞI TASARIMI VE GERÇEKLEMESİ

LİSANS TEZİ

Ömer Demirci
(040160235)
M. Enes SOLTEKİN
(040150021)

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Elektronik vs Haberleşme Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. Sıddıka Berna Örs YALÇIN

ARALIK 2020

Ömer Demirci and **M. Enes SOLTEKİN**, B.Sc. students of ITU Electrical and Electronics Engineering 040160235 , 040150021 successfully defended the thesis entitled **"DESIGNING AND IMPLEMENTING SECURE AUTOMOTİVE NETWORK FOR AUTONOMOUS CARS"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Sıddıka Berna Örs YALÇIN** Istanbul Technical University	.....
Jury Members :		.....
		.....
		.....

Date of Submission : **20 DECEMBER 2020**
Date of Defense : **20 FEBRUARY 2021**

To my family

FOREWORD

We would like to thank my supervisor Assoc. Prof. Dr. Sıddıka Berna Örs YALÇIN for her guidance, kind advice, and help throughout our B.Sc studies.

December 2020 Ömer Demirci
. M. Enes SOLTEKİN

TABLE OF CONTENTS

Page

ABBREVIATIONS

AES	**:** Advanced Encryption Standard
MAC	**:** Message Authentication Code
FPGA	**:** Field Programmable Gate Arrays
CMAC	**:** Cipher Message Authentication Code
ECU	**:** Electronic Control Unit
DLC	**:** Data Length Code
ALU	**:** Arithmetic Logic Unit
RISC	**:** Reduced Instruction Set Computer
SecOC	**:**Secure Onboard Communication
PDU	**:**Protocol Data Unit
RTL	**:**Register Transfer Logic
AUTOSAR	**:**Automotive Open System Architecture
OBD	**:**On - Board Diagnostic
IP	**:**Intellectual Property
AXI	**:**Advanced Expandable Interface

LIST OF TABLES

Page

LIST OF FIGURES

Page

DESIGNING AND IMPLEMENTING SECURE AUTOMOTİVE NETWORK FOR AUTONOMOUS CARS

SUMMARY

Thanks to the advances in technology in recent years, the development of autonomous systems has gained momentum. With the cheaper sensors, the acceleration of the computing platforms, and the support of the software by the communities, autonomous systems started to be tested on vehicles with low fault tolerance.

Nowadays, autonomous vehicles have developed significantly and the margin of error of these vehicles carrying people is quite low. Errors can be sensor-based or computational, or they can occur intentionally when an externally accessible spy directs the on-board computer. With this intervention from outside, the people inside the vehicle may be injured or the accident may result in death. In this project, a system on a chip System on Chip (SoC) was designed to prevent external intervention by using Advanced Encryption Standarts (AES) encryption methods for data exchange between between in-vehicle sensors and electronic control units (ECU) of autonomous vehicles.

Secure on-board communication (SecOC) with a layered structure in the literature has been chosen to provide secure communication between in-vehicle electronic control units (ECU). The security algorithm of this software-based system has been implemented on Field Programmable Gate Arrays (FPGA) with verilog hardware design language (HDL). After the hardware implementation was run, the synthesized block was run with a program written in C using the Microblaze processor that can be implemented on the FPGA. Finally, an application was written in the application layer with the working blocks and tested.

OTONOM ARAÇLAR İÇİN GÜVENLİ HABERLEŞME AĞI TASARIMI VE GERÇEKLEMESİ

ÖZET

Son yıllarda teknolojide yaşanan gelişmeler sayesinde otonom sistemlerin geliştirilmesi ivme kazandı. Sensörlerin ucuzlaması, hesaplama platformlarının hızlanması, yazılımların topluluklar tarafından desteklenmesi ile birlikte otonom sistemler hata toleransı düşük olması gereken araçlar üzerinde de test edilmeye başlandı

Günümüzde otonom araçlar önemli ölçüde gelişme kat etmiş olup, içerisinde insan taşıyan bu vasıtaların hata payları oldukça düşüktür. Hatalar sensör veya hesaplama tabanlı olabileceği gibi dışarıdan erişim sağlayan bir casusun araç bilgisayarını yönlendirmesi ile bilinçli bir şekilde de meydana gelebilir. Dışarıdan gelen bu müdahele ile araç içerisindeki kişilere zarar gelebilir yahut kaza ölüm ile sonuçlanabilir. Bu projede otonom araçların araç içi sensörler ve elektronik kontrol üniteleri (ECU) arasında yaptığı veri alışverişini Gelişmiş Şifreleme Standartları (AES) kriptolama yöntemleri kullanarak dışarıdan müdaheleyi engelleyecek bir çip üzerinde sistem Çip Üzerinde Sistem (SOC) tasarımı yapılmıştır.

Araç içi elektronik kontrol üniteleri (ECU) arası güvenli haberleşmeyi sağlamak için literatürdeki katmanlı bir yapıya sahip güvenli yerleşik iletişim (SecOC) seçilmiştir. Yazılım tabanlı olan bu sistemin güvenlik algoritması Alan Programlanabilir Kapı Dizileri (FPGA) üzerinde verilog donanım tasarlama dili (HDL) ile gerçeklenmiştir. Donanımsal gerçekleme çalıştırıldıkta sonra sentezlenen blok, FPGA üzerinde uygulanabilen Microblaze işlemcisi kullanılarak C dilinde yazılmış bir program ile çalıştırılmıştır. Son olarak çalışan bloklar ile uygulama katmanında bir uygulama yazılarak test edilmiştir.

1. Introduction

One of the most attractive technological development is the evolution of the autonomous systems. The most remarkable one is the enhancements in automobile technologies.

A standart car includes more than 70 Electronic Control Unit (ECU)'s inside it. All these ECU's are establishing the data transfer between car components. The connection between these ECU's are provided by a two-wired communication protocol called CAN. Although there are some other protocols like FlexRay and LIN, CAN is the price-performance protocol.

Standart Controller area network (CAN) is sufficient and reliable, however it is not secure and accessible if there is a device connected to the CAN network physically. A spy node connected to the car network via On-board diagnostic (OBD) port can be able to sniff the network and manipulate it with injecting third party messages. This security problem is not an acceptable situation in autonomous cars. An autonomous car on the road is under a cyber attack may ends up with deadly accidents or serious injuries. Preventing these attacks requires a secure networking inside ECU's. SecOC, which is released bu AUTOSAR, is one of the secure on-board solution as a software layer. Secure Onboard Communication (SecOC) protects the network by keeping the network freshness and authenticity, which is for the integrity and authenticity of the messages.

This project aims to implement software based SecOC module into hardware on an Field-programmable gate array (FPGA). Literature search reveals that SecOC is the speedy and robust way of secure communication. The system is implemented on FPGA and connected with a soft-processor called Microblaze.

2nd chapter is about the tools and algorithms that used in this implementation. This includes environments, concepts, hardware and software components.

3rd chapter is the SecOC details, sub-blocks and explanations about SecOC.

4th and 5th chapters are about implementation on hardware and software, simulation results and hardware results and conclusion.

2. Foreknowledge

This chapter summaries the tools and concepts that used in this thesis. There are basics of concepts and definition of terminology for this thesis.

2.1 Controller Area Network

Controller Area Network (CAN) was at first made by German automotive system provider Robert Bosch within the mid-1980s for car applications as a strategy for empowering serial communication [1]. The objective was to form automobiles more dependable, secure and fuel-efficient whereas reducing wiring harness weight and complexity.

2.1.1 Overview

Controller Area Network (CAN) is an asynchronous serial communication protocol which follows ISO 11898 standards and is widely accepted in automobiles due to its real time performance, low price, reliability and compatibility with wide range of devices [2]. CAN is a two wire differential bus with data rates up to 1 Mbps and offers a very high level of robust data transfer between nodes and replaces the harness with just 2 wires. Its robust, low cost and versatile technology made CAN applicable in other areas of applications where inter processor communication or elimination of excessive wiring is needed. Even though it has a wide application in different areas,one of the most important industry that uses CAN is the Automotive industry.

2.1.2 Controller Area Network Data Frame

A CAN frame consist of three basic units. First one is the ID (Identity) number, which is the identification number of an ECU. Second part is the DLC, which stands for the number indicates the byte number that will be transmitted with a frame. The last one is data. There are other flags and acknowledge bits inside the frame as it can be seen

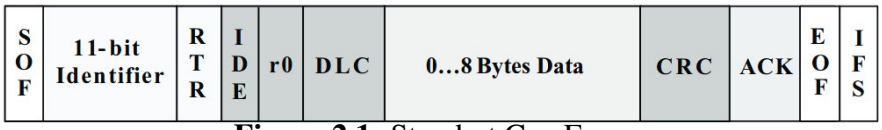

Figure 2.1: Standart Can Frame

from the Figure 2.1 [3]. This project mainly focuses on these three element of the CAN frame.

2.2 Advanced Encryption Standarts (AES)

Cryptography is a type of protection for computers that converts information from Its original form into an unreadable encrypted form. The key features that distinguish one algorithm from another are its ability to protect data against unauthorized use and encryption/decryption efficiency when transmitting and receiving final data [4]. Cryptographic algorithms are usually divided into two main algorithms. Symmetric Encryption Algorithms are the first group, and Asymmetric Encryption Algorithms are the second. The distinction between these two algorithms is that a single key for both encryption and decryption is used by the symmetric algorithms, whereas the asymmetric algorithms use two separate keys, i.e. One to encrypt and the other to decrypt.

One of the most popular symmetric encryption algorithms is Advance Encryption Standard (AES) [5]. AES was selected as a method of encrypting electronic data by National Institute of Standard and Technology (NIST) in 2001. Now used worldwide, this standard continues to be an active research area for implementation on both software and hardware platforms. The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES (Triple Data Encryption Standard (DES))
- Provide full specification and design details
- Software implementable in C and Java

All the computations are performed by AES on bytes rather than bits. Therefore the 128 bits of a plain text block are treated as 16 bytes by AES. These 16 bytes are organized as a matrix in four columns and four rows to process.

2.2.1 Encryption Process

It is possible to break the AES encryption process into three phases: the initial round the main round, and the final round. In various variations, all the phases use the same sub-operations as can be seen from Figure 2.2 [6].

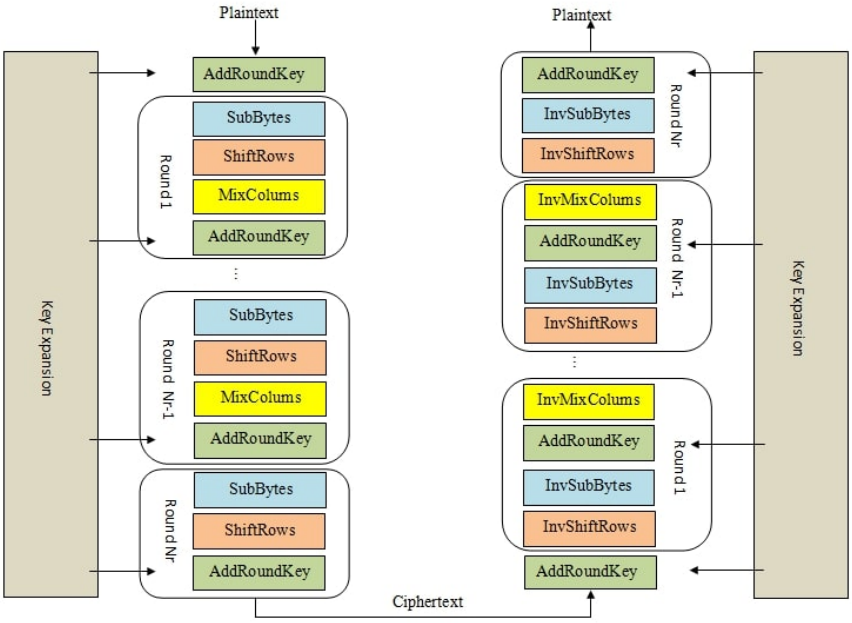

Figure 2.2: AES algorithm Schematic

1. Initial Round
    - AddRoundKey
2. Main Rounds
    - SubBytes
    - ShiftRows
    - MixColumns
    - AddRoundKey

3. Final Round

    - SubBytes
    - ShiftRows
    - AddRoundKey

2.2.1.1 AddRoundKey

The AddRoundKey operation is the only phase of AES encryption that directly operates on the AES round key. In this operation, the input to the round is exclusive-ored with the round key.

2.2.1.2 Byte Substitution (SubBytes)

By looking up a fixed table (S-box) provided in the design, the 16 input bytes are substituted. The result is four rows and four columns in a matrix.

2.2.1.3 Shiftrows

Each of the matrix's four rows is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. The change takes place as follows:

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other. As in shown Figure 2.3 [6]

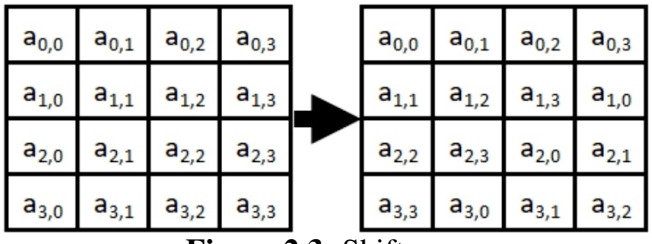

Figure 2.3: Shiftrows

2.2.1.4 MixColumns

Each four-byte column is now transformed using a special mathematical function. This function takes the four bytes of one column as input and outputs four completely new bytes, replacing the initial column. Another new matrix consisting of 16 new bytes gives the result.As in shown Figure 2.4 [6]

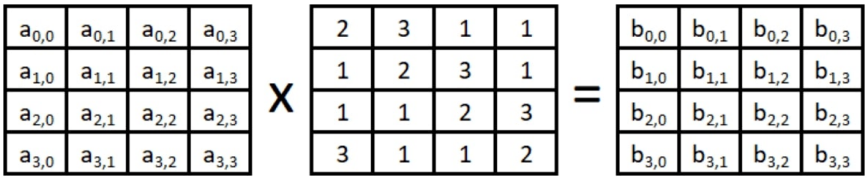

Figure 2.4: Mixcolumns

2.3 Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key. Sender and receiver both calculates the fixed-length code from data and key. Output of this calculation called MAC .Basically, a MAC is an encrypted checksum generated with message that is sent along with a message to ensure message authentication . The concept can be seen from the Figure 2.5 [7].

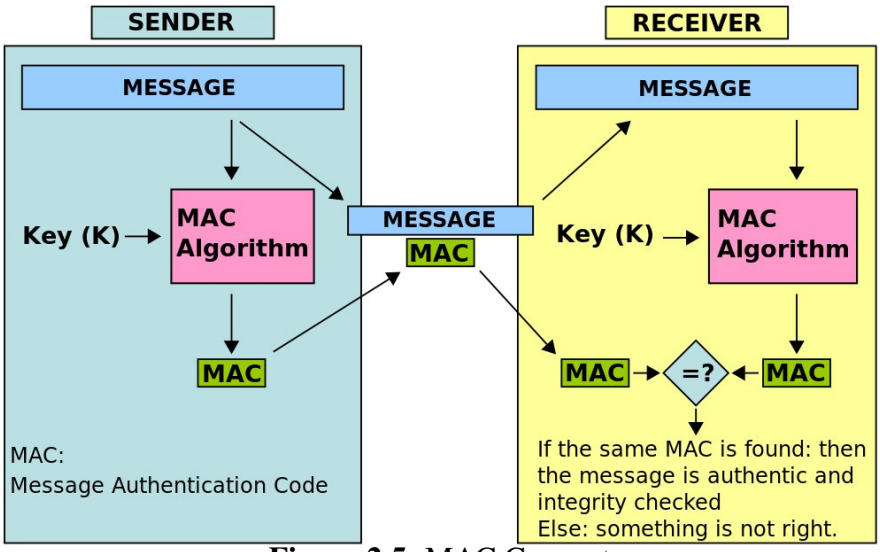

Figure 2.5: MAC Concept

2.3.1 Cipher-Message Authentication Code Generation

Cipher-MAC is a technique for constructing a message authentication code from a block cipher. The message is encrypted with a sequential block cipher algorithm to create a chain of blocks such that each block depends on the proper encryption of the previous block as shown in the Figure 2.6 [7]. This interdependence ensures that a change to any of the plain text bits will cause the final encrypted block to change in a way that cannot be predicted or counteracted without knowing the key to the block cipher. Message is divided into fixed length sub-messages *m*1, *m*2, ... *mn* and last block *mx* is the padded zero number of the *mn* last block after division.

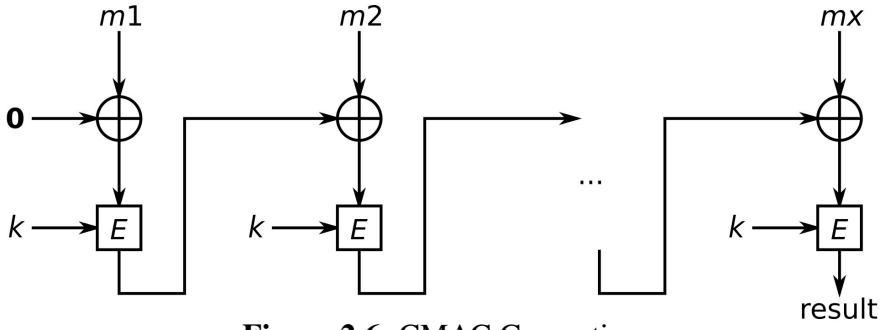

Figure 2.6: CMAC Genaration

2.4 Field Programmable Gate Arrays (FPGA)

Field-Programmable Gate Arrays (FPGAs) are pre-fabricated silicon devices that can be electrically programmed to become almost any kind of digital circuit or system [8]. They are formed by a two-dimensional array of programmable logic cells and managed switches. Logic cells can be configured to implement a simple function. Besides, connections can be established between programmable keys and logic cells. Digital hardware is implemented by programming logic cells and switches in this way. After the circuit is designed and synthesized using hardware description languages such as Verilog,Very High Speed Integrated Circuit Hardware Description Language (VHDL), the data string containing the desired logic cell and switch configuration is embedded in the FPGA with the help of a cable .

They provide a number of compelling advantages over fixed-function Application Specific Integrated Circuit (ASIC) technologies such as standard cells [9]. ASICs typically take months to produce and cost the first device hundreds of thousands to millions of dollars; FPGAs are set up in less than a second (and can also be

reconfigured if an error is made) and cost between a few dollars to a few thousand dollars anywhere.

FPGAs consist of an array of programmable logic blocks, including general logic, memory and multiplier blocks, of potentially different forms, surrounded by a programmable routing fabric that enables programmable interconnection of blocks. In FPGA, the "programmable" concept means an ability to program a feature into the chip after completion of silicon manufacturing. This customization is made possible by the programming technology, which is a method that can cause a change in the behavior of the pre-fabricated chip after fabrication, in the “field,” where system users create designs.

2.5 Nexys4 DDR FPGA Evaluation Board

The Nexys 4 DDR board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx® [10]. The Nexys A7 is the new name for our popular Nexys 4 DDR board. The Artix-7 FPGA is optimized for high performance logic, and offers more capacity, higher performance, and more resources than earlier designs. The appearance of the Nexys DDR4 board can be seen in Figure 2.7 [10]

The features of the Artix-7 device can be listed as follows:

- Xilinx Artix-7 FPGA XC7A100T-1CSG324C
- 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops
- 4,860 Kbits of fast block RAM
- Six clock management tiles, each with phase-locked loop (PLL)
- 240 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)
- 16 user switches

- USB-UART Bridge
- 12-bit VGA output
- 3-axis accelerometer
- 16 user LEDs
- Two 4-digit 7-segment displays
- Micro SD card connector
- PDM microphone
- Temperature senso
- Digilent USB-JTAG port for FPGA programming and communication

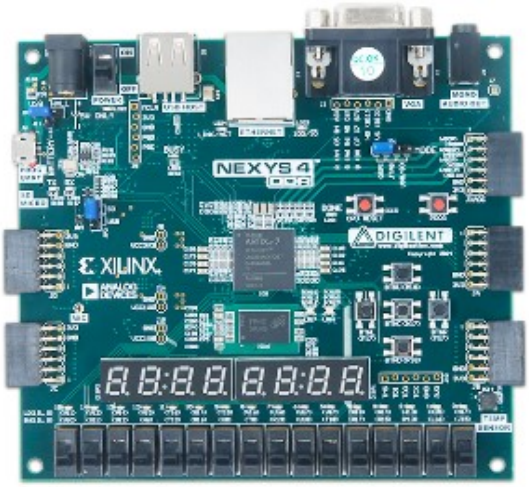

Figure 2.7: Nexys4 DDR

2.6 Microblaze Soft Processor

The MicroBlaze is a 32-bit soft microprocessor core designed for Xilinx field-programmable gate arrays (FPGA) with harward architecture [11]. For this project, microblaze soft processor is choosen for easy of usage with ip blocks. Microblaze supports expandable peripheral interface which is called AXI interface [12]. AXI enables the multiple peripheral connection to the microblaze processor . The structure of the microblaze is shown in the Figure 2.8 [11]. There

are 32 bit general purpose registers, ALU, program counter, instruction decoder, instruction buffer and some special registers. Design can be expandable with some other peripherals such that barrel shifter, multiplier and divider.

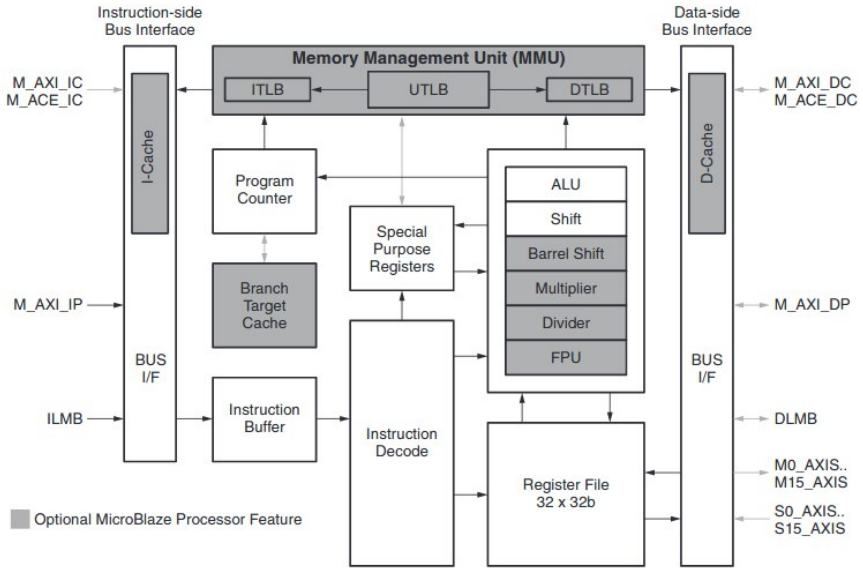

Figure 2.8: Microblaze Structure

Microblaze has a 32 bit parallel pipelined RISC (Reduced instruction set computer) based processor [13]. There are three steps fetch decode and execute when processor runs instructions. This procedure accelerates the processor by paralleling the operations.

2.7 Verilog Hardware Description Language

The Verilog language provides the digital system designer with the ability to define a digital system at a wide range of abstraction levels, while at the same time providing access to computer-aided design software to help in the design process at these levels. Structurally similar to the C language will be preferred in digital system design [14].

2.8 XILINX VIVADO

Xilinx Vivado Environment is an interface software used to program FPGAs developed by Vivado Xilinx company [15]. This package, which includes many programs that enable to make block-level design, facilitate the packaging of the designed hardware, design hardware over Matlab, turn the code written according to a certain rule with

the C language into hardware, this package has managed to become a great solution in FPGA design by eliminating the errors and accelerating as the updates arrive [16]. Thus, Xilinx has provided great convenience to designers by collecting solutions from many areas with Vivado in one package. WebPack Edition, the free version of Vivado's 2019.1 version, was used in this project. Vivado supports free FPGA boards such as Artix®-7, Kintex®-7, Kintex UltraScale ™, Zynq®-7000 All Programmable SoC [15]. It does not support old serial FPGA cards. The program automatically asks which tools to install so that it does not take up too much space during installation. Since the crypto design will be run with a processor in this project, the SDK tool has been installed.

2.8.1 Creating project

It has a very simple interface compared to Vivado ISE. When the program is run It is divided into 3 sections as quick start, tasks and learning center. A new project is created from the quick start section as a project type, RTL project is selected for adding Verilog/VHDL modules or IP block designs. Respectively, new source files and if desired physical or temporal constraint files are added. Source file language as Verilog or VHDL can be selected. The modules created in this project are written in Verilog language. After the project type is selected, the project name is given. Then it is selected on which FPGA board the project will be implemented. Opening window is as in the Figure 2.9 [15]

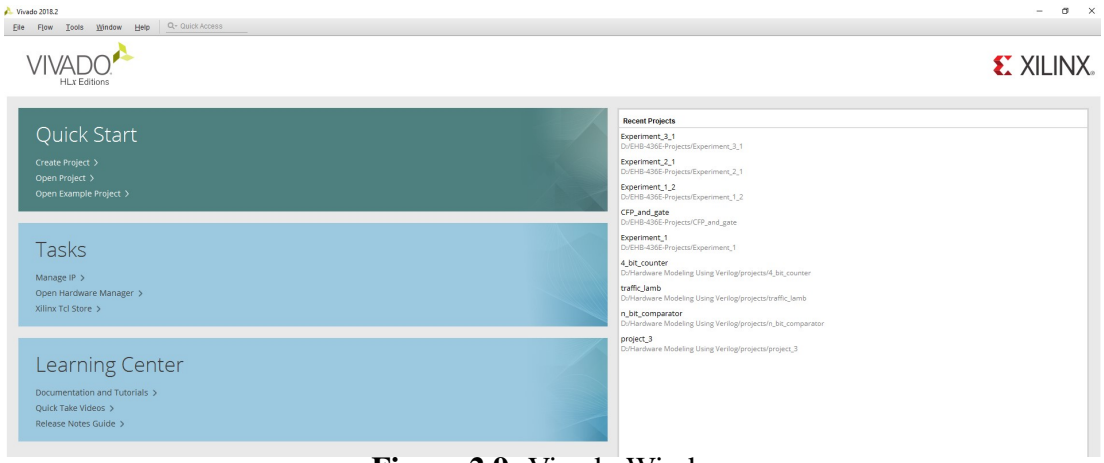

Figure 2.9: Vivado Window

2.8.2 Flow Navigator

Required transactions for the project are listed in Flow Navigator. As shown in Figure 2.10 [15] There are 6 basic sections under Flow Navigator: IP Integrator, Simulation, RTL analysis, Implementation and Program and Debug. Under Project Manager, under Settings> General tab To be implemented on the FPGA board and with which hardware description language it will be compiled with adjustable.

From Project Manager> Add Sources,New or existing design, simulation or constraint files are included in the project

Under Project Manager> IP Catolog tab, Intellectual Property (IP), ready by Xilinx, are available out of the box.Microprocessor in this project MicroBlaze Debug Module (MDM) IP will be used as.

It can be created new IPs in IP Integrator section or block designs can be produced with ready IPs offered by Xilinx.

In the Simulation Section, under the Run Simulation tab, behavioral, post synthesis and Post-implementation simulations can be done with the written testbench in simulation files. In behavioral synthesis, whether the design works functionally or not is tested.

RTL connections of the design are performed in the RTL Analysis section. How many cells fit the design, the number of input-output ports are specified.

In the Synthesis section, it is synthesized by making high level and low level optimizations in accordance with the design constraints. Open Synthesis Design section time constraints can be added to the project. Likewise Project Manager> Add Sources Time constraints file can be added from within. In addition, a synthesis utilization report can be created to get detailed information about the resource usage in design. At the same time, the total delays between each path can be viewed from the report timing tab.

In the Implementation section, considering the constraints of the created design, FPGA mapping is done on it. The hardware of the systems designed by the user detailed information such as how much space it occupies on it, how much line delay it has, and

where the design will be placed in the FPGA offers. In addition, primitives to be used on the device can be defined manually in this section.

With Generate Bitstream in Program and Debug section, bit file is created. By connecting FPGA card from Open Hardware tab FPGA is programmed.

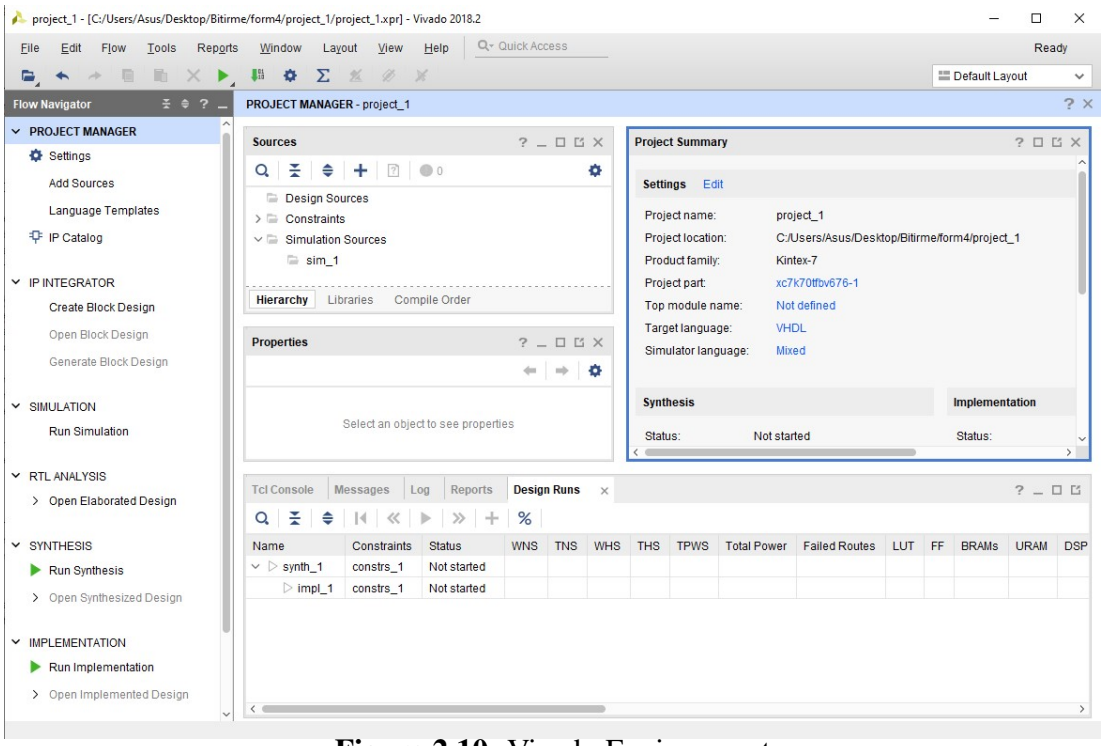

Figure 2.10: Vivado Environment

2.9 XILINX Software Development Kit

Software Development Kit (SDK) is an interface environment developed by Xilinx to realize the software design of microprocessor-centric digital system designs designed in Vivado environment [17]. In earlier versions of Xilinx's design environments, SDK was used together for software design and XPS to develop hardware. Later, Xilinx included the XPS environment in Vivado and gathered all of the hardware design processes in one place. SDK is used only for software design for designed hardware. Libraries of the user equipment and peripheral units of the system designed in Vivado environment are produced and the first step is taken in software design. At the same time, the user can easily control the microprocessor by adding the libraries produced by the SDK to the said software project.

- Feature-rich C/C++ code editor and compilation environment

- Project management
- Application build configuration and automatic Makefile generation
- Error navigation
- Well-integrated environment for seamless debugging and profiling of embedded targets

Features are major features provided by SDK.

Hardware designed with HDL or schematic drawings in Vivado environment is packaged as a special IP in Vivado environment and added together with other ready IPs that are desired to be added to the microprocessor-centric digital system design. In Vivado environment, the system whose hardware structure is completed is sent to the SDK environment and the software design is made after the libraries are automatically produced at this stage. Finally, after the hardware and the software designed to control these hardware are completed, the hardware file with the "bit" extension containing the hardware information and the software file with the extension "elf" are combined and sent to the FPGA via the SDK.

3. Secure Onboard Communication

In this chapter, it is going to be presented Secure Onboard Communication which is developed by AUTOSAR. SecOC concept and components of SecOC are given in detail.

3.1 Secure Onboar Communication Spesifications

The Secure Onboar Communication (SecOC) module is responsible for generating and verifying secure messages passed between ECUs over the in-vehicle communication network [18]. The AUTOSAR SecOC specification is designed with ‘resource-efficient and practicable authentication mechanisms in mind, so that legacy systems can also profit from it with minimal overhead. SecOC guarantees the freshness and authenticity of a dataframe with freshness counter CMAC based authentication mechanism.

SecOC module provides Protocol Data Unit (PDU) message integrity and authentication. There is a MAC based encryption and freshness based protection mechanism against replay attacks. SecOC does not provide an encrypted data. Every ECU connected the CAN network can listen messages that broadcasted on BUS. Main goal is the authentication of the message. Sender ECU has same pre-shared key with the receiver and they produces same MAC and FV with same id and data. The comparison is succeeded if exact match satisfied by receiver. Otherwise incoming data is not meaningful and do not be used in the controller.

3.1.1 AUTOSAR

AUTOSAR is a worldwide partnership of defines standards for software architecture [18]. It is a global development partnership of automotive interested parties founded in 2003. It pursues the objective to create and establish an open and standardized software architecture for automotive electronic control units

(ECUs). Goals include the scalability to different vehicle and platform variants, the consideration of availability and safety requirements, a collaboration between various partners, and maintainability during the whole product life cycle.

AUTOSAR uses three layer architecture:

- Basic Software
- Runtime Environment
- Application Layer

This project focuses on Basic Software layer . The Figure 3.1 [19] shows the organisation schematic of the SecOC inside basic software stack.

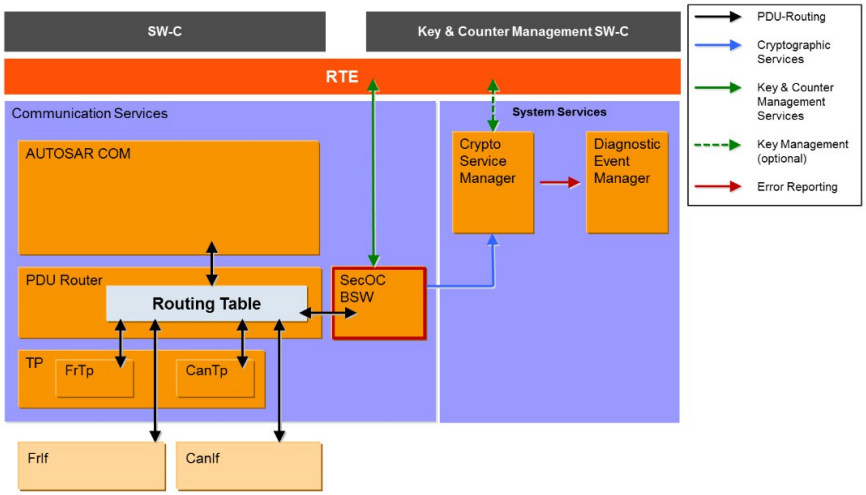

Figure 3.1: SecOC in AUTOSAR Basic Software Stack

3.1.2 Protocol Data Unit

PDU of the SecOC is shown in the Figure 3.2. The payload of the standart CAN filled with the actual data, truncated freshness value and truncated MAC. Standart CAN supports up to 8 bytes payload [20] .

3.1.3 Freshness Value

Freshness value is the key component for replay attacks. The Freshness value is generated and updated in a random sequence and it is added to current PDU [20]. The receiver updates itself by incoming freshness flags and compares it with the freshness

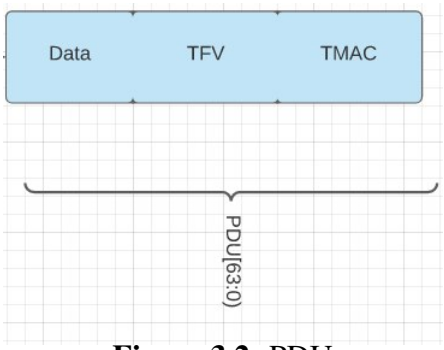

Figure 3.2: PDU

inside the receiver ECU. This process makes the system safe even if a spy node knows the key and the message, it can not be able to manipulate receiver side without knowing the freshness value.

Freshness value Consist of 4 parts trip counter, reset counter, message counter and reset flag as in shown Figure 3.3 [20]. Trip counter is monotonic counter that counts up and down according to the up and down flags produced in sender side. Reset counter counts the reset number. Flags updates the freshness counter and reset flag resets the freshness.

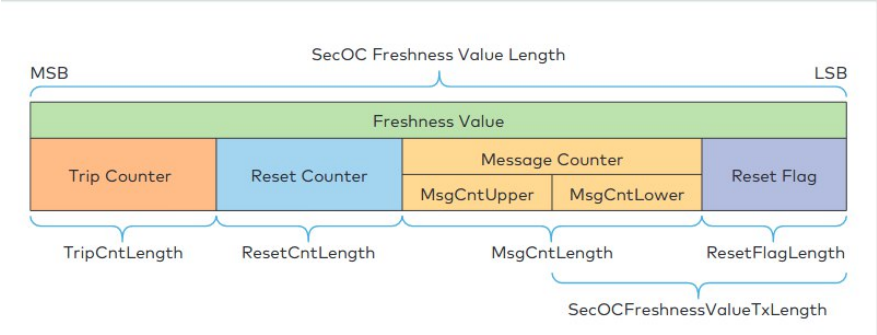

Figure 3.3: Freshness Value

3.1.4 Message Authentication Code Generation Algorithm

As shown in Figure 3.4 [21], the first step of CMAC calculation is to derive sub-keys from the symmetric key using the sub-key generation process [21]. The message is partitioned into a sequence of equal-sized data blocks, and AES symmetric encryption is applied sequentially to each data block, each being XOR-ed with the previous AES result. The MAC is obtained by truncating the last encryption result according to the MAC length parameter. Data, id and freshness value can be used as sub-message

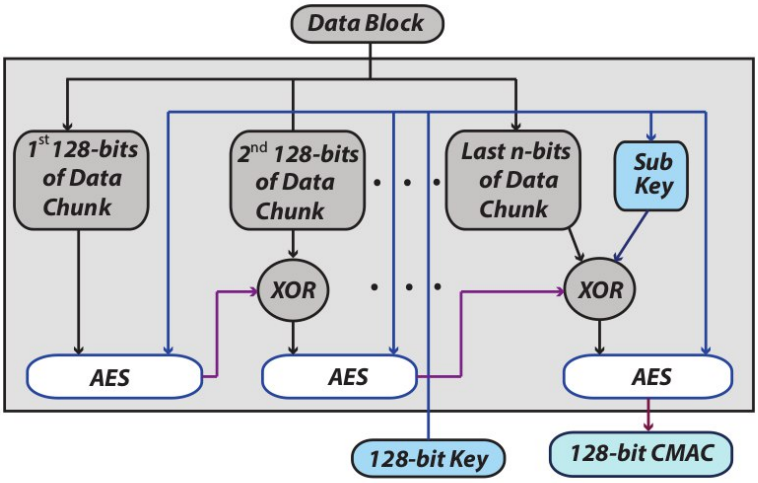

Figure 3.4: MAC Generation

blocks. Last n bits are padded with zeros until the 128 bit value achieved. Remained AES process is done with padded n bits and sub-key output.

3.1.5 Sender

The sender part shown in the Figure 3.5 [21] takes the freshness value, data and id values as input and converts them to the mac with mac generation block by using shared key. The output of the sender module is the combination of data, truncated freshness value and truncated MAC. SecOC module routes sender's output to the CAN driver to transmit receiver side.

3.1.6 Receiver

The receiver module schematic is given in the Figure 3.6 [21]. First, slave ECU receives the PDU from sender ECU. PDU splits into data, FV and MAC. Second step is comparing freshness values. After generating MAC from given inputs and comparing with received MAC value, if both FV and MAC are matched, receiver side authenticates the sender and allows the data to process by ECU.

3.2 Literature Survey on SoC Implementation of SecOC

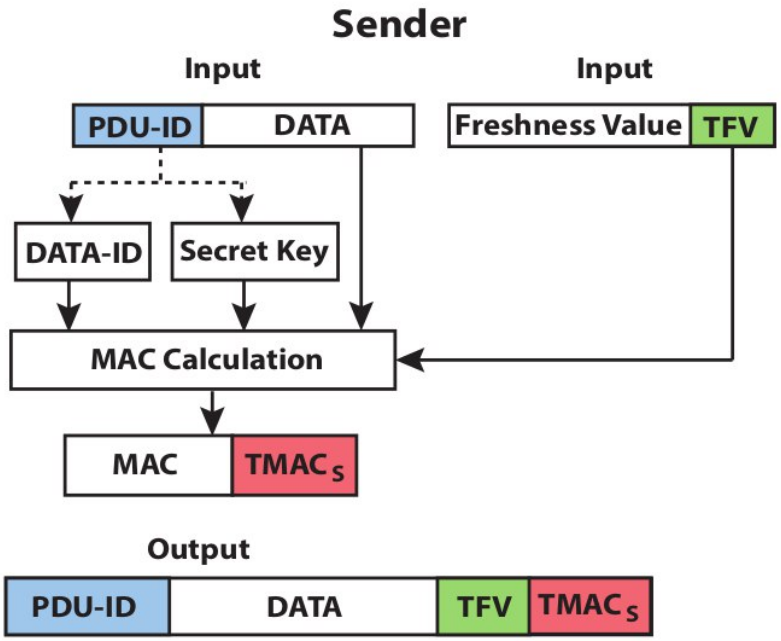

Figure 3.5: Sender Structure

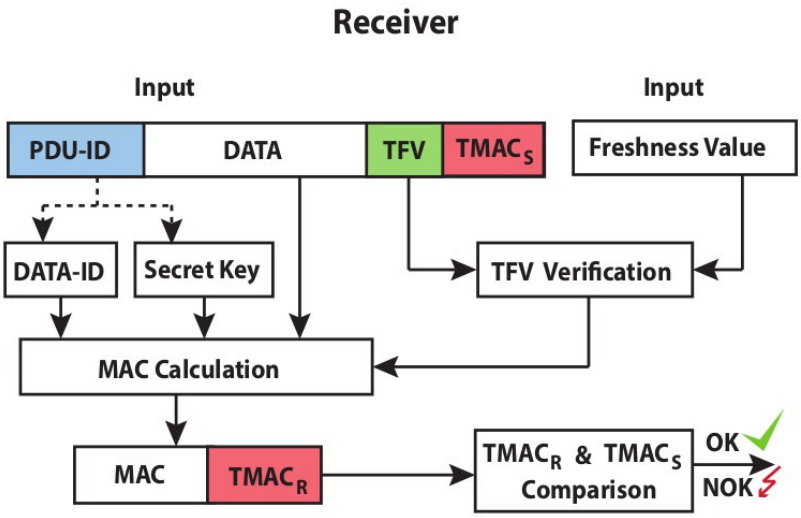

Figure 3.6: Receiver Structure

The combination of safety and security issues is still a main research focus and many different projects have formed around the goal to create a secure automobile. Security solutions for Controller Area Network (CAN) authentication were reviewed. All studies in this field were tried to be researched and observed.Studies will be classified under two headings.In the world and In turkey

3.2.1 Worldwide

Sigrid Gürgens and Daniel Zelle of Fraunhofer Institute, Germany, says that not only the connections to the outside world pose new threats, also the in-vehicle communication between ECUs, realized with bus systems like CAN, needs to be protected against manipulation and replay of messages [22]. Multiple countermeasures were presented in the past making use of Message Authentication Codes and specific values to provide message freshness, most prominently AUTOSAR's Secure Onboard Communication (SecOC). They also researched the HMAC based authentication procedure developed by Nürnberger and Rossow. However, they did not work on any hardware or software implementation of SecOC in their studies .

Thomas Rosenstatter, Christian Sandberg and Tomas Olovsson says in their paper that the CAN BUS system is one of the most common BUS technology and therefore has become the first target of attackers [23]. In this paper, they set up a system consisting of 2 receivers and 1 sender using Freescale MPC 5646C microcontrollers. AUTOSAR 4.3 crypto stack software module is used in this system.

Also, Qingwu Zou and her team researched Secure CAN Communication for Automotive Applications [24]. A deeper look into the freshness and synchronization method is carried out in their paper.In This paper refers to Infineon AURIX 32-bit microcontrollers. The hardware setup mainly consists of three application boards of TC234LP, one represents a sender ECU, one represents a receiver ECU, and one is used as 'man in the middle' attacker ECU.

Stefan Seifert and Roman Obermaisser published a paper on providing more secure communication for future cars [25]. They proposed a solution that also includes FlexRay system, which is another system that provides in-vehicle communication. As they mentioned on paper To detect attacks against the automotive networks such

as CAN and FlexRay, they introduce the concept of the so called ”security gateway” which is part of the automotive architecture and is located on transition points, where different networks connect with each other.

Ali Shuja Siddiqui, Yutian Gui Jim Plusquellic and Fareena Saqib propose a hardware based secure and trusted framework that implements lightweight PUF based mutual authentication and secure encryption over the insecure communication channel [26]. They emphasized the following 3 basic therates of CANBUS communication: eavesdropping, spoofing, and unauthorized access through telematics attack. After hardware design in Xilinx Vivado suite environment done, Experimental setup is installed on Xilinx Kintex KC705 FPGA Evaluation Board. They made an AES-based encryption.

In the paper they wrote, Mehmet Bozdal, Mohammad Samie and Ian Jennions stated that the CANBUS system is very vulnerable to cyber attacks and the CANBUS system is very critical for new generation vehicles, and they conducted a detailed research on the measures that can be taken against this problem [27].

Giampaolo Bella, Pietro Biondi, Gianpiero Costantino, Ilaria Matteucci stated in their articles that they developed the TOUCAN protocol to make the CANBUS system more secure [28]. Their prototype implementation exhibits performance on a STM32F407Disco board.

3.2.2 In Turkey

In Turkey, the number of research on CANBUS communication system shows lack of research on this topic. Besides there are few articles about the CANBUS system,there is no article or paper could be found regarding the SecOC protocol

3.2.3 Litarature Analysis

Many researchers mentioned in Section 3.2.1 have said that the CAN BUS system is vulnerable. Many solutions have been proposed in this regard. In order to make the CAN BUS line safer, the SecOC protocol developed by AUTOSAR was examined

in the software layer by some researcher , and even the basic CAN BUS system was installed using microcontrollers and implemented on this setup.

In this project, the SecOC protocol was designed using Verilog hardware description language on the Xilinx Vivado Program, tested on the designed module and checked in simulation, and this module was produced as custom IP in Vivado and implemented on an FPGA for the first time.

4. SoC Implementation of SecOC on an FPGA

This chapter mentions about implementation flow of the SecOC modules. There is hardware and software co-design in this project. Design is also tested on an FPGA.

4.1 Hardware Designs of Modules

The implementation of the Secure Onboard Communication (SecOC) module first started with the design of the sub-modules that make up this module, using the Verilog hardware definition language in the vivado program. After the hardware design phase is completed, testbench simulation modules have been written for each module to test their accuracy.

4.1.1 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is used as the encryption algorithm in the SecOC module. AES algorithm has been written by many people in verilog hardware design language. AES module written by Kalpataru Mallick was used in this project. The AES encryption process is described in detail in Section 2.2

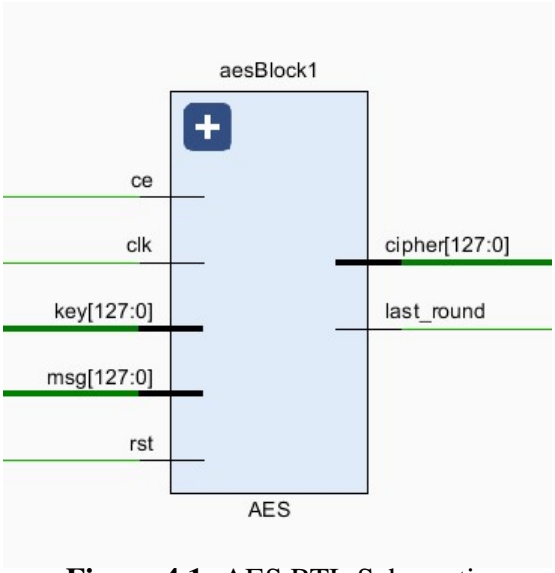

Figure 4.1: AES RTL Schematic

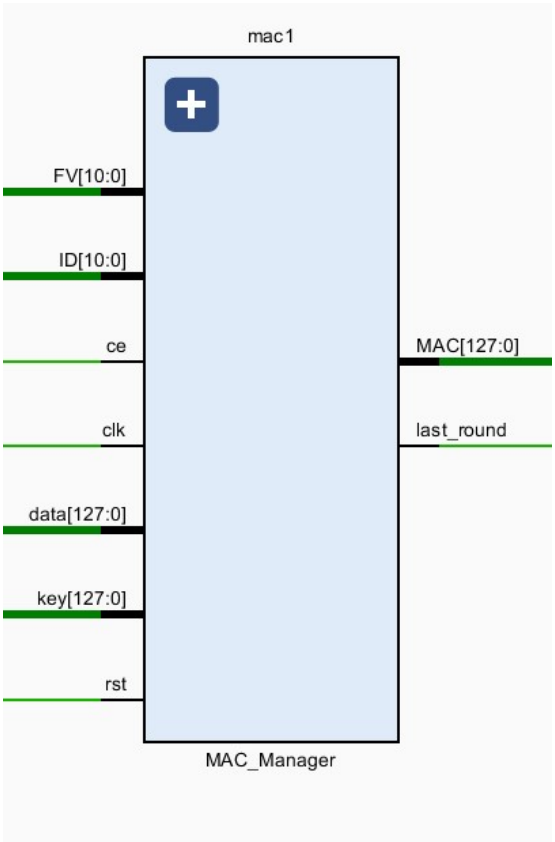

Figure 4.2: MAC Manager RTL Schematic

AES module has 128 bit key and msg inputs. Also, each AES block contains ce enable input signal. In addition, after the encryption process is completed, the 128 bit chipher is given as a output signal. The AES encryption process consists of 10 rounds. So when the last round is reached, the AES module sets the lastround signal. Thus, the ce input signal of the next AES module is activated by lastround signal of previous module.

4.1.2 Message Authentication Code Manager Module

The MAC Manager module contains 4 AES blocks. The chipher output generated by the first AES block is used as the msg input of the other AES block. MAC Manager module has 11 bit FV, 11bit ID, 128 bit data and 128 bit key inputs. The data coming into the MAC module is given as msg input to the first AES block. In addition, the key value is given as an input to each AES block. Then the ID and FV inputs coming to the MAC block are combined in the messagesub2 section and put into xor logic operation with the chipher, which is the output of the first AES block. The XOR output

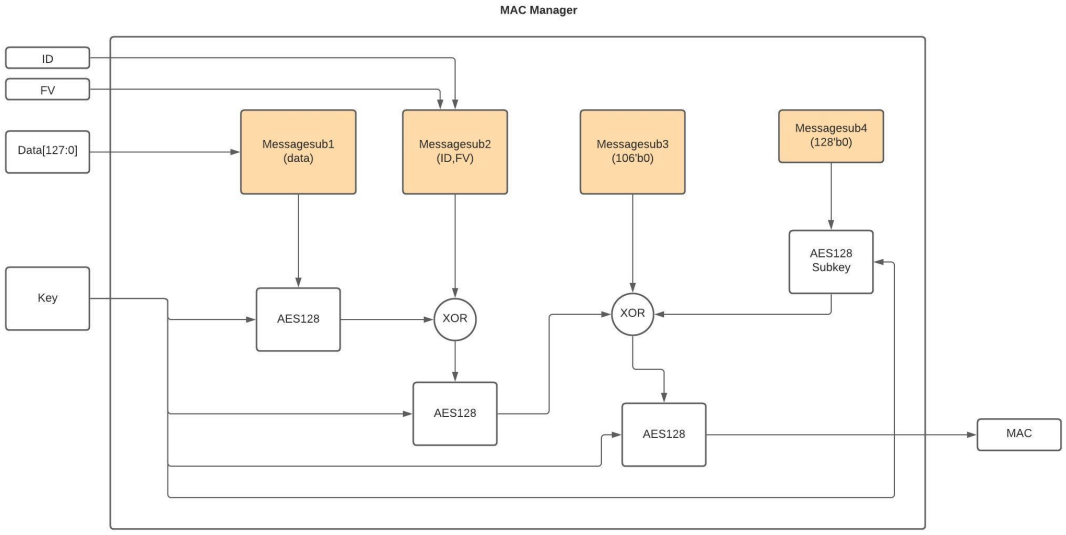

Figure 4.3: MAC Manager Algorithm

is given as the msg input to the second AES block. A 106-bit 0 value is created in the Messagesub3 section. This value is obtained by subtracting the ID and FV lengths from 128. This method is used in classical encryption algorithms. By the way, fourth AES module is used to generate Subkey. The 128-bit 0 value and the key are given as input to the AES block generating subkeys. The chipher value produced by the second AES block, subkey produced by fourth AES block and 106 bit 0 is put into the exor logic operation. Exor output is given as msg input to the third AES block. the output of this AES block is given as the MAC output.The algorithm of the MAC manager module is as in Figure 4.3. The ce (chip enable) input of each AES block depends on the lastround output of the preceding AES block and the system ce. The ce input of the first AES block is connected to the MAC ce input. Also, the lastround output of the last AES block is given to the lastround output of the MAC Manager module. Thus, AES modules are activated serially.

4.1.3 SecOC Sender Module

SecOC Sender module has 11 bit FV, 11 bit ID, 128 bit data, 128 bit key and 1-bit ce, rst inputs. Also, this module has 64-bit PDU data and lastround signal as a output. Under the SecOC Sender module, the MAC Manager module is called as a sub module and all inputs are given to the inputs of the MAC module. Last 21 bits of 128-bit MAC message generated by MAC Manager are taken as temp-mac. SecOC Sender Module

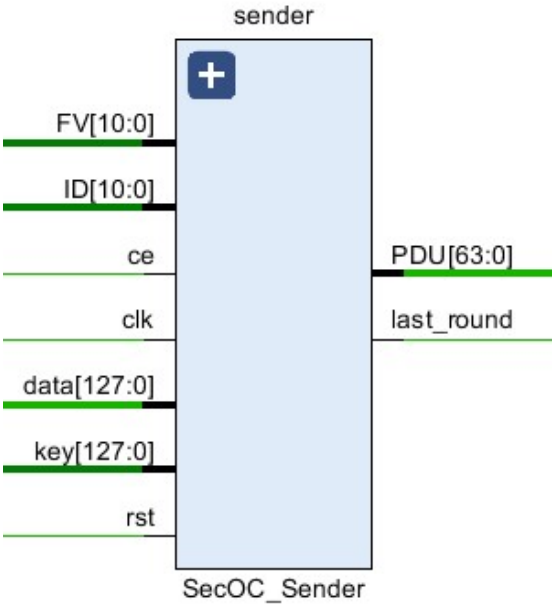

Figure 4.4: Sender Module RTL

generates PDU (Payload Data Unit). The PDU is consists of 32 bit data, 11 bit FV and 21 bit tempmac message. The lastround output given by the MAC Manager is also output as the SecOC Sender module.

The data to be entered into the SecOC module is actually 32 bits. But it is padded with a 128-bit 0 and a 128-bit value is given as data. Therefore, the last 32 bits of the data were taken while creating the PDU.

4.1.4 SecOC Receiver Module

SecOC Receiver Module has 11 bit FV, 11 bit ID, 64 bit PDU, 128 bit key and 1 bit ce and rst inputs.Also, has 128-bit dataout and 1-bit status as output. Under the SecOC Receiver module, the MAC Manager module is called as a sub module. 32 bit data is taken from the PDU data and put into padding with 128 bit 0. The MAC Manager generates a MAC value. A temp-mac value is created again by taking the last 21 bits of this produced MAC value. Then the tempmac value in the PDU is compared with the newly generated this value. After FV verification is done, the module sets the status output. In addition, the verified data is given to the data-out output.

4.1.5 SecOC Top Module

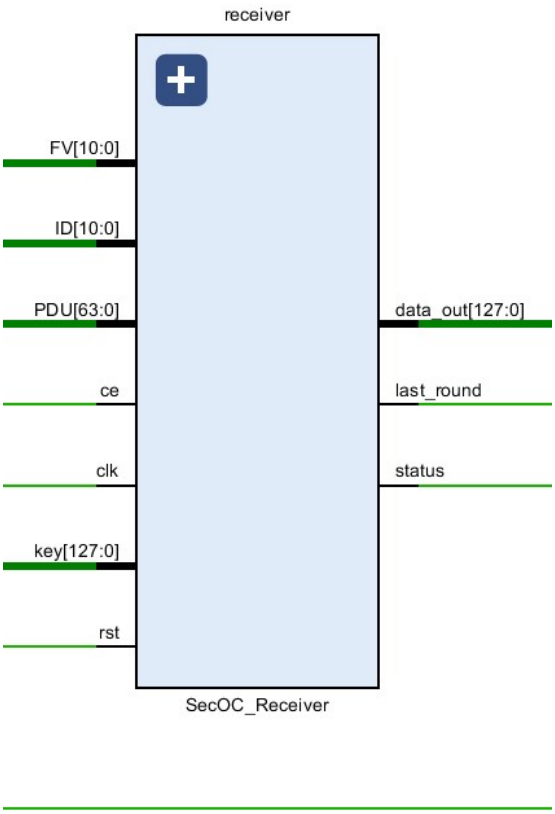

Figure 4.5: Receiver Module RTL

SecOC module has 128 bit data-in, 11 bit ID, 11bit FV 128 bit key, 64 bit PDU and 1 bit ce and rst inputs. Also SecOC module has 1 bit mode-select input. As a output, SecOC module has 64 bit PDU-out,32 bit data-out and 1 bit status and last-round. Under the SecOC module, both SecOC-receiver and SecOC-sender modules are called as sub-modules. According to the mode-select input, it is decided which of these sub-modules will be active. If the sender module is activated, data-in, ID, FV and key inputs will be used and PDU will be produced as output. This generated PDU will be given to the output of the SecOC module as PDU-out. If receiver mode is selected, the SecOC-Receiver module will be active and PDU-in, ID, FV and key input will be given. Verified data-out and status will be output.

4.2 Packaging Intellectual Property (IP)

Custom IP creator tool of the vivado provides IP-centric design flow that enables the compact and reusable packaging of functions or algorithms. IP packager tool is based on IP-XACT standart which is a unique design reuse feature. The packager tool

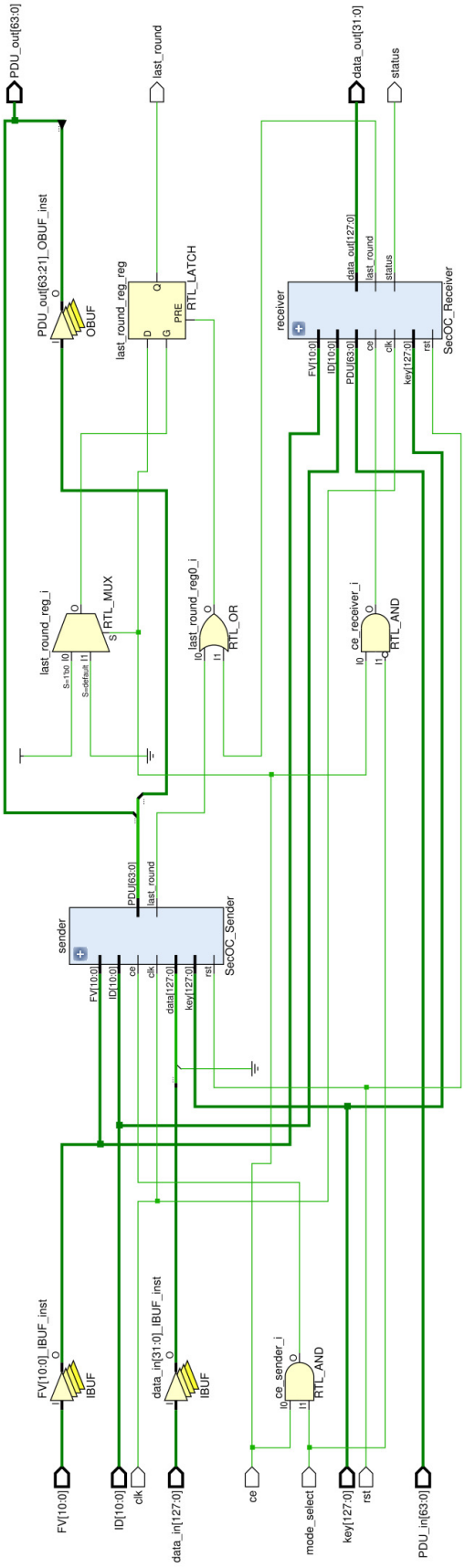

Figure 4.6: SecOC Top Module RTL

eligible to design an IP in any steps of the design flow and integrate it with the actual design as system-level IP [29].

4.2.1 Creating a Custom Intellectual Property (IP) with IP Integrator

A custom AXI IP core is created in vivado projects and verilog HDL codes of the SecOC module is integrated with that custom block. After completing the packaging, Block design is constructed with SecOC custom IP and Microblaze as shown in the Figure 4.7. AXI connections are completed with vivado auto-complete function. Microblaze is the controler of SecOC module. A code written in C controls the write-read operations of the SecOC IP's registers. Registers have 32 bit width. This project required 16 registers.

4.3 Software Layer

The bit stream is created with the implementation of the block design. Created bit stream is transferred to the XSDK tool for writing a program in C language. Application project created and a source file added to the project as shown in the Figure 4.8. The C code basically writes the data coming from ECUs to the connected registers of the SecOC module by addresses. After writing to the registers, the software waits until the module finishes the operation. The results appears when the operation succeeded.

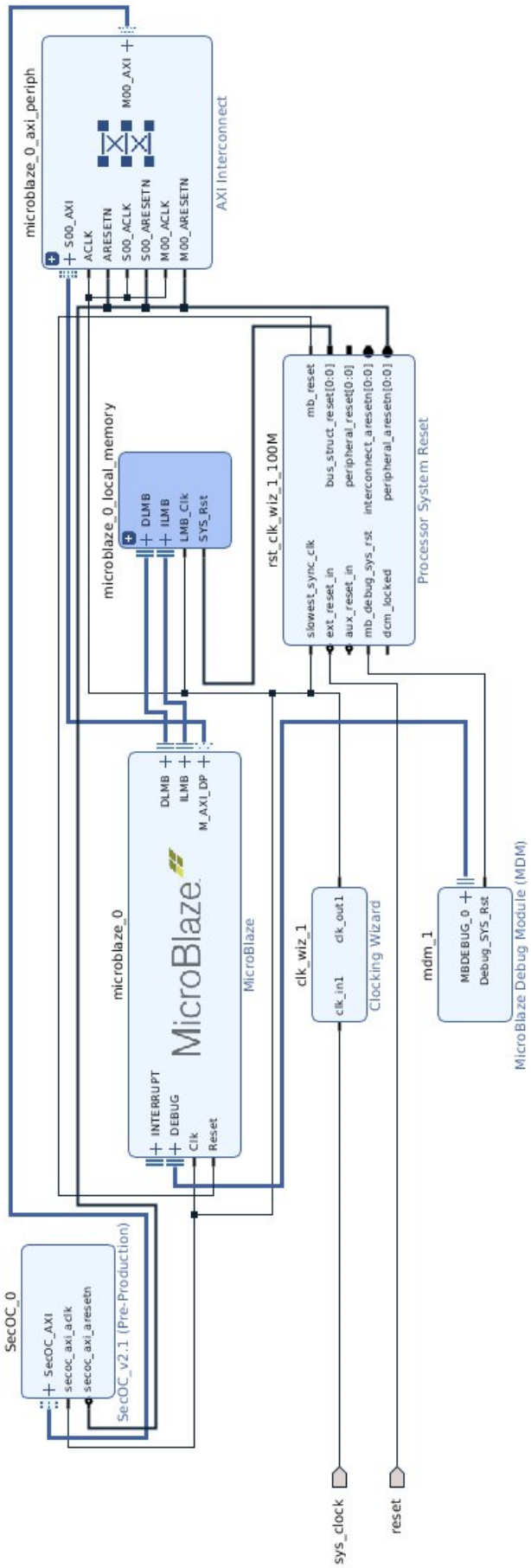

Figure 4.7: Connecting SecOC IP with Microblaze

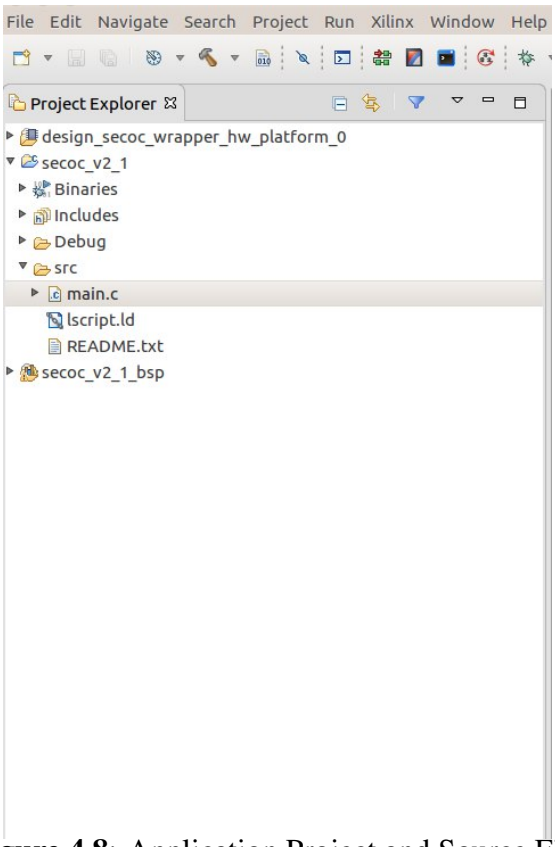

Figure 4.8: Application Project and Source File

5. Conclusion And Future Works

Autonomous vehicle's security must be studied in a sensitive way. This field of study is not only a data security problem, but also a matter of life or death. Autunomous cars, autonomous airplanes and many of the self driving system must have protection against attacks in addition to the their functionality.

This project's goal is constructing a secure automotive CAN network by implementing SecOC module in hardware. For this purpose, algorithm is implemented on FPGA and then connected with a soft processor called Microblaze. Software part is written for microblaze processor. After these steps, hardware and software co-design is completed and tested.

Implementation of MAC Manager is completed with 4 sequential AES block. This method occupies 4 times AES block area. This situation will be improved with adding memory elements to the MAC and just with one AES block, same results will be taken from MAC Manager.

There were no timing constraints in this implementation. Timing constraints will be added after MAC Manager optimisation process is finished.

Standart 8 bit payload CAN is choosen as the protocol that will be transfer the information. Newer CAN concepts like CANFD 64 byte payload or any other protocol that supports more than 8 byte can be chosen for more byte transmition in a single operation. This enables the more data transfer, more bytes of MAC and Freshness value transmission in a single frame.

REFERENCES

[1] **BOSCH**, 1991, "BOSCH CAN Specification".

[2] **ISO**, 2015, "Controller area network (CAN)".

[3] **Instrument, T.**, 2016, "Introduction to the Controller Area Network (CAN)".

[4] **Hussain, S. and Zaidi, S.I.H.** Fast Hardware Implementation of AES-128 Algorithm in Streaming Output Feedback Mode for Real Time Ciphering", journal = "International Journal of Security and its Applications ", volume =.

[5] **Tutorialspoint**, The Advanced Encryption Standard, `https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm`.

[6] **CommonLounge**, The Advanced Encryption Standard (AES) Algorithm, `https://www.commonlounge.com/discussion/e32fdd267aaa4240a4464723bc74d0a5`.

[7] **Kissell, J.**, 2010. Mac® Security Bible, Wiley Publishing, Inc.

[8] **Kuon, I.**, **Tessier, R. and Rose, J.**, 2008. FPGA Architecture: Survey and Challenges.

[9] **Ashenden, P.J.**, 2007. Digital Design (VHDL): An Embedded Systems Approach Using VHDL, Morgan Kaufmann.

[10] **DIGILENT**, Nexys 4 DDR, `https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/start`.

[11] **XILINX**, 2018, "MicroBlaze Processor Reference Guide".

[12] **XILINX**, 2012, "AXI Reference Guide".

[13] **Churiwala, S.**, 2017. Designing with Xilinx® FPGAs, Springer International Publishing.

[14] **Thomas, D.E. and Moorby, P.R.**, 2002. The Verilog® Hardware Description Language, Kluwer Academic Publishers.

[15] **XILINX**, 2020, "Vivado Design Suite User Guide".

[16] **MathWorks**, Design digital FPGA, SoC FPGA, or ASIC hardware, `https://www.mathworks.com/discovery/hardware-design.html`.

[17] **XILINX**, 2017, "Getting Started with Xilinx SDK".

[18] **AUTOSAR**, 2017, "Specification of Module Secure Onboard Communication, Classic Platform".

[19] **AUTOSAR**, 2017, "Specification of Secure Onboard Communication".

[20] **Vector Japan Co., L.**, 2020, "For Beginners AUTOSAR SecOC".

[21] **Sjafrie, H.**, 2019. Introduction to Self-Driving Vehicle Technology.

[22] **Gürgens, S. and Zelle, D.**, 2019. A Hardware Based Solution for Freshness of Secure Onboard Communication in Vehicles.

[23] **Rosenstatter, T.**, **Sandberg, C. and Olovsson, T.**, 2019, Extending AUTOSAR's Counter-based Solution for Freshness of Authenticated Messages in Vehicles.

[24] **Zou, Q.**, **Chan, W.K.**, **Gui, K.C.**, **Chen, Q.**, **Scheibert, K.**, **Heidt, L. and Seow, E.**, 2017, "The Study of Secure CAN Communication for Automotive Applications".

[25] **Seifert, S. and Obermaisser, R.**, 2014, "Secure Automotive Gateway –Secure Communication for Future Cars".

[26] **Siddiqui, A.S.**, **Gui, Y.**, **Plusquellic, J. and Saqib, F.**, 2017. A Secure Communication Framework for ECUs Körper. (U.S.A.), *Annalen der Physik*, **2(3)**, 1307–1313.

[27] **Bozdal, M.**, **Samie, M. and Jennions, I.**, 2018, "A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions".

[28] **Bella, G.**, **Biondi, P.**, **Costantino, G. and Matteucci, I.**, 2019, "TOUCAN: A proTocol tO secUre Controller Area Network".

[29] **XILINX**, 2018, "Creating, Packaging Custom IP Tutorial".

CURRICULUM VITAE

Name Surename: Ömer Demirci

Place and Date of Birth: Ordu -Turkey, 1998

E-Mail: demirciomer98@gmail.com

Education:

- **B.Sc.:** Istanbul Technical University

Professional Experience: 2019 - 2020 : Autonomous Systems and Robotics Developer at Tubitak BTE

Name Surename: M. Enes SOLTEKİN

Place and Date of Birth: Mugla -Turkey, 1997

E-Mail: menessoltekin@gmail.com

Education:

- **B.Sc.:** Istanbul Technical University

Professional Experience: 2019 - 2020 : Autonomous Systems and Robotics Developer at Tubitak BTE