Hardware Based Root of Trust for Electronic Control Units

Yutian Gui
University of North Carolina at Charlotte
Charlotte, USA
ygui@uncc.edu

Ali Shuja Siddiqui
University of North Carolina at Charlotte
Charlotte, USA
asiddiq6@uncc.edu

Fareena Saqib
University of North Carolina at Charlotte
Charlotte, USA
fsaqib@uncc.edu

***Abstract*—Electronic Control Units (ECUs) are enriched devices to control mechanical parts through communication of control signals between them. Vehicles, like other internet of things, are vulnerable to device and network level attacks that are undeniably a safety concern. In this paper, we present a hardware-based framework with Trusted Platform Modules (TPM) enabled secure boot, traffic control for secure communication and mitigating Denial of Service (DoS) attacks by whitelisting, blacklisting and thresholding mechanism. Additionally, at the time of possible compromise, the proposed system provides self-perseverance constructs that are discussed with respect to architecture, implementation and security analysis.**

Keywords—Hardware security; Access control; Traffic control; Electronic control unit; Denial of Service (DoS)

I. INTRODUCTION

Electronic Control Unit (ECU) is an electronic device for data processing, communication and controlling a series of actuators for optimal vehicle operations by automating one or more mechanical or electrical systems. The number of ECUs in a vehicle was few with simple architecture of old cars but has increased tremendously with the addition of new functionalities, such as airbag system, cruise control, and parking distance control to provide safety features and provide advanced driver assistance system. To manage all these ECUs, Controller Area Network (CAN) serial bus system was first introduced by Robert Bosch GmbH in 1986. Benefit from the high efficiency and reliability, CANBus has become an international standard (ISO11898) [1] and is widely applied to not only modern vehicles but also industrial and medical devices.

The ECU and CANBus are vulnerable to malicious attacks. The data transmission in CAN network is normally unprotected and the data frame can be easily monitored and collected. Also, if one of the ECUs is hacked, the attacker then has the ability to send malicious frames to other ECUs to control their behavior. In 2015, a successful demonstration of hacking was shown by taking over the control of Jeep using the vulnerabilities in the telematics and compromising the entertainment system [2]. In the experiment, adversaries could take over charge of its control system and disable the braking system. [3] proves that the current keyless entry systems are insecure. All these works show that the CANBus and ECUs are vulnerable and unsafe.

To protect ECUs from potential attacks, many countermeasures have been proposed. [4] aims at the physical level and demonstrates a novel architecture of ECU based on FPGA platform. By modifying the structure and integrating Steer-By-Wire (SBW), the security and dependability are increased significantly. [5] presents a Security ECU (SECU) module which provides the functionality of authentication by using a hash function to generate MAC address for each node in CAN network, thereby the malicious frame with invalid MAC address will be detected. Similarly, [6] uses Advanced Encryption Standard (AES) to encrypt message frame to increase the difficulty of hacking and proposes a hardware-based key generator for encryption to reduce the risk of attacks direct at encryption algorithm. Hardware-based security framework is proposed in [7] [8] that provides secure authentication in a client-server architecture and encryption based communication for CAN network based on time delay based Physical Unclonable Function (PUF). [9] improved this approach with adding an additional authentication and shared key exchange process and applied it to the CAN Flexible Data-rate (CANFD) frame. [10] shows more details about PUF implementation and security analysis which are used in this hardware-based secure framework.

Current solutions for ECU and CANBus have limitations. Most of existing security solutions focus on the encryption and authentication process but neglect some attacks which aim at network traffic, such as the Denial of Service (DoS) attack. If any ECU in a CAN network is hacked, the attacker can easily occupy the whole bandwidth of CANBus by sending numerous useless frames to other nodes even without knowing the specific encryption and authentication algorithms. As a result, the CAN network will be paralyzed due to the DoS attack.

Even the existed countermeasures improve the security of intra-vehicular communication, one point is ignored. The security of message itself can be ensured by using encryption and authentication, but the communication is still vulnerable to Denial of Service (DoS) attack which aims at network traffic. By inserting and sending numerous worthless message frames to the bus, the attacker can take over the whole bus traffic even

without knowing the key of encryption and authentication mechanism.

In this work, we present a novel solution which provides security against DoS attack, secure boot, traffic control and attack detection for ECUs to defend from the hostile attacks ECUs and CANBus. Our major contributions are summarized as follows:

- We use Trusted Platform Module (TPM) to provide a secure key generation and storage cryptosystem for ECUs. To the best of our knowledge, we are the first to use TPM to improve the security of ECUs.
- Based on TPM, we propose two scenarios to realize secure boot. The proposed framework ensures that no malware can tamper the ECU during the boot process, computation and communication.
- We design a secure communication model which consists of a whitelisting module with thresholding on sender-side and a blacklisting on receiver-side to realize access and traffic control. These modules block the compromised data on both of sender-side and receiver-side and reduce the risk of DoS attack.
- We design a new protocol for secure communication and apply it over CANFD without changing any existing structure of message frame. The protocol works based on a co-work of several secure solutions, including whitelisting, thresholding, blacklisting, and encryption, to implement access control, traffic control, attack detection and secure authentication.

II. Related Work

A. Controller Area Network (CAN) Bus

Electronic Control Unit (ECU) provides the ability to control different electrical and mechanical subsystems in the vehicle. Like other embedded systems, ECU consists of a microcontroller which is programmed with exact needs, Random Access Memory (RAM) and Read Only Memory (ROM) for data storage, some input and output ports and an Analog/Digital (A/D) converter for reading outputs from sensors basically. To meet different needs, some ECUs also provide specific components with customized functionalities.

The main responsibility of CANbus is to support communication between different ECUs and manage these ECUs with high speed and low cost. Since there is no host node in CANbus, all the communication is achieved by exchanging message frames. In CANbus, the data is divided into many equal-sized message frames and sent in sequence. Different frames have different priory depending on the importance of each ECU. CANbus provides several standards of message frames for automobile manufacturers to satisfy different demands, including standard frame, extended frame, remote frame, etc. Different standards have different lengths, but the structures are similar.

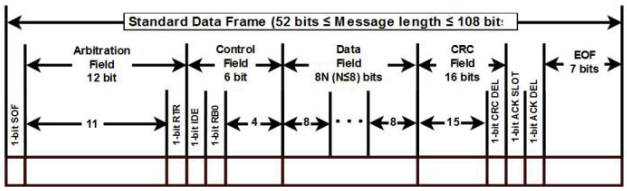

Fig. 1. Standard CANbus message frame

Figure 1 shows the structure of the standard frame. The length of the standard CANbus frame can vary from 52 bits to 108 bits, depending on the length of data to be sent. CAN frame has a 1-bit Start of Frame and a 12-bit arbitration field. The arbitration field contains an 11-bit identifier (ID), and this identifier is normally used for representing the target address. Next is the 6-bit control field, which consists of two reserved dominant bits (IDE and RB0) and 4-bit DLC which declares the number of data bytes contained in the frame. The data field consists of 1 to 8 bytes. The CRC field is used for cyclic redundancy check, 2-bits acknowledge for ACK check and 7-bit End of Frame (EoF) field [11]. By adding two additional bits in the control field after IDE and RB0 bit, the CAN frame can be transformed to a CAN Flexible Data-rate (CANFD) frame. The first bit named Flexible Data-rate Frame (FDF) is used for indicating if the frame is a CANFD frame or not. The second bit named Bit Rate Switch (BRS) bit is used for switching the data field to flexible data mode. The CANFD allows transmission of up to 64 bytes of data in a single frame which is equal to eight standard or extended CAN frames. Meanwhile, the transmission speed of CANFD is higher than standard CAN frames.

B. Trusted Platform Module (TPM)

As a global standard [12] for the secure cryptoprocessor, TPM is a specialized chip which is designed to enhance the security of computing devices. TPM provides the functionality of Root of Trust (RoT) realized by a series of operations, including key generation, key storage, and authentication.

Specifically, each TPM chip contains asymmetric keys embedded in the TPM, known as endorsement key (EK). When first time a user activates a new device with built-in TPM chip, a root key is generated based on endorsement key pair which is referred as attestation key. The root key is unchangeable because the EK is burned by the manufacturer, and never leaves the TPM. The derived attestation identity key (AIK) is used in cryptographic operations such as for encryption and signing frames with hash functions.

C. Secure Boot

The core concept of secure boot is to check the validity and integrity of operating system and software during the boot process. This can be achieved with the Trusted Platform Module (TPM) and Unified Extensible Firmware Interface (UEFI) for the over the air firmware updates. Specifically, UEFI is a firmware interface which plays the role of an interpreter between the operating system and platform firmware. Comparing to traditional basic input/output system, UEFI provides a better compatibility and a higher flexibility for the hardware design and security requirement.

D. Whitelisting and Blacklisting

Generally, a whitelist is a list of objects which can access to a particular system. Like a filter, the whitelist helps users to permit any content they are authorized to receive and block any unwished or malicious data. The term of whitelisting normally represents the process of building a whitelist for any systems in need. On the contrary, the blacklisting means the process of creating and using a blacklist for specific demands

The access control can be realized by whitelisting and blacklisting in the communication fields. The basic idea is to label each frame by using MAC address or ID numbers and use these unique labels to build whitelists and blacklists depending on the exact requirement. [13] Provides a novel idea that integrates the functionality of whitelisting into traditional network monitor as a security measure for industrial control systems. [14] Presents a multilevel anti-SPAM mechanism which is based on blacklisting.

III. PROPOSED ARCHITECTURE

We propose a secure framework and a hardware mechanism to mitigate denial of service attack. The architecture of proposed design is shown in Figure 2. The TPM module is used for secure boot, data encryption/decryption and message signing (optional) to reduce the risk of eavesdropping. Furthermore, to prevent compromised node from sending any malicious frame to other nodes, we add a blacklisting module and a whitelisting module with thresholding functionality inside each ECU to limit transmission rate and filter out malicious frames on both of sender-side and receiver-side. In each ECU module, we design an error detection and logging mechanism to check the validity of all the denied message frames during diagnostics.

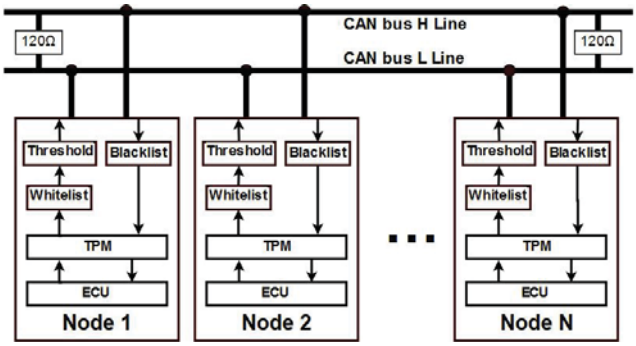

Fig. 2. Proposed framework

Steps of the proposed protocol are discussed below:

1. The whitelisting with thresholding and blacklisting for each node in CANbus are configured in a trustworthy facility. The security of environment is realized by secure boot scenario which is based on TPM. Once the configuration is complete, the interfaces are disabled, and the device is deployed in the field.
2. Each device securely generates public and private keys. The public keys are shared among the whitelisted nodes and the whitelisted and blacklisted identifiers (IDs) are stored in each node's blacklist and whitelist. The keys, including public keys from other trusted nodes, are stored inside a Non-volatile Memory (NVM) inside the TPM of each ECU node. This process is predefined by the manufacturer.
3. Once deployment is completed, the message is encrypted or signed (optional) by the TPM. The encrypted message is encapsulated in an extend CANFD frame with the IDs of target/destination in arbitration field and sender in the data field as a flag for access control.
4. The encrypted and signed (optional) frame is sent to the whitelisting module for checking and verification by whitelisting to allow the further propagation of the message over CANbus. If the target ID of the message frame is on the whitelist, then the frame can be regarded as secure and sent after verifying the frames sent by the node has not exceeded the threshold value. Otherwise, it is considered as a compromised frame, that is blocked and logged.
5. On receiver-side, the recipient ECU compares the sender ID in data filed of each coming frame. If the sender ID matches any IDs on the blacklist, it will be considered as a malicious message, logged and blocked, otherwise the node will accept the frame, lookup the ID-Key table to get the key of the sender and use the shared key to verify and decrypt the message frame.

A. Secure Communication Model

To mitigate malicious intrusion and Dos attack, we implement access control and traffic control based on the co-work of whitelisting, blacklisting, thresholding and data authentication among different nodes in CAN network.

1) Message Format for Secure Communication

Comparing to standard CAN frames, CANFD can carry up to 64 bytes (512 bits) of data in one frame with higher transmission speed. In this paper, we use extended CANFD to carry messages between different ECU nodes because the identifier in arbitration field of extended CANFD is longer than the standard frame, therefore, the CANbus can connect more ECUs.

In CAN message frame, the arbitration field is used for storing the ID of the target and does not store the sender information, therefore, making the mutual authentication impossible. In this work, we store the sender ID in the data field to enable the bidirectional access control.

The proposed frame of extended CANFD is shown in Figure 3. All the messages are encrypted/decrypted with AES-128 standard using TPM. The 29-bit target ID is stored in the arbitration field and the sender ID is stored in the first 32 bits (29 bits in 32-bit HEX) of the 512-bit data field. The rest of bits (480 bits) in data filed is used for storing encrypted data and/or signature (optional). The receiver decrypts the packet with its own private key, identifies the sender, and uses the public key of the sender to validate the signature and confirms

the data has not been changed during the transmission (optional).

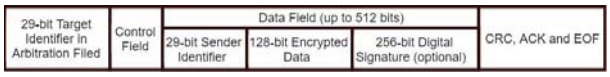

Fig. 3. Extended CANFD frame

2) Sending Process

As we discussed before, all the EUCs are connected to same CANbus in an intra-car communication network, but each ECU node only communicates with a limit number of other ECUs depending on their own usages and configuration. If node A is dependent on the data or commands from node B, then the process of message exchanging is required between these two nodes, and node A can be considered as a whitelisted node to node B. In another case, if there is no data communication between node A and node B, then they can be considered as blacklisted nodes to each other. The relationship between each ECU is defined by the manufacturer depending on the actual condition of each vehicle. In this work, we assume that the whitelist and blacklist in each node are pre-configured in each node before it leaves the factory.

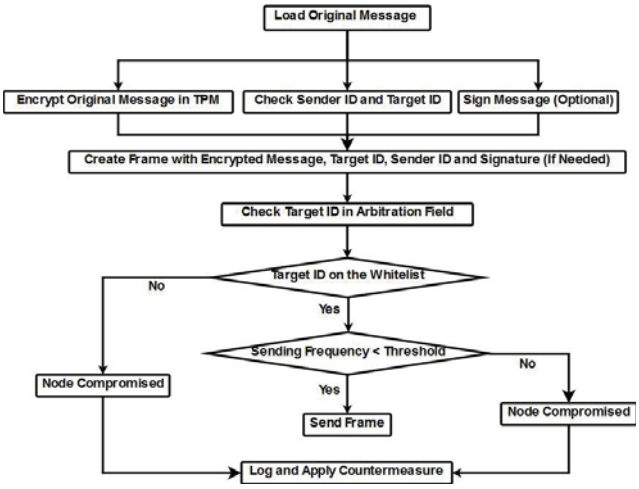

Fig. 4. Flowchart of Sending

Since the functionality and relationship of each ECU are constant in a specific model of vehicle, we can easily make a list for each ECU which contains the ID of all the whitelisted nodes. If the ID of node A is on node B's whitelist, node A is safe to node B and node B can send the frame to node A. Otherwise, the frame will be blocked, then the countermeasure will be executed to cope with the attack.

In case of one node is compromised and the attacker wants to take over the whole bus traffic and block other nodes to communicate with each other by sending numerous junk message frames, a list of thresholds for all whitelisted nodes is assigned in each node by the trusted facility at the time of configuration. After a frame passes the verification by whitelisting, the thresholding module checks the maximum sending rate by using the target ID in arbitration field. For example, let's assume that node B is a whitelisted node for node A, and the threshold for node B in node A is "n". Therefore, node A is only allowed to send "n" frames per window to whitelisted node B. If node A wants to send frames to node B with a frequency which is higher than the threshold, the rest of frames to be sent will be blocked and logged. This mechanism mitigates the DoS attack and helps in identifying the source of the possible compromise.

The specific flowchart of proposed sending process is shown in Figure 4. The original message is firstly encrypted in the TPM module. Then the frame is generated with the receiver ID in arbitration field, the sender ID and encrypted message in the data field. The whitelisting module and thresholding module check the arbitration field and threshold value to validate the target ECU is legitimate and the number of frames sent has not been exceeded for the given window. If the sending frequency is lower than the sending threshold, then the node will send the frame. Otherwise, if the frame fails in whitelisting check or thresholding, the node will be regarded as a compromised node and the incident will be reported in the log file for later diagnostics.

3) Receiving Process

Figure 5 shows the process of receiving. Similar to the whitelist, the blacklist is also pre-configured by the manufacturer during production. On the receiving end, the validity of the received frame is checked by comparing the sender ID with IDs on the blacklist. If the sender ID is not on the blacklist, then it will be considered as a valid frame, otherwise, this error will be logged, and the frame will be blocked. If the frame contains a digital signature, the receiver looks up the ID-Key table where stores the IDs and public keys of other nodes by using the sender ID as a flag and validates the sender. The message is decrypted with receiver's private key. If the final authentication succeeds, the frame will be accepted otherwise the receiver blocks this message as a compromised frame and reports the sender ID for later diagnostics.

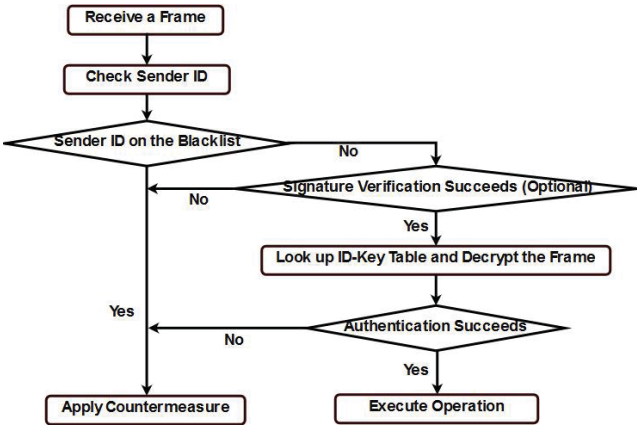

Fig. 5. Flowchart of receiving

B. Secure Boot

To protect the security of ECU and detect the malicious intrusion at the time of boot, we propose two scenarios.

1) Scenario 1: Code Execution from Read Only Memory.

Traditionally, the boot code is placed in the Read Only Memory (ROM) (e.g. BIOS). This code is copied into the RAM at the time of execution. This technique is known as "shadowing". However, with the advancement in high-speed memory technologies and interfaces, it is now possible to execute code from non-volatile memories without the need to copy the code onto to RAM. It is referred to as "eXecute In Place" (XIP) [15]. It is achievable using newer interfaces such as Quad Serial Peripheral Interface (SPI). Quad SPI allows a higher throughput than the classical SPI. MCUs and the memory can be configured to use XIP mode. Depending on the type of memories and the frequencies of memory accesses, the MCU and the ROM can both be configured to a Random Access or Sequential (Send Instruction Only Once) modes [16].

In XIP all executable code will exist in the ROM, where the ROM is mapped to the first executable address of the attached MCU. For example, at address 0x0 for a CPU. Depending on the size of the memory, higher address spaces can be mapped to a RAM. The RAM can be used for storing dynamic structures. With this scheme, it can be assumed that the Program Counter (PC) of the CPU can only remain within the ROM address space. With this execution model, any memory access for code outside of this region can be considered as an illegal access. A hardware monitor is used to check the current value of the PC register. This monitor reads the value of PC at each cycle and it screens each packet and confirms that it is within the ROM address space. With any address access outside this region, the MCU is considered compromised and is logged into the system.

2) Scenario 2: Hardware-Based/Assisted Core Root of Trust Measurement.

In case the executable code is on RAM, it can be modified at the boot time. To avoid the attack during the boot process, the TPM maintains the golden measurements of the trusted executable code. We propose a TPM based application code sealing mechanism to avoid firmware update/modify attacks on automotive. In this approach, we employ "Core Root of Trust Measurement" (CRTM) [17] for implementing code sealing before the application code can be executed on the processor.

A TPM provides a set of registers known as Platform Configuration Registers (PCRs). PCRs hold an Hash-based Message Authentication Code (HMAC) digest of the operations performed on a TPM. These register values are changed only using extend operations. Extend uses an old result, for example, HMAC from a previous operation, concatenates the new result to it and then computes the hash value again. This value is saved to the assigned PCR. By use of this chaining process, the resultant is used as a verification checksum.

The proposed scheme has two phases towards providing CRTM. The first phase is sealing the application code. This is performed in a trusted execution environment. A pair of public and private keys are generated on the TPM. The public key is exported whereas the private key is stored inside the TPM. Using the pair of keys, we seal the application code. This process requires passing the TPM sequential chunks of the application code, the keys generated and the policy for the PCRs chosen for the sealing process. This process returns the encrypted sealed application code and the computed digest values. All the keys and the digest values are stored in the persistent area of the TPM. These areas are then made immutable with TPM authorization policies. Figure 6 describes the measurable secure process in the proposed framework.

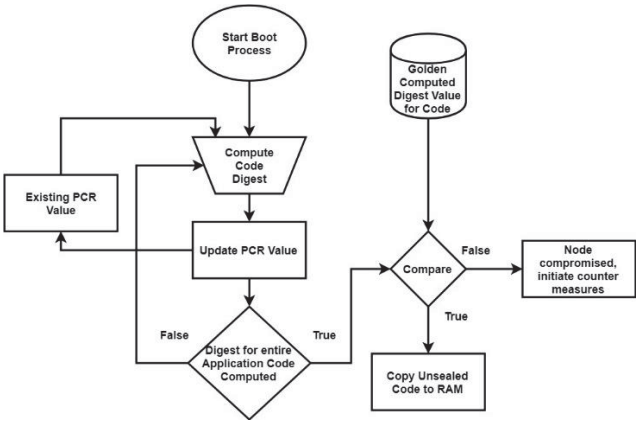

Fig. 6. On the field Code Unsealing.

IV. Experimental Setup and Configuration

The test bed includes arm cortex A53 on Raspberry PI 3 [18], Infineon OPTIGA™ TPM 2.0 SLB9670 [19] and SKPang PiCAN 2 duo Shield. The components are connected with SPI interface and share the communication and the clock signals. The Chip Select signal is used to select between the TPM and the CANBus shield. In the setup, implementation includes Raspbian with Linux Kernel Version 4.4, and Intel TPM software stack TPM2-TSS to establish an interface with the TPM device. TSS is provided by Intel Corporation as a user space C language library stack for implementing a wrapper around the TPM2 commands as provided in the TPM2 specifications. TPM2-Tools are a collection of tools that are based on the system API provided by the TSS. In our experiments, we used the TPM2-Tools to generate set of keys. Additionally, we calculated HMAC of blocks of data, performed encryption and decryption operations.

The PiCAN shield is attached to the Raspberry Pi3 and programmed with the vendor provided library and tools to construct and send messages over a CANBus network. We used the TPM device to generate encrypted messages using AES-128 and then transmit them over the CAN network. Since the standard CAN provides a maximum of 64 bits of data, we need two CAN messages per encrypted message to send the data. This limitation is only because of the component used in the experiment does not have CANFD, otherwise, CANFD will allow the message to be sent in one frame. Figure 7 shows the implementation of the proposed framework.

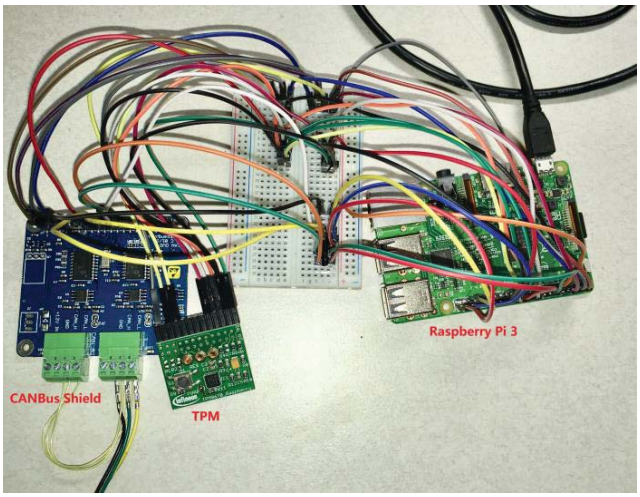

Fig. 7. TPM and CANbus Setup

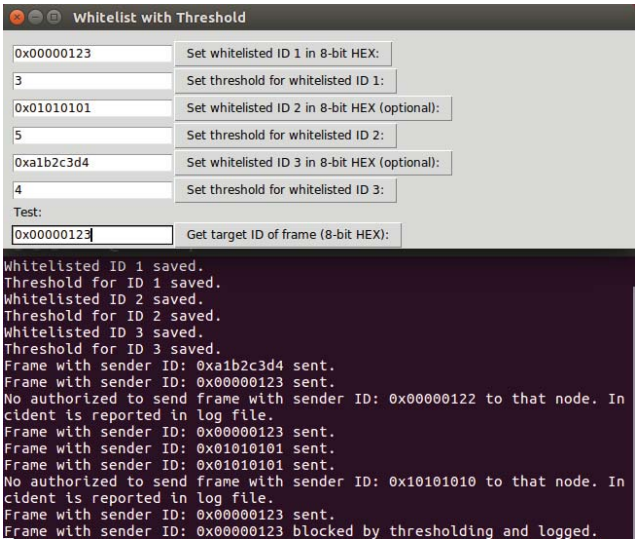

Fig. 8. Whitelisting and Thresholding

Figure 8 shows the tool of whitelisting with thresholding module for access control realization and the result of a running example. The manufacturer can set the whitelisted target IDs (0x00000123, 0x01010101, and 0xa1b2c3d4 in the example) with the corresponding thresholds (3, 5 and 4 respectively) that is the maximum sending rate. Each time a new frame is sent, the sender's whitelisting module compares the target ID in the arbitration field of CANFD frame with all the whitelisted IDs and forwards the frame only in case the ID is part of the whitelisting database. The target ID is compared with all the registered and authorized target nodes/ECU IDs as shown in Figure 8. The second thresholding feature keeps monitoring the sending frequency. Once the sending threshold is reached, thresholding module detects and blocks the remaining frames until next sending window starts.

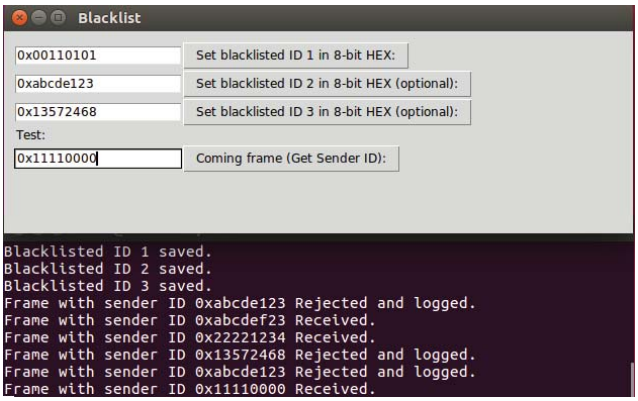

Fig. 9. Blacklisting

The tool and a running example of blacklisting module are shown in Figure 9. The blacklist in each node is configured by the manufacturer (0x00110101, 0xabcde123 and 0x13572468 in the example). Each time a new frame is received, the blacklisting module compares the sender ID which is included in the data field of the CANFD frame with all the blacklisted IDs. If it matches any blacklisted ID, the frame is rejected, otherwise, it is considered as a secure frame and received for decryption and verification (optional).

V. Security Analysis

In this work, the proposed security enhanced framework improves the security of ECUs by providing secure boot process with TPM enabled measurable boot, encrypted communication and mitigating the denial of service attacks. The boot process enables secure loading of executable code and compares the digest of the application code with the golden digest stored on the TPM at the time of trusted configuration. The framework assumes a secure facility configures the golden reference of code digest, the whitelist programming and secure exchange of public keys of the connected nodes. The keys are stored in a tamper-resistant permanent storage in the TPM and thus the keys are secure from the physical and invasive attacks. In the proposed framework, the frames are encrypted and only the destination node can decrypt it with its private key and thus the man in the middle attack is not possible. The TPM module provides a full protection for the message frame. In each node, the message is encrypted by AES-128 in TPM by using its own private key. Furthermore, the proposed design provides the functionality of signing off as an option to realize bidirectional authentication and provide data integrity.

On the sender side, the whitelisting mechanism checks the target ID in frame's arbitration field before sending, so the frame can be only sent to whitelisted nodes. If the target ID is not on the whitelist, the frame can be considered as a compromised frame and it will be blocked inside the ECU. This activity will be logged for the future design of countermeasures. In case a compromised node passes the check by whitelisting and wants to perform DoS attack on the CAN network, it will be limited by the thresholding mechanism. Thresholding limits the maximum sending frequency in each node. The frames that exceed the maximum threshold are dropped and thus mitigate the DoS attack. The logging

mechanism further enhances the design of future vulnerabilities in the framework. Meanwhile, the blacklist module blocks any frame with a blacklisted sender ID and logs the incident on the receiver side to realize bidirectional access control.

VI. Conclusion

In this paper, we present a secure framework for vehicular network communication using CANBus. The proposed framework implements secure boot based on TPM and prevents code update or malicious modifications during the boot process. The framework provides secure communication, access, and traffic control by whitelisting, blacklisting and thresholding mechanism proposed to mitigate denial of service attacks. The paper discusses the implementation details and hardware components used to establish the framework. The security analysis section evaluates the security of the proposed scheme with respect to threats of man in the middle, compromised device, and denial of service (DoS).

Acknowledgment

This research has been sponsored by the National Science Foundation under grant no. 1566530 and 1623299.

References

[1] Road vehicles -- Controller area network (CAN). 2015. International Organization for Standardization. Retrieved Nov 14, 2017 from https://www.iso.org/standard/63648.html

[2] C.Miller and C.Valasek. 2015. Remote Exploitation of an Unaltered Passenger Vehicle. In *Black Hat 2015, Las Vegas, NV, USA, August 2015*.

[3] F.Garcia, D.Oswald, and T.Kasper. 2016. Lock It and Still Lose It-On the (In) Security of Automotive Remote Keyless Entry Systems. In *USENIX Security 2016, Austin, TX, USA, August 2016*.

[4] Bikash Poudel and Arslan Munir. 2017. Design and evaluation of a novel ECU architecture for secure and dependable automotive CPS. In *Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual*, *Las Vegas, NV, USA, January 2017*. 841–847. DOI: 10.1109/CCNC.2017.7983243

[5] Eric Wang, William Xu, Suhas Sastry, Songsong Liu and Kai Zeng. 2017. Hardware Module-Based Message Authentication in Intra-Vehicle Networks. In Cyber-Physical Systems (ICCPS), 2017 ACM/IEEE 8th International Conference *on*, *Pittsburgh, PA, USA, April 2017*. 207–216.

[6] Adirek Maruaisap and Pinit Kumhom. A hardware-based security scheme for in-vehicle CAN. 2017. In *Computer Science and Engineering Conference (ICSEC), 2016 International, Chiang Mai, Thailand, December 2016*. 1-5. DOI: 10.1109/ICSEC.2016.7859891.

[7] Ali Shuja Siddiqui, Yutian Gui, Jim Plusquellic and Fareena Saqib. 2017. Poster: Hardware based security enhanced framework for automotives. In *Vehicular Networking Conference (VNC), 2016 IEEE, Columbus, OH, USA, December 2016*. 1-2. DOI: 10.1109/VNC.2016.7835977

[8] Ali Shuja Siddiqui, Yutian Gui, Jim Plusquellic and Fareena Saqib. 2017. Secure communication over CANbus. In *Circuits and Systems (MWSCAS), 2017 IEEE 60th International Midwest Symposium on, Boston, MA, USA, August 2017*. 1264-1267. DOI: 10.1109/MWSCAS.2017.8053160

[9] Ali Shuja Siddiqui, Chia-Che Lee, Wenjie Che, Jim Plusquellic and Fareena Saqib. 2017. Secure Intra-Vehicular Communication over CANFD. In *Hardware-Oriented Security and Trust (AsianHOST), IEEE Asian 2017, Beijing, China, October 2017*.

[10] Ali Shuja Siddiqui, Yutian Gui, Jim Plusquellic and Fareena Saqib. 2017. A Secure Communication Framework for ECUs. *Advances in Science, Technology and Engineering Systems Journal.* Volume 2, Issue 3, 1307-1313. DOI: 10.25046/aj0203165

[11] MCP2515 Stand-Alone CAN Controller with SPI Interface. 2016. Microchip. Retrieved Nov 16, 2018, from http://ww1.microchip.com/downloads/en/DeviceDoc/20001801H.pdf

[12] ISO/IEC 11889-1:2015. 2015. Information technology -- Trusted platform module library. Retrieved January 13, 2018, from https://www.iso.org/standard/66510.html

[13] Koichi Shimizu, Teruyoshi Yamaguchi, Tsunato Nakai, Takeshi Ueda, Nobuhiro Kobayashi and Benoît BoyerA. 2017. A Trusted Approach to Design a Network Monitor. In *Formal Methods in Software Engineering (FormaliSE), 2017 IEEE/ACM 5th International FME Workshop on. Buenos Aires, Argentina, May 2017*. 17-23. DOI: 10.1109/FormaliSE.2017.3

[14] Tomas Sochor and Radim Farana. 2013. Improving Efficiency of E-mail Communication via SPAM Elimination Using Blacklisting. In *2013 21st Telecommunications Forum Telfor (TELFOR). Belgrade, Serbia, January 2014.* 924-927. DOI: 10.1109/TELFOR.2013.6716382

[15] Aurelien Francillon, Quan Nguyen, Kasper B. Rasmussen and Gene Tsudik. 2014. A minimalist approach to remote attestation. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), Dresden, Germany, March 2014*. 1-6. DOI: 10.7873/DATE.2014.257

[16] Quad-SPI (QSPI) interface on STM32 microcontrollers. 2016. STMicroelectronics. Retrieved January 5, 2018, from www.st.com/resource/en/application_note/dm00227538.pdf

[17] Stefan Berger, Kenneth Goldman, Dimitrios Pendarakis, David Safford, Enriquillo Valdez and Mimi Zohar. 2015. Scalable Attestation: A Step toward Secure and Trusted Clouds. In *Cloud Engineering (IC2E), 2015 IEEE International Conference on, Tempe, AZ, USA, March 2015*. 185 – 194. 10.1109/IC2E.2015.32

[18] Raspberry Pi 3 Model B - Raspberry Pi. 2017. Raspberry Pi. Retrieved January 02, 2018 from https://www.raspberrypi.org/products/raspberry-pi-3-model-b.

[19] SLB 9670 VQ1.2 FW6.40 - Infineon Technologies. 2017. Infineon. Retrieved January 02, 2018 from https://www.infineon.com/cms/en/product/security-smart-card-solutions/optiga-embedded-security-solutions/optiga-tpm/slb-9670-vq1.2-fw6.40