Secure Communication over CANBus

Ali Shuja Siddiqui, Yutian Gui Jim Plusquellic[+] and Fareena Saqib
University of North Carolina Charlotte, USA, University of New Mexico[+], USA
asiddiqui@uncc.edu, ygui@uncc.edu, jimp@ece.unm.edu, fsaqib@uncc.edu

Abstract — **In this paper, we propose hardware based secure and trusted communication over CAN bus in the intra vehicle network connecting electronic Control Units (ECUs). CAN bus is an insecure communication channel, connecting resource constraint devices that have limited resources to devote for data security and real-time requirements to meet the safety critical design specifications. We propose a hardware based secure and trusted framework that implements lightweight PUF based mutual authentication and secure encryption over the insecure communication channel. The paper discusses the framework design and implementation details along with the resource utilization and performance analysis of the proposed system.**

Keywords— Controller area network; keyless fobs; access control; asymmetric encryption; physical unclonable functions PUF; hardware security

I. Introduction

Controller Area Network (CAN) bus is a low cost device network that is a standard for intra vehicle communication. CANBus allows the connected electronic control units (ECUs) to communicate in a real time constraint environment [1]. The CAN bus by design is not secure, and the behavior of connected devices can be manipulated and compromised. Several attacks and hacks on the automotive electronics have been reported in recent years, such as [2], a group of researchers hacked a Jeep and were able to remotely control the accelerator and braking system. This attack demonstrated that currently automobiles are insecure and therefore unsafe. [3] discusses the security vulnerabilities of keyless entry systems that rely on a few global master keys. Such security mechanism can be compromised by reverse engineering and cloning the keys from ECU to have unauthorized access of a vehicle, as seen in the most recent key fob hacks to unlock car doors.

Traditional cryptographic primitives are computation intensive and rely on secrecy of shared or session keys. These techniques are not suitable for embedded devices with resource constraints. Existing hardware security modules (HSM) based security solutions are costly and require additional physical area and power. We propose a hardware based authentication framework using strong delay based physical unclonable function (PUF) for enhanced security.

Section II covers the CANBus protocol and the frame format, and section III details the vulnerabilities of CANBus and demonstrate the attacks. In section IV, the design of proposed framework is discussed and section V discusses the implementation and resource utilization. The Security analysis is discussed in section VI.

II. The CAN Bus Protocol

The CAN bus, known as ISO 11898 architecture is a multi-master bus that connects embedded components, I/O interfaces and gateways to communicate with the external world. The components of CAN include a controller and transceiver to send and receive information between subsystems with object and transfer layers for message filtering and status handling. CAN provides arbitration, fault detection but does not implement security in any layers of abstraction provided in CAN protocol. The communication is done using a differential pair CAN high and CAN low signals, and can be configured on standard data rates of 125kbps, 500 kbps and 1Mbps. In the more recent standard for CAN Bus, named *CAN Flexible Data rate*, the CAN frame allows transfers of up to 64 bytes at higher speeds.

SOF	Arbitration	Control	Data	CRC Field	ACK	End of Frame
1 bit	12 bits	6 bits	0 – 64 bits	16 bits	2 bits	7 bits

Fig. 1. Standard CAN Bus Frame.

Figure 1 describes the standard data frame of CAN bus. In the frame, SOF bit denotes the start of frame, 11 bits store arbitration / destination ID, the data field is of 64 bits and CRC field is 15 bits to hold the checksum of the entire packet. The length of the standard CAN frame is 108 bits whereas 131 for an extended CAN frame. The physical medium CAN uses non return to zero (NRZ) encoding.

III. Threat Models

This section presents security vulnerabilities of CAN bus. The test bed is composed of Raspberry Pi 2, SK Pang PiCAN Duo CANBus Shield, Arduino UNO and Sparkfun CAN-Bus Shield. In the test bed configuration, CAN0 and CAN1 are the legitimate nodes and CAN2 is malicious node, as shown in Fig. 2.

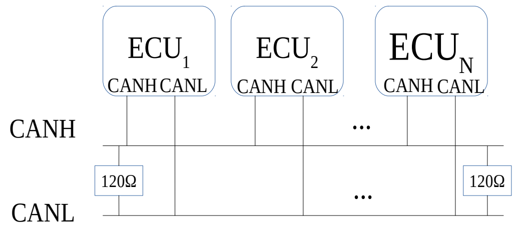

Fig. 2. Experiment Setup.

The threats of CANBus communication include eavesdropping, spoofing, and unauthorized access through telematics attack [4] and are discussed in following sub sections.

A. *Eavesdropping*

A device physically connected to the CAN bus can access all the network data. With the legitimate node CAN0 communicating with CAN1 over CANBus, the adversary node CAN2 once physically or remotely connected to the network can listen to the communication without requiring any additional effort. To mitigate eavesdropping, the information passing on the bus needs to be securely encrypted. In [5], the authors present a physical security solution for bus communication for an alarm system that uses CAN bus for communication between nodes. The solution requires 3DES[6] for encryption and an intermediary node to control the transfers and monitor the network. The added area, power requirements and slower performance of the software based encryption and decryption processes make the solution impractical.

B. *Stealing Identifiers*

The frame consists of the destination identifier and not source identifier. All devices probe the ID field of each message received and based on the ID, deduce if the message was meant for them or not. A malicious device can sniff the network to read the messages and find the ID of the victim device. The malicious device can then send crafted messages to its victim, based on its accepted protocol.

A layer for authentication and access control is needed to provide privacy and integrity of the sender and the data. [7] Proposes a key-hashed message authentication code (HMAC) to hash the messages. Such a technique is not viable for resource constraint devices. Secondly the use of global keys makes the system vulnerable to more attacks. In [8], the solution provides authentication mechanism and encryption of data on the bus but it requires about three times more CPU cycles than the cycles required for normal operation of a CAN node.

C. *External Unauthorized Access using Keyless Fobs*

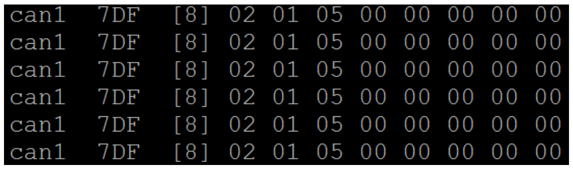
```
can1  7DF   [8]  02 01 05 00 00 00 00 00
can1  7DF   [8]  02 01 05 00 00 00 00 00
can1  7DF   [8]  02 01 05 00 00 00 00 00
can1  7DF   [8]  02 01 05 00 00 00 00 00
can1  7DF   [8]  02 01 05 00 00 00 00 00
can1  7DF   [8]  02 01 05 00 00 00 00 00
```

Fig. 3. Screenshot of CAN2 flooding the network.

In a recent work[3], researchers were able to get unauthorized access to a vehicle by first extracting the contents from the memory block present on a key fob, reverse engineering the source code of the encryption engine and by extracting the symmetric keys and were able to get remote access to the vehicle.

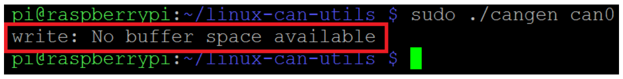
```
pi@raspberrypi:~/linux-can-utils $ sudo ./cangen can0
write: No buffer space available
pi@raspberrypi:~/linux-can-utils $
```

Fig. 4. Screenshot demonstrating buffer overflow ocurring at CAN0.

D. *Denial of Service (DoS)*

Denial of Service attacks makes the resources or services unreachable; in our case the CAN communication channel is made inaccessible by the malicious node CAN2 by broadcasting random messages on the bus without pause, while CAN0 and CAN1 were exchanging information. With the vendor provided code, the legitimate nodes stopped operation and notified about receiving buffer overflow, as shown in Fig. 3 and 4.

IV. Hardware Based Encryption and Authentication Framework

We propose a hardware enriched electronic control unit design for secure nodes. The security framework is based on client server architecture. The server on the network is responsible for handling all clients that are connected to the network. Each client is registered in a trusted environment, referred as secure registration and are authenticated with the server before the devices can initiate communication with the other nodes on a network. The server is assumed to be secure and trusted. Registration is performed during manufacturing, assembly of vehicle components and with the trusted diagnostic centers. Figure 5, demonstrates the overall framework of enchanted secure ECU design. Following subsections describes each step in detail.

A. *Physical Unclonalbe Function Block*

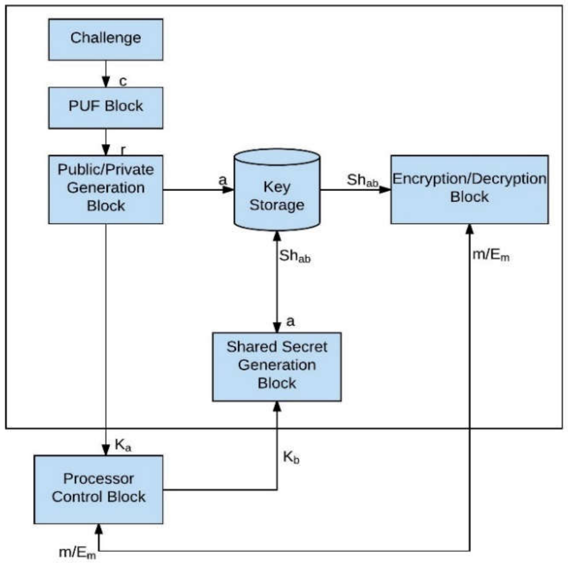

Fig. 3. Secure block implemented at each client node.

Physical unclonable function (PUF) is a physical layer cryptographic primitive used in hardware security and privacy protocols. These are embedded structures that utilize inherent manufacturing process variations to extract unique but reproducible secrets [9][11]. PUFs are based on a challenge-response pair (CRP) mechanism. The challenge for a PUF is defined as a digital input, usually in the form of a bitstring of '0's and '1's. The output of a PUF is also digital but for most PUFs, this requires an on-chip mechanism to convert the small analog variations leveraged by the PUF to be digitized.

During the registration process, each registered ECU is challenged with seed and configuration parameters. Each ECU produces a unique reproducible response/key that serves as private key. This response along with the configuration parameters is used to generate public private pair. Helper data is stored on the NVM of ECU to regenerate same response for the given challenge and public key is communicated to the secure server, where it is stored. Furthermore PUF based authentication is implemented within vehicle network[10]. All client nodes (or devices) are enrolled at the server before they can be used. The registration process is summarized in Table I.

TABLE I. REGISTRATION OF NODES IN A TRUSTED ENVIRONMENT

```
Algorithm: Enrollment process of client nodes in a trusted environment.
Input: Challenge c, configuration parameters for the ECC curve.
Output: Public key $K_a$ of ECU.
1: At each client node, input a challenge $c$ for the PUF Block.
2: PUF block computes a response $r$.
3: Generate a public private key pair (a, $K_a$) using PUF response r and configuration parameters.
4: $K_a$ for each ECU is stored communicated to server, where it is stored in a public database accessible to every node on the bus.
5: Public key of the server, $K_s$ is stored in non-volatile memory at each client node for later access.
```

B. Private/Public Key Generation Block

Elliptic curve Diffie-Hellman based asymmetric encryption and key exchange engine is implemented in hardware. Elliptic Curve Cryptography (ECC) is suitable for the resource constraint devices providing same strength of security as RSA. ECC algorithms are implemented in Galois Field(GF). National Institute of Standards and Technology (NIST) provides a set of recommended elliptic curves which have been computed to be secure[11]. This block is used to generate public and private keys for a node. It is used in the one-time registration process (as described in the previous subsection). The public and private keys are generated on each ECU during initialization process that is every time the vehicle starts (turning on of the ignition).

The authentication algorithm is formulated in Table II. This is a timed process. If any node fails to register itself in the time allocated, then it is blacklisted for communication for the entirety of that session. Private keys for all nodes are generated at run time and are not stored to ensure security. No private keys leave the node and are erased at the end of session.

TABLE II. ALGORITHM FOR INITIALIZATION PERFORMED AT THE BEGINNING OF EACH SESSION.

```
Algorithm: Initialization process.
Input: None.
Output: None.
1: Each ECU generates PUF response r using stored challenge c, and helper data.
2: With $r$ as the input to the Public / Private Key Generator Block, generate public and private keys, $K_a$ and $a$ respectively.
3: With public key of the server $K_s$ and private key $a$, generate shared symmetric encryption key $Sh_{ab}$ using the Shared Key Generation Block.
4: Encrypt the generated public key $K_a$ with the shared key $Sh_{ab}$.to form message $E_m$.
5: Transmit $E_m$ to the server.
6: At the server, once an encrypted message $E_m$ is received, Server decrypts it to get message $m$, which holds the public key of node A, $K_a$.
7: If $K_a$ received is equal to the $K_a$ stored in the database then consider node A authorized for the rest of the session.
8: Wait for other nodes to register themselves.
9: Server prepares a message for each ECU encrypting with the ECU public key containing the public keys of all the registered nodes.
10: Once this message is received at a node, decrypt it using the shared key.
11: Each node communicates with the other nodes and generates session keys using both its private key and other nodes public key.
```

C. Shared Key Generation Block

This block is used to generate a shared key between two communicating nodes to have an encrypted communication. For each device/ECU, the private key of the current node and the public key of the communicating device, are used in Elliptic curve Diffie-Hellman (ECDH) algorithm for the secret shared key exchange. This shared key is used in symmetric key encryption algorithm. All generated keys are stored in the volatile memory.

D. Key Storage

The key storage block consists of a non-volatile memory (NVM) to store:

- Seed, challenge and helper data to generate unique, random and reproducible PUF responses.
- Public key Ks of the server for server authentication.
- Parameters to generate public private key pairs.

The framework allows each ECU to generate the private key on fly thus does not require it to be stored in a NVM. Private keys can be stored temporarily on a volatile storage along with the shared session keys between ECU's. For each pair of communicating nodes, the following elements are stored in volatile Key Storage:

- Sh_{ab}: Generated Shared secret key for encryption for each pair of nodes (a being the current node and b the node being communicated).
- a: Private key generated at Node A.
- K_b: Public key of the communicating node.

E. Encryption / Decryption Block

Once shared keys are generated for each pair of nodes, these keys can be used for encryption and decryption for the given session. Every message that is sent must first be encrypted before it can be transmitted. Selection of a symmetric key algorithm is dependent on the choice of the system designer, for example an AES-256 engine. The message is sent to the Encryption / Decryption Block that encrypts the message at the sender and decrypts the message at the receiving node using the shared key.

V. IMPLEMENTATION AND PERFORMANCE ANALYSIS

Following section details the experimental setup and performance analysis. Xilinx Kintex KC705 FPGA Evaluation Board is an automotive approved part, and is used in demonstrating the complete RTL design using Xilinx Vivado suite.

A. PUF Implementation

For our design, we are using Hardware embedded delay PUF (HELP) PUF [10]. This is a strong PUF and extracts the entropy from the delay variations in the existing functional unit. HELP derives randomness from within-die path delay variations that occur along the paths within a hardware implementation that is an implementation of AES cryptographic primitive.

B. Key Generation Subsystem

We have selected an ECC engine operating over the binary field GF (2^163)[12]. For computation of a single point multiplication, the ECC core requires a total of 1428 cycles. With the clock running at 200 MHz, the total time to compute a single multiplication point is 7.14 microseconds. In the process for shared key generation, two single point multiplication operations at each node are performed. The ECC core uses a total of 25,454 LUT slices.

C. Encryption/ Decryption

The cryptographic engine AES-128 is implemented on the programmable logic to encrypt the communication. The AES engine uses the shared key generated by the ECC subsystem to encrypt the traffic leaving the node and decrypt incoming traffic. AES-128 engine implementation uses a total of 22 clock cycles for encryption and decryption. Considering the operating frequency of the target platform, it takes a total of 110 nanoseconds. The AES engine has a footprint of 2560 LUT slices.

Table III shows a comparison of different block computation speeds at different clock speeds. Table IV shows the time taken for message transmission at different CAN speeds during the authentication process.

TABLE III. COMPARITIVE ANALYSIS OF BLOCK COMPUTATION SPEEDS AT DIFFERENT SYSTEM CLOCK RATES

	AES-128	**ECC Operation**
At 60 MHz	366.66 ns	23.8 µs
At 100 MHz	220 ns	14.28 µs
At 200 MHz	110 ns	7.14 µs

D. Key Storage

The generated keys are stored in an on-chip volatile memory, block RAM resources. A block RAM of 128-bit width and 512 elements is instantiated. Key storage area overhead is 8KB BRAM. This memory element has a read and write speed of one cycle.

TABLE IV. OVERHEAD OVERVIEW AT STARNDARD CAN CONNECTION SPEEDS

	125 kbps	**500 kbps**	**1 Mbps**
Time for sending one frame	864µs	216µs	108µs
During Authentication			
Node sending encrypted message to server (2 frames)	1.728ms	432µs	216µs
Server's reply (3 frames)	2.592ms	648µs	324µs

VI. SECURITY ANALYSIS

The security enhanced framework improves the security at the device level and communication over the CAN bus. The proposed framework assumes a secure server, and the legitimate ECU's are not compromised and cannot perform unauthorized code execution.

In case of a masquerading node that falsely tries to register during the initialization process, the server will not find the corresponding public key in its public key database, hence eliminating the threat of eavesdropping or compromising the system by physically connecting to the network. Thus a malicious node cannot add to the network unless it is registered in the trusted environment.

In the scenario where an attacker has acquired the public key of a legitimate node on the network or in case of a server database compromise, adversary can only retrieve the public keys of participating ECUs. Once a communication from the adversary is initiated, the legitimate node will reject it since public key was not initially sent by the server.

VII. CONCLUSION

In this paper, we cover the threat model of the existing communication protocol including the vulnerabilities and different attacks on the electronic control units that have surfaced targeting internal automotive networks. We present a framework for providing secure and trusted communication over a non-secure vehicle network bus (CAN bus). The proposed hardware based security enhanced framework can be integrated with existing resource constraint embedded devices with real time response requirements.

VIII. ACKNOWLEDGEMENT

This work is supported by the NSF under grant no. 1566530 and 1623299.

IX. REFERENCES

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

[2] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," *Black Hat USA*, 2015.

[3] F. Garcia, D. Oswald, and T. Kasper, "Lock It and Still Lose It–On the (In) Security of Automotive Remote Keyless Entry Systems," in *USENIX Security 2016*, 2016.

[4] Argus Cyber Security, "A remote attack on an aftermarket telematics service," 2014. [Online]. Available: https://argus-sec.com/remote-attack-aftermarket-telematics-service/. [Accessed: 01-Apr-2017].

[5] A. Hanacek and M. Sysel, "Design and Implementation of an Integrated System with Secure Encrypted Data Transmission," Springer International Publishing, 2016, pp. 217–224.

[6] G. Singh, "A study of encryption algorithms (RSA, DES, 3DES and AES) for information security," *Int. J. Comput. Appl.*, 2013.

[7] A. Van Herrewege and D. Singelee, "CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus," *ECRYPT Work.*, 2011.

[8] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *2014 International Conference on the Internet of Things (IOT)*, 2014, pp. 13–18.

[9] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, Aug. 2014.

[10] W. Che, F. Saqib, and J. Plusquellic, "PUF-based authentication," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 337–344.

[11] NIST, "RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE," 1999.

[12] Y. Zhang, D. Chen, Y. Choi, L. Chen, and S.-B. Ko, "A high performance pseudo-multi-core ECC processor over GF(2 ^ 163)," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 701–704.