A PUF Based CAN Security Framework

Tyler Cultice, Carson Labrado, and Himanshu Thapliyal[1]

***Abstract*—We propose a method to include security and reliability to the messages sent over the CAN bus. Our approach adheres to CAN standard ISO 11898-1. A reliable PUF response is used in key generation to create a unique shared AES-256 key between each ECU, allowing for all message paths to be encrypted. In addition, an HMAC system with a counter is implemented to help protect against replay attacks and message tampering within the network.**

I. Proposed Design

We propose a more efficient and more secure CAN security framework utilizing encryption and hash engines. Proper usage of modern encryption algorithms, such as ECDH, call for large key sizes. Due to the large performance cost of messages, we propose methods to minimize the amount of packets required for authorization and produce more secure communication between authorized nodes. This proposed design requires no changes to the standard CAN protocol.

A. Architecture

The architecture of this system requires no change to the CAN protocol. To allow for random private key generation, a Physically Unclonable Function (PUF) is included with each node. These nodes should all have their own ID associated with a message ID. Without the unique IDs, the server would not know which shared key to decrypt with. The system should send a 16 byte message using two 8 byte CAN message payloads. The server should contain a TRNG which will be used to initialize the counter for each node.

The public keys are safe to be stored within nonvolatile memory. Private and shared keys should be generated each time during the authorization phase rather than being stored between sessions. The general layout is shown in Figure 1.

B. Encryption and MAC

The design will utilize ECDH to generate asymmetric keys and the shared key. Asymmetric keys will be derived

The authors are with the Department of Electrical and Computer Engineering, University of Kentucky, Lexington, KY, 40506 USA

[1]Contact e-mail: (hthapliyal@uky.edu). This research was partially supported by National Science Foundation Grant No:1738662.

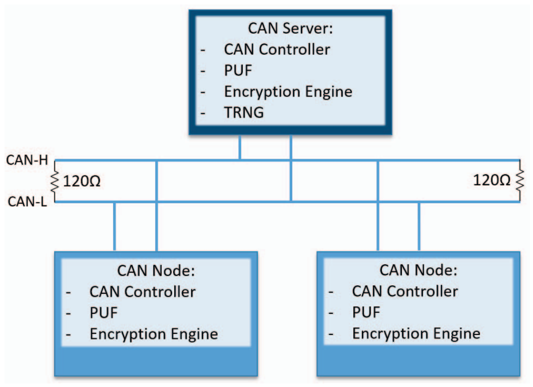

Fig. 1. Proposed CAN Authorization System

from PUFs. The PUF will generate the private keys each time they are needed. The standard of security for these algorithms call for key lengths that make previous proposed frameworks dramatically increase in message quantities. AES-128 or AES-256 are recommended for the symmetric encryption based on the similar block size and key size requirements to what is generated by the proposed CAN framework. All messages should be encrypted with AES using the shared key generated from one node's private key and the communicated node's public key. This results in 2 CAN frames for every message passed between nodes utilizing this framework.

AES must encrypt blocks of 16 bytes at a time. Standard CAN frames only contain 8 bytes of data so the remaining 8 bytes can be used for protection from replay attacks and data integrity. HMAC should be appended to packets and should be checked before the data is used by the ECU. Protection from replay attacks requires including a counter in the payload. The server will first initialize the counter with a random value and then share it with the nodes. Each node keeps a record of the counter and it should increment by at least one each time it is used. HMAC+counter systems have proved useful within a simple authorization protocol using CAN+ [1]. In our proposed implementation, the counter system is known by both the sender and receiver and an HMAC algorithm is used to ensure that sent authorization data is authentic.

C. Authorization Phase

Authorization will utilize a server node framework. Because authorization only requires data that can be verified and not easily replicated, any string of bytes known by both the server and node can be encrypted to provide sufficient verification proof. AES-256 uses block sizes of 16 bytes. Therefore, we can use any 16 byte string of data, such as hashed data, for our verification message. Due to the 16 byte minimum limit, we propose a specific message structure. The message contains a bit-field of 64 bits to transmit the verified list to each node relative to the node's list of communicated IDs. All node public keys can be stored in advance instead of having to send them over the bus. The nodes will use this bit field as a means to activate and deactivate communications to others based on the bit flags.

The remaining 64 bits of data will utilize a HMAC structure with counter to provide integrity to the data being sent. A node should immediately reject messages if the MAC does not properly authenticate the received data. This method of checking authenticity and security is called Authenticate-then-Encrypt (AtE). The AES encryption will result in a 16 byte final size per data block with HMAC.

D. After Authorization

After authorization, the nodes should generate shared keys using Diffie-Hellman for only the nodes that are authorized. This makes sure that the private and shared keys are both in volatile memory, assumed to be secure by the user. These keys should be used throughout the rest of the session to MAC and encrypt all communication between each node. Error handling should be used to ensure that attempts to communicate with disabled nodes are denied to prevent insecure communication or unexpected issues.

II. Performance and Security

The goal of security within systems that determine user safety is to provide the best security possible without significant cost. Time expensive frameworks and algorithms slow communication systems down risking the safety of the user. An ideal system provides strong security, affordable architecture, and low impact performance costs. For fair comparison, we are comparing our work with the existing framework [2] as a base.

A. Message Costs and Performance

The lowest standard of acceptable security according to NIST is 224 bits for the generated public key [3]. However, using a bit length of 256 and $len(p) = 256$, we can generate a key that is the size needed for symmetric AES encryption. For the sake of performance analysis we will consider the public key to be 256 bits per NIST standards. Packets sent from the node to the server for authorization require 16 bytes per node to verify. This is split into two messages, making the number of CAN messages sent for n nodes to be $2n$ for the proposed implementation. $4n$ messages are required to send the 256 bit public keys to verify n nodes. This results in a significant decrease from the existing approach [2] as the verification in our proposed design requires two packets less per node.

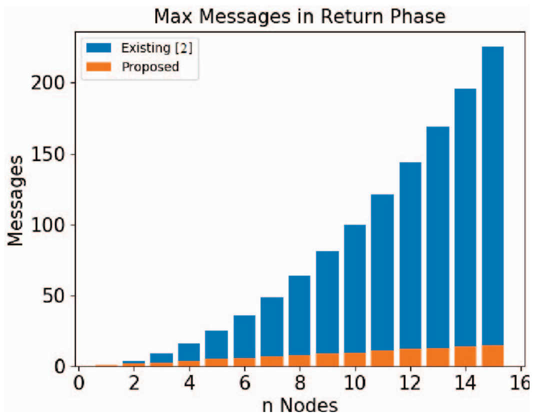

Fig. 2. Graph Comparing Returned Message Count

The largest performance gain is within the whitelisting phase of the framework. Sending a binary map of all the verified nodes to each node results in n messages for our proposed design. This is compared to the existing design's n^2 messages. As shown in Figure 2, this difference becomes very significant as the number of nodes grows.

III. Concluding Remarks

Many systems today still use the original CAN protocol and thus suffer the limitations with message size and transmission speed. Many CAN systems remain unencrypted, leaving these essential communication systems vulnerable to extremely simple hacks. Authorization and encryption need to become standard in a vehicle. Developing encryption methods that can be implemented without driver or device changes are essential to encouraging security in vehicular communication systems.

References

[1] A. Herrewege, D. Singele, and I. Verbauwhede, "Canauth - a simple, backward compatible broadcast authentication protocol for can bus," 01 2011, p. 7.

[2] A. S. Siddiqui, Y. Gui, J. Plusquellic, and F. Saqib, "A secure communication framework for ecus," *Advances in Science, Technology and Engineering Systems (ASTES)*, vol. 2, no. 3, pp. 1307–1313, 2017.

[3] L. Chen, D. Moody, A. Regenscheid, and K. Randall, "Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters," 10 2019, p. 9.