Prevention of Information Mis-translation by a Malicious Gateway in Connected Vehicles

Kyusuk Han and Kang G. Shin

Real-Time Computing Laboratory
EECS/CSE, The University of Michigan
Ann Arbor, MI 48109-2121, U.S.A.
{kyusuk,kgshin}@umich.edu

***Abstract*—As connectivity is becoming a norm for modern vehicles, data exchange between in-vehicle components and external entities is becoming common. However, car makers is concerned about a third-party's extraction of their proprietary vehicle data. To address this concern, an intermediate ECU (or a gateway) is introduced in between internal and external networks to translate proprietary in-vehicle data to rich type data, thus preventing the exposure of raw in-vehicle data. However, the translation relies solely on the gateway which can be a direct target of cyber attacks, making it difficult to trust the data passed through the gateway. This, in turn, requires authentication of the translated data. We present a new, effective protocol that provides secure communications between the vehicle's internal components and external entities against a compromised gateway.**

I. Introduction

Recent advances of in-vehicle technology have paved the way to connect vehicles to the external world. Car makers are adding various connectivity and telematics solutions for passenger and fleet vehicles. They have also introduced integrated solutions that either use an embedded modem or connect to the Internet via the driver/passenger's cellphone (e.g., GM OnStar, Ford Sync). Besides, fleet-solution providers offer solutions attachable to the vehicle's on-board diagnostics (OBD2) port (e.g., Delphi Connect and Zubie). As a result, in-vehicle components/subsystems are being connected to an external communication channel for remote diagnostics and triggering on-board functions from a remote site.

Externally-connected devices (e.g., cellphones/tablets) collect in-vehicle data from, and inject messages into, in-vehicle networks. A controller area network (CAN)—the *de facto* in-vehicle network—is connected to commonly available external networks, such as 3G/4G, WiFi, and Bluetooth. The device between internal and external networks is called an *external interface ECU*, or simply a *gateway* as shown in Fig. 1.

Car manufacturers do not want to expose their intellectual assets via vehicle connectivity since their in-vehicle message semantics are usually proprietary. Thus, the gateway translates in-vehicle data to rich type data (e.g., JSON, XML), concealing their proprietary data inside the vehicle.

However, the gateway may be compromised and then become a potential threat to vehicle safety and security. That is, since the transmission from and to an external entity relies entirely on the gateway, the communicated data becomes untrustworthy once the gateway is compromised. For example, the compromised gateway can make incorrect translation of, or drop/delay messages; we call this the *bogus interpreter problem*.

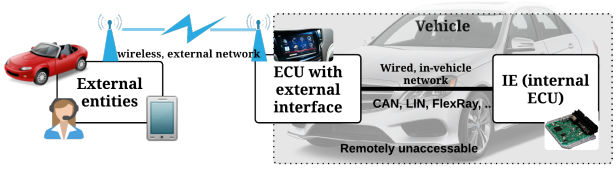

Fig. 1. In-vehicle components are connected to external entities through a gateway which is nothing but an ECU with external interfaces

Existing communication models only consider the communication security between the vehicle's gateway and an external entity by applying a network security layer, such as TLS. There have also been various efforts to provide cyber-vehicle security [2]–[4], [6], [7], [9], but they are still unable to secure data exchange between internal ECUs and external networks.

To remedy this deficiency, we propose a secure communication protocol between internal ECUs and external devices, and, in particular, provide data security against the compromised gateway. It includes the translation and security of end-to-end communication between an external entity (e.g., the car maker's server) and in-vehicle components that cannot be achieved with a naive approach such as TLS, mainly because the in-vehicle bus (e.g., CAN) and ECUs are severely resource-limited. Our protocol is shown to be resilient against the message forgery and drop by a compromised gateway.

The rest of this paper is organized as follows. Section II provides a brief overview of automotive system's characteristics and connectivity issues. Section III first introduces the bogus interpreter problem caused by a compromised gateway, and then specifies the security and design requirements. The drawbacks of state-of-the-art approaches are also discussed there. Section IV details our protocol for secure communica-

tion between internal ECUs and external devices. The security and performance of the proposed protocol are evaluated in Sections V and VI, respectively. Finally, the paper concludes with Section VII.

II. Data Exchange Between Internal and External Entities

We first briefly describe the characteristics of in-vehicle systems, and then discuss in-vehicle data exchange models and issues.

A. Characteristics of In-vehicle systems

Automotive systems are hard real-time systems, equipped with various electronic control units (ECUs) which are interconnected via wireline networks, such as Controller Area Networks (CAN), Local interconnect network (LIN), and Flexray. CAN has become the *de facto* standard for in-vehicle communications due to its low cost and widespread deployment.

1) In-vehicle network model: In-vehicle networks use the multi-master model. Every frame is broadcast on the bus, and every ECU listens to it and grabs only relevant frames by comparing their IDs (i.e., the arbitration field in each frame in Fig. 2) with those stored in the message/frame filter.

2) In-vehicle data format: CAN allows only up to 64 bits for data and provides a 16-bit CRC field (with a 1-bit CRC delimiter) to check the integrity of each received frame as shown in Fig. 2. Since the bus speed is only up to 1 Mbps, 30–40% of which is commonly utilized, it is important to use less bits of data in each frame.

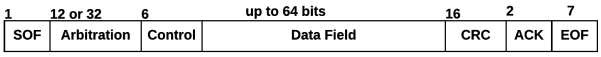

Fig. 2. CAN frame format

3) Proprietary in-vehicle data: CAN is the *de facto* standard and in-vehicle data is treated as the car-maker's proprietary information. Each car manufacturer uses its own semantic for the CAN ID and payload, hence concealing its proprietary information.

B. Data exchange between in-vehicle and external networks

The vehicle connectivity requires knowledge of CAN data for its intended use, forcing car makers to seek ways to make the data available without exposing its proprietary information to unauthorized entities. In addition, there is a large gap between in-vehicle and external networks, such as IP-based networks. So, car makers deploy ECUs with external interfaces as a gateway (GW) to communicate with the external world as shown in Fig. 1.

1) On-board diagnostics: Car makers have already have been extracting in-vehicle information for diagnostic purposes. Every car manufactured since 1996 is mandated to have an on-board diagnostic (OBD) port for emission control, enabling connection of a diagnostic device (OBD-II scanner).

Instead of revealing the actual data, car makers defined publicly accessible special types of messages for diagnostics purposes (i.e., diagnostic trouble codes (DTC)). As depicted in Fig. 3, the OBD-II scanner sends *parameter IDs* (PID) to ECUs through the OBD-II port. Recipient ECUs then respond with DTC on PID.

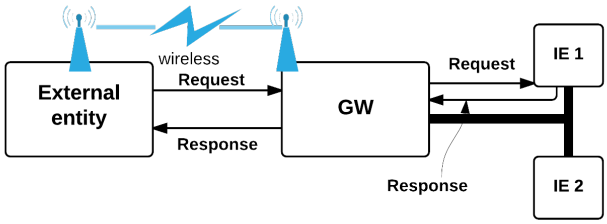

Fig. 3. System model 1: External entity requests data from an internal ECU.

2) Exchanging real in-vehicle data: While the diagnostics messages are limited to providing the information service, more advanced applications such as remote control, intrusion detection, and autonomous driving, will eventually require utilization of actual data by authorized external entities as shown in Fig. 4. Thus, car makers are trying to allow exposure of real in-vehicle messages without revealing their proprietary asset.

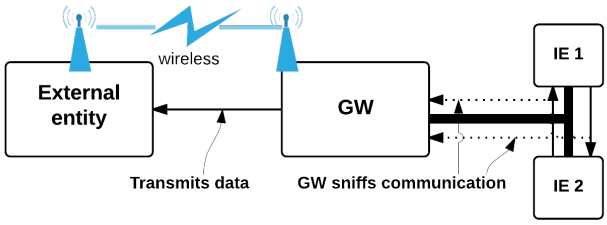

Fig. 4. System model 2: External entity reads data from the bus.

3) In-vehicle data translation: Two ways of utilizing the internal vehicle data while concealing the original format are *encapsulation* and *translation*.

Encapsulation is commonly used in computer networks. The original data is masked until the authorized entity receives it. Message encryption is an example of encapsulation. However, encapsulation is difficult to use for the case when an entity wanting to use the data is not authorized to know the original CAN raw data. Usually, a large number of cars from various car makers are on the road. Thus, data translation is considered as a practical solution, converting the original data format to another. For example, Ford has recently introduced a new vehicular interface model, called *OpenXC*, that GW translates the proprietary CAN raw data to the readable JavaScript Object Notation (JSON) format data, enabling external devices to know the vehicle data without knowing actual raw in-vehicle data. Fig. 5 shows an example data translator converting raw CAN data to JSON format. (Section VI-A will provide more details of this.)

However, there is a serious security risk in data translation which we call "bogus interpreter problem," as we we discuss next.

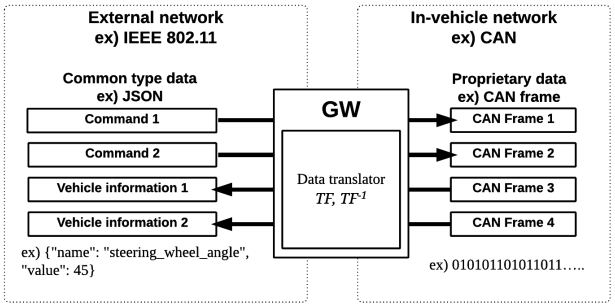

Fig. 5. The data translator in the gateway ECU (GW) enables the exchange of data in different formats

III. Bogus interpreter problem and counterpart

Since the gateway is the conduit to external networks, it can be the primary target by the cyber attackers [1], [8], [11]. A compromised gateway can turn into an attacker injecting messages into in-vehicle networks which we call the "bogus interpreter" problem. The design requirements for combating this problem are specified in in Section III-B.

A. Bogus interpreter's behavior

A compromised gateway may forge the original message, delay or even drop it. Since the original message format is not preserved, recipients cannot know whether the received messages are genuine, and the senders cannot know whether the messages are delivered to their intended receivers. We do

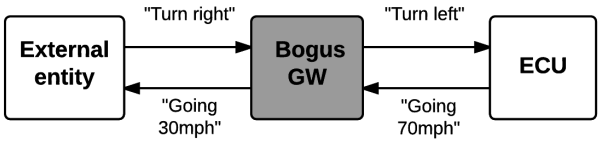

Fig. 6. Bogus gateway will change the original content.

not consider how to compromise the gateway itself and the privacy issues when the bogus gateway transmits the raw and translated messages to external attackers.

B. Design requirements

To prevent the bogus interpreter problem, the security architecture needs to meet the following design requirements.

- **In-vehicle Message Authentication**: The receiver ECUs must be able to determine if messages in in-vehicle networks are from a valid sender ECU.
- **Translated Message Authentication**: When messages are transmitted through the gateway, the receivers, external entities or ECUs should be able to determine their validity.
- **Proof of Delivery**: Either ECUs or external entities should be able to determine if messages are properly communicated to the intended receivers through the gateway.

C. State-of-art Approaches

In-vehicle networks had originally been designed to operate in an isolated environment, and hence security was not accounted for in their design. Although security is becoming important in automotive systems, it is difficult to modify the existing communication standard for cost and compatibility reasons, and hence security solutions have to be devised within the existing standards.

Several research efforts introduced the solutions to meet this requirement [3], [4], [6], [7], [9], [10], [12], [13], but they only considered communications *within* in-vehicle networks.

While attacks by compromising ECUs through an external gateway are demonstrated in [1], [8], [11], most of these efforts only considered communications between an external device and the infotainment component in the vehicle through wireless communications, e.g., Bluetooth and Wi-Fi; few considered security for communications between external devices and internal ECUs.

In a previous study [5] we proposed a three-step authentication protocol to establish a secure channel between external devices and internal ECUs through an intermediate gateway. They defined a key pre-distribution model among entities with different lifetimes, but did not consider the data exchange that conceals the original in-vehicle raw data and the attack by a compromised gateway.

None of the existing approaches considered the important case in which a compromised gateway mounts attacks on in-vehicle systems by using the secret key it has acquired.

IV. Proposed Design

We now present a novel way of securing information exchanged between internal and external components against the bogus (gateway) interpreter.

A. System model

We use two types of data as shown in Fig. 7 and list their notations in Table I.

a) Assumptions:

- ECUs and the gateway are equipped with cryptographic computation capabilities, as specified in the AUTOSAR standards.[1]
- Internal ECUs are trusted, as they are located inside of the vehicle, and are seldom changed and their access is limited during the vehicle's operation life. Physical hardware modifications are not considered in this paper.

[1]Automotive open system architecture, http://www.autosar.org/.

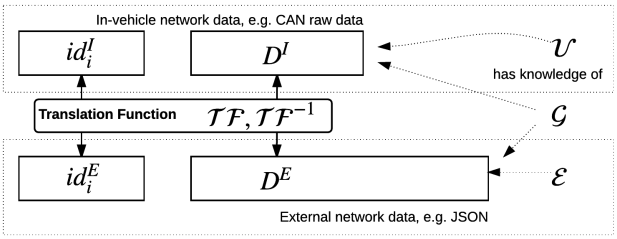

Fig. 7. Information flow between internal ECU U and external entity $\mathcal{E}$

TABLE I
NOTATIONS

Type	Description
id_i^I	Message ID i in the in-vehicle network. e.g., CAN frame ID defined by automotive manufacturers
id_i^E	Message ID i in the external network, associated with id_i^I.
$\mathcal{D}^I$	Data in the in-vehicle network. e.g., CAN raw data
$\mathcal{D}^E$	Common type data. We use JSON in this paper.
$\mathcal{U}_i$	Internal ECU i, only connected to the in-vehicle network.
$\mathcal{G}$	Gateway, connected to both in-vehicle and external networks
$\mathcal{E}$	External entity
$\mathcal{TF}$	Translation function. $\mathcal{D}^I \rightarrow \mathcal{D}^E$
$\mathcal{TF}^{-1}$	Inverse translation function: $\mathcal{D}^E \rightarrow \mathcal{D}^I$

- A trusted third party issues keys and has the knowledge of both in-vehicle data and common types, e.g., car makers know raw CAN data and its JSON translation.

B. Pre-phase: Secure channel setup

We follow the secure channel setup model of [5]. This phase consists of key pre-distribution and secure channel setup (see [5] for more details).

1) Pre-distribution of keys: The trusted third party (**TTP**) issues keys with different life times to each entity.

a) Key pre-distribution for internal ECUs: Suppose ECU $\mathcal{U}_i$ is designed to communicate with ECU $\mathcal{U}_j$ in the in-vehicle network, where $1 \leq i, j \leq n$, $i \neq j$, and n is the number of ECUs with a security function. First, the TTP randomly selects two long-term secret keys $lk_{i,j}$ and lk_i^I, where $lk_{i,j}$ is a shared key between $\mathcal{U}_i$ and $\mathcal{U}_j$, and lk_i^I is a seed key for secure communication with the gateway. The TTP then deploys $lk_{i,j}$ and lk_i^I in $\mathcal{U}_i$ during vehicle manufacturing. We assume this stage is secure, which is easy to satisfy.

b) Key distribution for the gateway: For the gateway $\mathcal{G}$ to communicate $\mathcal{U}_1$, TTP first randomly selects r_g and generates a timestamp TS_g, a mid-term secret key mk_1^I and certificate $cert_1^g$, where $mk_1^I = KDF(lk_1^I \ |r_g)$ and $cert_1^g = h\{lk_1^I, r_g|TS_g\}$, KDF is the key derivation function, and $h\{k, m\}$ is a MAC for message m with key k. For the other ECU U_i, the TTP generates mk_i^I and $cert_i^g$ and also randomly selects mk^E for communications with external entities.

$\mathcal{G}$ receives r_g, TS_g, a key set MK^I, a certificate set $CERT^g$, and a key mk^E from TTP during the installation stage, where $MK^I = \{mk_i^I|0 \leq i \leq n\}$, $CERT^g = \{cert_i^g|0 \leq i \leq n\}$. This stage is also assumed to be secure.

c) Key distribution for external entities: Suppose an external entity $\mathcal{E}$ wants connection to the vehicle, thus requesting keys from the TTP. Upon receiving the request from $\mathcal{E}$, the TTP first randomly selects r_e and a timestamp TS_e. Then, it generates a key set SK^I and a certificate set $CERT^I$, where $SK^I = \{sk_i^I|1 \leq i \leq n\}$, $CERT^I = \{cert_i^I|1 \leq i \leq n\}$, $sk_i^I = KDF(mk_i^I|r_e)$, $cert_i^I = h\{lk_i^I, r_e|TS_e\}$. The TTP also generates a secret key sk^E and a certificate $cert^E$, where $sk^E = KDF(mk^E, r_e)$, and $cert^E = h\{mk^E, \mathcal{E}|r_e|TS_e\}$.

2) Secure channel setup: This phase secures communications between internal ECUs and the external entity, secure channels between internal ECUs and the gateway over the in-vehicle network, and between the gateway and the external entity over an external network.

a) Authenticated key agreement between the gateway and internal ECUs: Let each $\mathcal{U}_i$ only accept id_i^I and also id_o^I. When the gateway $\mathcal{G}$ is attached to the vehicle for the first time, $\mathcal{G}$ broadcasts authentication requests to the in-vehicle network as follows.

First, $\mathcal{G}$ broadcasts 64-bit r_g and 32-bit TS_g with id_o^I to CAN bus. Then, $\mathcal{G}$ broadcasts n 64-bit certificates $cert_i^g$ with each id_i^I to CAN bus. Since a modern vehicle is equipped with 70–100 ECUs, and $\mathcal{G}$ may broadcast 72 to 102 CAN frames on the bus, each $\mathcal{U}_i$ may receive three values: r_g, TS_g and $cert_i^g$. After verifying r_g and TS_g with $cert_i^g$, each $\mathcal{U}_i$ generates mk_i^*, where $mk_i^* = KDF\{lk_i^I, r_g\}$ and $mk_i^* \equiv mk_i^I$. Note that mk_i^* is valid until TS_g expires.

b) Authenticated key agreement between the gateway and external entities: Unlike the in-vehicle network, we assume external networks support two-way communications. Let an external entity $\mathcal{E}$ attempt connection to the gateway $\mathcal{G}$. Then, $\mathcal{E}$ randomly selects a nonce r_1 and generates a challenge $chal$, where $chal = enc(sk^E, r_1|h\{r_1\})$, and $enc(k, m)$ denotes a function enc that encrypts a message m with key k. $\mathcal{E}$ then sends $chal$ with r_e, TS_e and $cert^E$ to $\mathcal{G}$.

After verifying $cert^E$, $\mathcal{G}$ generates sk^* and decrypts $chal$ with sk^*. $\mathcal{G}$ then randomly selects r_2 and generates the response res, where $res = h\{sk^*, \mathcal{G}|\mathcal{E}|r_e|r_1|r_2\}$. $\mathcal{G}$ sends r_2 and res to $\mathcal{E}$.

After verifying res, $\mathcal{E}$ generates a session key sk^e, where $sk^e = KDF\{sk^E, r_e|r^*\}$. $\mathcal{E}$ also generates the confirmation $con = h\{sk^e, \mathcal{E}|\mathcal{G}|r^*|r_e\}$, and sends con to $\mathcal{G}$. $\mathcal{G}$ also generates sk^e, and verifies con. If con is verified, then $\mathcal{G}$ stores sk^E until TS_e expires, and uses sk^e as a session key.

c) Authenticated key agreement between an external entity and internal ECUs: Upon receiving $CERT^I$ from $\mathcal{E}$, $\mathcal{G}$ broadcasts $CERT^I$ as follows: First, $\mathcal{G}$ broadcasts 64-bit r_e and 32-bit TS_e with id_o^I to CAN bus. Then, it broadcasts n 64-bit $cert_i^I$ to each $\mathcal{U}_i$. Optionally, $\mathcal{G}$ also broadcasts n 64-bit $proof_i$ to each $\mathcal{U}_i$, where $proof_i = h\{mk_i, r_e|TS_e|cert_i^I\}$.

$\mathcal{U}_i$ checks $proof_i$ and $cert_i^I$, and then derives sk_i^I.

C. Secure exchange of translated data via gateway

We present a secure translated data exchange protocol and two models: reading information from the bus, and request data from the ECU.

1) Data reading from the bus: Algorithm 1 shows the general in-vehicle communication between ECUs. When valid

Algorithm 1 General in-vehicle communication

procedure IN-VEHICLE COMMUNICATION
  for $\mathcal{U}_i$, $1 \leq i \leq n$, at frame count γ **do**
    Generate $\mathcal{D}_\gamma^I$ for id_j^I, $1 \leq j \leq n$, $i \neq j$
    $\mathcal{A}_\gamma^I \leftarrow h\{lk_{i,j}, \mathcal{D}_\gamma^I\}$
    Broadcast id_j^I, D_γ^I, $\mathcal{A}_\gamma^I$ to CAN bus
  end for
  for $\mathcal{U}_j$ **do**
    if Read messages with id_j^I **then**
      if $\mathcal{A}_\gamma^I$ is valid **then** Accepts D_γ^I
      end if
    end if
  end for
end procedure

$\mathcal{E}$ requests CAN data (let the count is γ), $\mathcal{G}$ reads the data from the CAN bus as described in Algorithm 2.

Algorithm 2 Data extraction from Bus

procedure DATA EXTRACTION FROM THE BUS
  for $\mathcal{G}$ **do**
    if read messages with id_j^I **then**
      $\{id_j^E, \mathcal{D}_\gamma^E\} \leftarrow TF(id_j^I, \mathcal{D}_\gamma^I)$
      $m_\gamma = \{id_j^E | \mathcal{D}_\gamma^E | \mathcal{A}_\gamma^I\}$
      while the number of m_γ ¡ K **do**
        Put m_γ into $\mathcal{M}^E$, $\mathcal{M}^E = \{m_\gamma | 1 \leq \gamma \leq K\}$
      end while
      $C^E \leftarrow enc(sk^e, \mathcal{M}^E)$ (Encryption)
      $\mathcal{A}^E \leftarrow h\{sk^e, \mathcal{G}|\mathcal{E}|C^E\}$ (MAC)
      Send $\mathcal{G}, \mathcal{E}, C^E, \mathcal{A}^E$ to $\mathcal{E}$
    end if
  end for
end procedure

Receiving C^E and $\mathcal{A}^E$ from $\mathcal{G}$, $\mathcal{E}$ proceed as described in Algorithm 3.

2) Data request from the ECU: Let $\mathcal{E}$ extract data from $\mathcal{U}_1$. Unlike the previous case, $\mathcal{E}$ needs to issue the data request to $\mathcal{U}_1$. It first sends the request to $\mathcal{G}$ as shown in Algorithm 4. $\mathcal{G}$ then relays the request to $\mathcal{U}$ as shown in Algorithm 5. Upon receiving a valid request, $\mathcal{U}$ sends the data to the gateway as described in Algorithm 6. $\mathcal{G}$ transmits the data to $\mathcal{E}$ as algorithm 3.

V. SECURITY EVALUATION

We now evaluate the security of the proposed protocol that combats a compromised gateway, or the bogus interpreter.

Algorithm 3 Translated data verification

procedure TRANSLATED DATA VERIFICATION
  for $\mathcal{E}$ **do**
    $\mathcal{A}^* \leftarrow h\{sk^e, \mathcal{G}|\mathcal{E}|C^E\}$
    if $\mathcal{A}^E \equiv \mathcal{A}^*$ **then**
      Retrieve $\mathcal{M}^E$ by decrypting C^E
      Send $\mathcal{M}^E$ to **TTP**
    end if
  end for
  for TTP do
    if Receive $\mathcal{M}^E$ from $\mathcal{E}$ **then**
      $\mathcal{D}_\gamma^I \leftarrow TF^{-1}(\mathcal{D}_\gamma^E)$
      $\mathcal{A}_\gamma^* \leftarrow h\{lk_{i,j}, \mathcal{D}_\gamma^I\}$.
      if $\mathcal{A}_\gamma^* \equiv \mathcal{A}_\gamma^I$, where $1 \leq \gamma \leq K$ **then**
        Return VALID.
      end if
    end if
  end for
end procedure

Algorithm 4 Sending data request to gateway

procedure SENDING DATA REQUEST TO THE GATEWAY
  for $\mathcal{E}$ **do** Contact **TTP**
    Get authorization of D^E for id_i^E
  end for
  for TTP do
    Find id_1^I relevant to id_i^E
    $\mathcal{D}^I \leftarrow TF^{-1}(\mathcal{D}^E)$
    $cert_i^t \leftarrow h\{lk_i^I | \mathcal{D}^I\}$
    Send $cert_i^t$ to $\mathcal{E}$
  end for
  for $\mathcal{E}$ **do**
    $C^e \leftarrow enc(sk^e, id_i^E | D^E | cert_i^t)$
    $\mathcal{A}^e \leftarrow h\{sk^e, \mathcal{E}|C^e\}$
    Send $C^e, \mathcal{A}^e$ to $\mathcal{G}$
  end for
end procedure

Suppose the compromised gateway Adv^g attempts to forge the communication messages between $\mathcal{U}_i$ and $\mathcal{U}_j$ and sends a bogus message to an external entity or to an internal ECU, or issues unauthorized/garbage messages to $\mathcal{U}_i$, or even drop/delay messages.

We do not consider the key update in this paper.

A. Forging translation of messages to an external entity

Adv^g can proceed with fake translation of the in-vehicle data $\mathcal{D}^I$ to $\mathcal{D}^{F,E}$.

To succeed in this attack, Adv^g generates $\mathcal{A}^{F,I}$, $\mathcal{A}^{F,I} \equiv \mathcal{A}^{*,I}$ without knowledge of $lk_{i,j}$, where $\mathcal{A}^{*,I} = h\{lk_{i,j}, \mathcal{D}^{F,E}\}$.

While Adv^g sends a set of messages $\mathcal{M}^E$ and a set of MACs $\mathcal{A}^E$, Adv^g has to generate K $\mathcal{A}^{F,I}$s. For example, compromising 64 different 3-bit $\mathcal{A}^{F,I}$'s is difficult to break 160 bit SHA-1, making the attack infeasible.

Algorithm 5 Inject data request to ECU

procedure INJECT REQUEST TO ECU
  for $\mathcal{G}$ **do**
    if Receive $C^e, \mathcal{A}^e$ **then**
      $\mathcal{A}^* \leftarrow h\{sk^e, \mathcal{E}|C^e\}$
      if $\mathcal{A}^e \equiv \mathcal{A}^*$ **then**
        $\{id_i^E, D^E, cert_i^t\} \leftarrow dec\{sk^e, C^e\}$
        $\{id_i^I, \mathcal{D}^I\} \leftarrow TF^{-1}(id_i^E, \mathcal{D}^E)$
        $\mathcal{A}_i^g \leftarrow h\{mk_i^I, id_i^I|cert_i^t|\mathcal{D}^I\}$
        Broadcast $id_i^I, \mathcal{D}^I, cert_i^t, \mathcal{A}_i^g$
      end if
    end if
  end for
  for $\mathcal{U}_1$ **do**
    if Receive messages with id_1^I **then**
      Verify $\mathcal{A}_1^g$ and $cert_1^t$
      if $\mathcal{A}_1^g$ and $cert_1^t$ are valid **then**
        Accept $\mathcal{D}^I$.
      end if
    end if
  end for
end procedure

Algorithm 6 Data Extraction from ECU

procedure DATA EXTRACTION FROM ECU
  for $\mathcal{U}_1$ **do**
    if Receive authorized request from $\mathcal{E}$ **then**
      Generate $\mathcal{D}^I$
      $\mathcal{A}^e \leftarrow h\{lk^I|\mathcal{D}^I\}$
      $\mathcal{A}^g \leftarrow h\{mk^I|\mathcal{D}^I|\mathcal{A}^e\}$
      Broadcast $id_g^I|\mathcal{D}^I, \mathcal{A}^e, \mathcal{A}^g$ to CAN bus
    end if
  end for
  for $\mathcal{G}$ **do**
    if $\mathcal{A}^g$ is valid **then**
      $\mathcal{D}^E \leftarrow TF(\mathcal{D}^I)$
      $\mathcal{C}^E \leftarrow enc(sk^e, id_1^E|D^E|\mathcal{A}^e)$
      $\mathcal{A}^E \leftarrow h\{sk^e, \mathcal{C}^E\}$
      Send $\mathcal{C}^E, \mathcal{A}^E$ to $\mathcal{E}$
    end if
  end for
end procedure

B. Forging translation of messages to an internal ECU

Adv^g may forge the information from $\mathcal{E}$ and send the fake translation $\mathcal{D}^{F,I}$ to $\mathcal{U}_i$. To mount this attack, Adv^g has to generate a 16 or 32-bit $cert^F$ without knowing lk_i^I, where $cert^F \equiv cert_i^t$ and $cert_i^t = h\{lk_i^I|\mathcal{D}^{F,I}\}$. Thus, this attack is practically infeasible.

C. Inject bogus messages in the in-vehicle network

Adv^g may attempt to generate and send fake information to $\mathcal{U}_i$. In this attack, Adv^g can generate the fake message $\mathcal{D}^{F,I}$ and $cert^F$. However, as in the previous case, Adv^g also has to generate $cert^F$ without knowing $lk_{i,j}$, making the attack practically infeasible.

D. Dropping messages

Adv^g may attempt to drop messages between $\mathcal{E}$ and $\mathcal{U}_i$. Internal ECUs cannot notice if Adv^g dropped messages, but $\mathcal{E}$ can notice if the attack is mounted. For example, $\mathcal{E}$ has a rule that it receives messages from $\mathcal{G}$ periodically, or receives response upon request within a certain time.

VI. PERFORMANCE EVALUATION

We now evaluate the performance of the proposed protocol.

A. Evaluation setup

Since CAN data are proprietary for car manufacturers and hence unavailable, we designed our own CAN frame format, called RTCL-CAN, based on the public information made available by Ford Motor Company.[2] Table II lists the CAN frame definitions of RTCL-CAN. RTCL-CAN defines 19 types of information including steering wheel angle, torque at transmission, engine speed, and so on.

Each data in RTCL-CAN can be translated to/from JSON by the gateway $\mathcal{G}$ and **TTP** using TF and TF^{-1}. Next, we present an illustrative example of translation.

1) A CAN translation example: As shown in Table II, a frame for controlling the transmission gear position has CAN ID 0x06 and 4-bit data. It has 8 different states as shown in Table III. Note the translation rule in this table is only for illustration, and the actual rules are proprietary to vehicle manufacturers and hence unavailable.

Fig. 8 shows how CAN raw data of the transmission gear position is set to 'sixth.' In the raw CAN data $\mathcal{D}^i$, **SOF** and **EOF** fields are fixed, and **RTR** is also set to '0' for a data frame. ID '00000000110' indicates that the frame is for the transmission gear position. Data field contains '0110' and MAC.

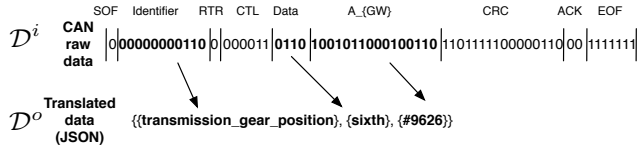

Fig. 8. Example CAN translation: transmission gear position at 'sixth'

$\mathcal{G}$ converts $\mathcal{D}^I$ to the translated format $\mathcal{D}^E$. $\mathcal{G}$ translates the data with '**transmission_gear _position**', '**sixth**' using Table III and $\mathcal{A}^I$.

When $\mathcal{E}$ sends a command $\{\{transmission\_gear\_position\}, \{neutral\}\}$ to $\mathcal{G}$, $\mathcal{G}$ translates it to '0x006' and '0000'.

[2]http://www.openxcplatform.com

TABLE II
RTCL-CAN FRAMES

Data type	Identifier	data bits	T ms	Frame bits	Data type	Identifier	data bits	T ms	Frame bits
steering_wheel_angle	0x030	11	100	60	torque_at_transmission	0x035	12	100	60
engine_speed	0x055	14	100	60	vehicle_speed	0x065	10	100	60
accelerator_pedal_position	0x020	7	100	52	parking_brake_status	0x00001	0	1000	44
brake_pedal_status	0x00010	0	1000	44	transmission_gear_position	0x006	4	1000	52
odometer	0x10011	34	100	84	ignition_status	0x00011	2	1000	52
fuel_level	0x01101	7	500	52	fuel_consumed_since_restart	0x10010	36	100	84
door_status	0x10001	4	1000	52	headlamp_status	0x00100	1	1000	52
high_beam_status	0x00101	0	1000	44	windshield_wiper_status	0x00111	1	1000	52
latitude	0x01111	12	1000	60	longitude	0x01110	13	1000	60
button_event	0x10000	6	N/A	52					

TABLE III
TRANSMISSION_GEAR_POSITION (4 BITS) ID 0X06, FREQUENCY = 1HZ, AND IMMEDIATELY AFTER THE DRIVER CHANGES THE GEAR POSITION

Data	Status	Data	Status	Data	Status	Data	Status
0000	neutral	0001	first	0010	second	0011	third
0100	fourth	0101	fifth	0110	sixth	0111	seventh
1000	eighth	1001	reserve	1010	reserve	1011	reserve
1100	reserve	1101	reserve	1110	reserve	1111	reverse

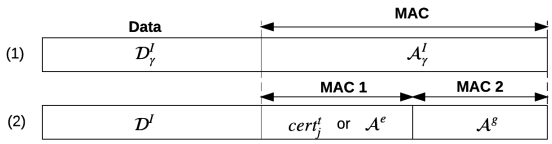

Fig. 9. (1) One MAC is used for sniffing case, (2) two MACs are assigned for data request case. Total size of MAC 1 and MAC 2 in (2) is the same as MAC in (1).

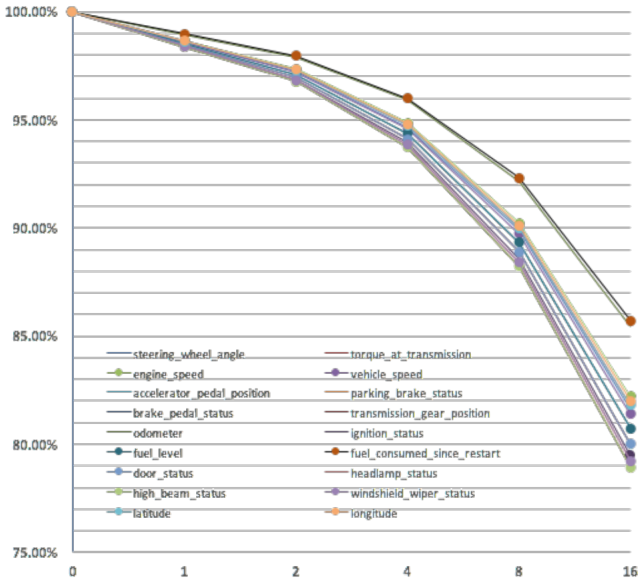

Fig. 10. Performance comparison with different sizes of MAC1 with 16-bit MAC2

B. Assigning MAC

Since CAN only provides up to 64 bits of data, it is not possible to assign a full-size MAC; for example, HMAC with SHA-1 generates 160-bit hash outputs. Thus, we assign truncated MACs for CAN, e.g., $\mathcal{A}_\gamma^I$ or $\mathcal{A}^g$.

We use two types of MAC as shown in Fig. 9: (1) MAC for the sniffing case and (2) MAC for the messages on request. Note that external networks, such as $\mathcal{A}^E$, utilize full-size MACs.

C. Communication overhead

1) In-vehicle communication overhead: We evaluate the in-vehicle communication overhead of the protocol for different sizes of **MAC 1**. The overhead of **MAC 2** is the same as other general models.

Fig. 10 shows the comparison of performance for different MAC sizes in RTCL-CAN of Table II during a 1-second period. Assigning 1 bit to **MAC 1** with 16-bit **MAC 2** yields 98.70% of performance of the original RTCL-CAN with a 16-bit MAC. Assigning 2, 4, 8, and 16 bits shows 97.43, 94.98, 90.45 and 82.56% of the original RTCL-CAN, respectively.

2) External communication overhead: While the above results show that assigning 1-bit **MAC 1** provides the best performance, we also have to consider the external communication overhead.

The size of set $\mathcal{M}^E$, is determined based on the size of **MAC 1**. The size of MAC is recommended to be more than 64 bits to achieve the practical security strength. Thus, assigning 1 bit for **MAC 1** requires $64 \leq n$ at a time. For example, RTCL-CAN generates 82 messages per second and approximately an 8.2-Kbyte message can be transmitted at a time over external networks. The probability of Adv^g's success in compromising RTCL-CAN is $^1/_{2^{82}}$ with 1-bit **MAC 1**. However, sending large-size $\mathcal{M}^E$ could degrade performance, since $\mathcal{E}$ discards the entire set of messages even when one iof many messages transmitted over the external network is found to have been compromised.

In contrast, assigning a larger-size **MAC 1** ($\mathcal{A}_\gamma^I$) reduces potential transmission overhead over the external network,

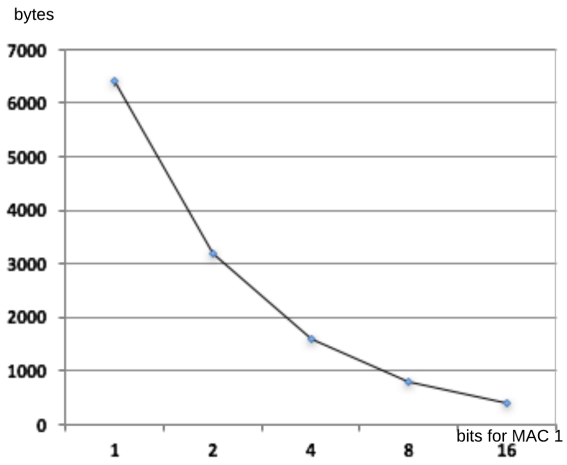

Fig. 11. The minimum size of $\mathcal{M}^E$ that $\mathcal{G}$ transmits to $\mathcal{E}$ over external networks at a time

although it incurs slight performance degradation in CAN.

Fig. 11 shows the data transmission overhead for different sizes of **MAC 1**. Although assigning 16 bits to **MAC 1** incurs approximately 18% performance degradation in CAN, it provides not only 16x better efficiency for transmission than assigning 1 bit, but also enables detection of compromised $\mathcal{D}_\gamma^E$ in $\mathcal{M}^E$.

3) Tradeoff in a real network environment: The bit assignment depends on the underlying network environment. For example, implementing CAN-FD or FlexRay allows more bits for **MAC 1**. Also, implementing faster technologies such as LTE, LTE-A as external networks supports larger $\mathcal{M}^E$.

Urban areas generally have better network connectivity, so assigning a smaller MAC in each CAN frame will preserve the performance of the in-vehicle network. In contrast, rural areas may only support poor connectivity, so assigning a larger MAC could preserve the performance in external networks.

D. Data processing overhead

The computation overhead is a critical issue to powertrain control systems and SAE specifies $1ms$ as an allowable delay for safety-critical control systems. In our protocol, $\mathcal{U}_i$ only needs 2 hash computations (**2H**) to authenticate a message. $\mathcal{U}_i$ generates $\mathcal{A}^I$ and $\mathcal{A}^g$, and verifies $cert$ and $\mathcal{A}^I$. Thus, we evaluated the overheads on two industry-use processors; 32-bit MIPS (Chipkit MAX32) and ARM Cortex M3 (LPC1768) as $\mathcal{U}_i$ by implementing the HMAC-SHA1 function. The time for **2H** only takes about 460 μs and 45 μs which is far below the 1-ms requirement.

VII. Conclusion

As vehicles and external devices are increasingly connected, their safety and security have become a serious concern. In this paper, we have proposed a secure communication protocol between internal and external components through a gateway. This protocol is shown to solve the bogus interpreter problem, securing communications of connected vehicles. This protection is achieved at the cost of minor performance degradation on RTCL-CAN, which is designed in compliance with CAN data format made available by Ford and can thus be used as a practical reference implementation.

References

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *SEC'11: Proceedings of the 20th USENIX conference on Security*. USENIX Association, Aug. 2011, pp. 1–16.

[2] A. Elouafiq, "Authentication and encryption in GSM and 3G UMTS: An emphasis on protocols and algorithms," *CoRR*, vol. abs/1204.1651, Apr. 2012.

[3] B. Groza and P.-S. Murvay, "Broadcast authentication in a low speed controller area network," *E-Business and Telecommunications, International Joint Conference, ICETE 2011, Seville, Spain, July 18-21, 2011, Revised Selected Papers*, vol. 314, pp. 330–344, Feb. 2012.

[4] B. Groza, S. Murvay, A. van Herrewege, and I. Verbauwhede, "LiBrA-CAN: a lightweight broadcast authentication protocol for controller area networksan: a lightweight broadcast authentication protocol for controller area networks," *Proceedings of 11th International Conference, CANS 2012, Darmstadt, Germany.*, pp. 185–200, December 2012.

[5] K. Han, S. D. Potluri, and K. G. Shin, "On authentication in a connected vehicle: Secure integration of mobile devices with vehicular networks," *Proceeding of ICCPS 2013*, pp. 160–169, Apr. 2013.

[6] O. Hartkopp, C. Reuber, and R. Schilling, "MaCAN - message authenticated CAN," *escar 2012, Embedded Security in Cars Conference 2012, Berlin - Germany*, November 2012.

[7] A. V. Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth - a simple, backward compatible broadcast authentication protocol for CAN bus," *10th escar Embedded Security in Cars Conference*, November 2011.

[8] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental security analysis of a modern automobile," *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 447–462, 2010.

[9] C. W. Lin and A. Sangiovanni-Vincentelli, "Cyber-security for the controller area network (CAN) communication protocol," *ASE Science Journal*, vol. 1, no. 2, pp. 80–92, December 2012.

[10] D. Nilsson, U. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, 2008, pp. 1–5.

[11] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study," in *USENIX Security'10: Proceedings of the 19th USENIX conference on Security*. USENIX Association, Aug. 2010.

[12] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, "Car2X communication: Securing the last meter," *WIVEC 2011, 4th IEEE International Symposium on Wireless Vehicular Communications, 5-6 September 2011, San Francisco, CA, United States*, pp. 1–5, Jun. 2011.

[13] C. Szilagyi and P. Koopman, "Low cost multicast authentication via validity voting in time-triggered embedded control networks," *Proceedings of the 5th Workshop on Embedded Systems Security*, pp. 10:1–10:10, 2010. [Online]. Available: http://doi.acm.org/10.1145/1873548.1873558