TACAN: Transmitter Authentication through Covert Channels in Controller Area Networks

Xuhang Ying
Department of Electrical and Computer Engineering
University of Washington
Seattle, WA 98195
xhying@uw.edu

Giuseppe Bernieri
Department of Mathematics
University of Padua
Padua, Italy
bernieri@math.unipd.it

Mauro Conti
Department of Mathematics
University of Padua
Padua, Italy
conti@math.unipd.it

Radha Poovendran
Department of Electrical and Computer Engineering
University of Washington
Seattle, WA 98195
rp3@uw.edu

ABSTRACT

Nowadays, the interconnection of automotive systems with modern digital devices offers advanced user experiences to drivers. Electronic Control Units (ECUs) carry out a multitude of operations using the insecure Controller Area Network (CAN) bus in automotive Cyber-Physical Systems (CPSs). Therefore, dangerous attacks, such as disabling brakes, are possible and the safety of passengers is at risk. In this paper, we present TACAN (Transmitter Authentication in CAN), which provides secure authentication of ECUs by exploiting the *covert channels* without introducing CAN protocol modifications or traffic overheads (i.e., no extra bits or messages are used). TACAN turns upside-down the originally malicious concept of covert channels and exploits it to build an effective defensive technique that facilitates transmitter authentication via a trusted Monitor Node. TACAN consists of three different covert channels for ECU authentication: 1) Inter-Arrival Time (IAT)-based, leveraging the IATs of CAN messages; 2) offset-based, exploiting the clock offsets of CAN messages; 3) Least Significant Bit (LSB)-based, concealing authentication messages into the LSBs of normal CAN data. We implement the covert channels on the University of Washington (UW) EcoCAR testbed and evaluate their performance through extensive experiments. We demonstrate the feasibility of TACAN, highlighting no traffic overheads and attesting the regular functionality of ECUs. In particular, the bit error ratios are within 0.1% and 0.42% for the IAT-based and offset-based covert channels, respectively. Furthermore, the bit error ratio of the LSB-based covert channel is equal to that of a normal CAN bus, which is 3.1×10^{-7}%.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Security and privacy** → *Cryptography*;

ICCPS '19, April 16–18, 2019, Montreal, QC, Canada

ACM ISBN 978-1-4503-6285-6/19/04…$15.00
https://doi.org/10.1145/3302509.3313783

KEYWORDS

Transmitter authentication, Controller Area Network (CAN), covert channel, Cyber-Physical System (CPS) security, intrusion detection

ACM Reference format:
Xuhang Ying, Giuseppe Bernieri, Mauro Conti, and Radha Poovendran. 2019. TACAN: Transmitter Authentication through Covert Channels in Controller Area Networks. In *Proceedings of 10th ACM/IEEE International Conference on Cyber-Physical Systems (with CPS-IoT Week 2019), Montreal, QC, Canada, April 16–18, 2019 (ICCPS '19),* 12 pages.
https://doi.org/10.1145/3302509.3313783

1 INTRODUCTION

The Controller Area Network (CAN) enables communication among Electronic Control Units (ECUs) for closed in-vehicle networks [2, 16]. The security of CAN bus is crucial to the functionality and safety of today's automobiles and future's autonomous cars [1, 34]. Since the CAN bus is a broadcast medium without authentication, a compromised ECU can be used to masquerade as any targeted ECU by transmitting messages with the forged message ID (masquerade attack [3]). Modern externally accessible ECUs with additional connectivity interfaces such as cellular, Wi-Fi or Bluetooth disrupt the closed in-vehicle network assumption. Consequently, the CAN bus has shown to be vulnerable to cyber attacks, such as disabled brakes [3] and remotely controlled steering [30].

Use of cryptographic primitives such as message authentication is one way to defend against attacks (notably the masquerade attack) on the CAN bus. However, it can be challenging in practice due to the low throughput and tight bit budget of the CAN protocol, and current solutions such as [14, 20, 26, 33] require protocol modifications or introduce traffic overheads. An alternative is to deploy anomaly-based Intrusion Detection Systems (IDSs) without modifying the CAN protocol [5–7, 25], including timing-based and voltage-based IDSs. The timing-based IDS in [5] exploits CAN message periodicity to estimate clock skew as a unique fingerprint to detect masquerade attacks. Nevertheless, it was later shown to be ineffective against the cloaking attack that modifies the inter-transmission time to emulate the clock skew of the targeted ECU [28, 35]. The voltage-based IDSs [6, 7, 17, 24] attempt to fingerprint the attacker through voltage signal characteristics. However,

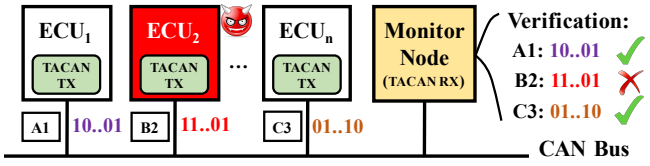

Figure 1: Illustration of TACAN. Legitimate ECUs transmit unique authentication messages that are embedded into the timing and LSBs of normal CAN messages (e.g., A1, B2, and C3) using the proposed covert channel methodologies. The Monitor Node authenticates transmitting ECUs by verifying the received authentication messages. If the compromised ECUs cannot generate valid authentication messages, then the attack will be detected by the Monitor Node.

if the attacker uses IDs that the compromised ECU is allowed to use under normal conditions, the attack will not be detected.

In this work, we develop TACAN that allows the trusted Monitor Node (MN) to verify the authenticity of a transmitting ECU and detect CAN bus anomalies, as illustrated in Figure 1. In TACAN, a master key is shared between an ECU and the MN for generating shared session keys. Consistently with [14, 26, 33], we assume that the keys are stored in the tamper-resistant memory of a security module such as the Trusted Platform Module (TPM) [12]. Each ECU embeds unique authentication messages into CAN messages and continuously transmits them through covert channels, which can be received and verified by the MN.

Therefore, if the attacker has no access to the TPM of the targeted ECU, it cannot use the compromised ECU or external device to generate valid authentication messages, thus causing verification failures and triggering the alarm at the MN side. Moreover, TACAN will detect attacks (e.g., Denial-of-Service (DoS) attack) that interrupt the transmission of CAN messages that are used to embed the authentication messages. The main benefits of using covert channels for TACAN are that they do not introduce protocol modifications or traffic overheads (i.e., extra bits or messages). In addition, by requiring ECUs to transmit authentication messages much less frequently than per-message authentication schemes, TACAN can significantly reduce the computational burden of the resource-constrained ECUs.

Contributions: In this paper, we make the following contributions:

- We identify and exploit covert channels to facilitate ECU authentication on the CAN bus. Hence, covert channels are used for security instead of malicious communication.
- We propose TACAN, which consists of a suite of three covert channels for transmitting authentication messages, including two timing-based covert channels that modify the inter-transmission times of CAN messages to affect the inter-arrival times (IATs) and offsets observed by the MN, and one storage-based covert channel that hides the information in the LSBs of the data payload of normal CAN messages.
- We implement the covert channels of TACAN in a real vehicle testbed (the UW EcoCAR [10]). We also conduct extensive experiments to demonstrate the feasibility of such covert channels using two real CAN traffic datasets, the publicly available Toyota dataset [8] and the EcoCAR dataset. Our results show that the bit error ratios are within 0.1% and 0.42% for the IAT-based and offset-based covert channels, respectively. The bit error ratio of the LSB-based covert channel is equal to 3.1×10^{-7}%, which is the bit error ratio of a typical CAN bus.

Organization. The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 presents our system and adversary models. Section 4 presents TACAN. A security discussion of TACAN is provided in Section 5. Section 6 presents experimental evaluation. Section 7 concludes this paper.

2 RELATED WORK

Recent experimental studies have demonstrated that an attacker is able to infiltrate in-vehicle ECUs physically or remotely and mount cyber attacks that would cause potentially life-threatening consequences by disabling breaks or overriding steering [3, 22]. One way to secure the CAN bus is to deploy anomaly-based IDSs based on traffic analysis (e.g., timing/frequency [15]), entropy [25], or physical invariants such as clock skew [5, 28] and signal characteristics [6, 7, 17, 24]. While voltage-based IDSs are effective against ongoing masquerade attackers, they cannot detect a compromised ECU before attacks are launched (e.g., a stealthy attacker may not launch the attack until the car is in drive mode). Moreover, it has been recently shown in [27] that the extra wires required by voltage-based IDSs may introduce new attack surfaces.

Researchers have also attempted to add cryptographic primitives such as Message Authentication Code (MAC) to the CAN bus, including CANAuth [33], LCAP [14], CaCAN [20], and LeiA [26]. Due to the tight bit budget and low throughput of CAN, authentication information is usually embedded into existing CAN messages (i.e., the ID or data field) and transmitted through additional CAN messages, thus introducing traffic overheads or increasing the bus load [14, 20, 26]. In this work, we focus on transmitter authentication rather than per-message authentication to avoid such traffic overheads. The key novelty of this work is the use of covert channels, a well-known malicious technique that is converted into defensive applications for authentication purposes. Compared to previous authentication schemes, our scheme does not require protocol modifications or introduce extra bits or CAN messages.

In literature, there are two main categories of covert channels: timing-based and storage-based. In timing-based covert channels, only the timing of events or traffic is modified to share information between two parties and the contents of data stream remain intact. The storage-based covert channels hide data in a shared resource that is not designed for transferring data, e.g., by exploiting reserved or used fields in the data packet or concealing data in the payload. Compared to steganography techniques that require some form of content as cover, the covert channels exploit network protocols as carrier [36]. In [32], Taylor *et al.* propose an approach where the exploitation of covert channel enhances the Modbus/TCP protocol security for industrial control system applications. To the best of our knowledge, this is the first paper that explores covert channels for automotive CAN buses.

Table 1: Frequently used notations.

Notation	Description
MK_i, SK_i	Master and session keys of ECU_i
g_i, l_i	Global and local counters of ECU_i
T	Message period
S	Clock skew
t_i, a_i	Transmit and arrival timestamps
η_i	Noise in the arrival timesatmp of message i
Δt_i	Inter-transmission time (ITT) between messages $(i-1)$ and i
Δa_i	Inter-arrival time (IAT) between messages $(i-1)$ and i
$\Delta\bar{a}[i]$	The i-th sample of averaged Δa_i
δ	Deviation (added to ITTs at the transmitter side)
L	Window length or number of least significant bits
κ, Γ_u, Γ_l	Reference level, upper threshold, lower threshold
A_m, A_f	Authentication message, authentication frame
n_m, n_f, n_s	Number of bits in A_m and A_f; number of silence bits
$\hat{O}_k[i]$	The observed clock offset up to message i in batch k

3 SYSTEM AND ADVERSARY MODELS

In this section, we present the system (Section 3.1) and adversary (Section 3.2) models for the CAN bus. A list of frequently used notations is provided in Table 1.

3.1 System Model

CAN Bus. As a broadcast medium, the CAN bus connects all ECUs to the same, shared bus line and allows them to transmit any messages to any ECU and observe all ongoing transmissions. Each CAN frame (or message) has a set of predefined fields, including notably the Arbitration field (which includes the message ID) and the Data field (up to 8 bytes). More details about the CAN frame structure are provided in Appendix A.1.

If two (or more) ECUs attempt to transmit messages simultaneously, an arbitration scheme based on priority (a smaller message ID indicates a higher priority) is used to determine which ECU transmits first. The CAN messages do not have transmit timestamps and do not support encryption or authentication.

Clock Skew. In automotive CAN, most messages are transmitted periodically as per the local clocks of transmitting ECUs[1]. Since there exists no clock synchronization among ECUs, the frequencies of local clocks are different due to clock skew, a physical property caused by variations in the clock's hardware crystal.

Let $C_A(t)$ be the time reported by clock A and $C_{true}(t) = t$ be the true time. According to the Network Time Protocol (NTP) [23], the *clock offset* of clock A is defined as $O_A(t) = C_A(t) - C_{true}(t)$, and the *clock skew* is the first derivative of clock offset, i.e., $S_A(t) = O'_A(t) = C'_A(t) - 1$, which is usually measured in microseconds per second (μs/s) or parts per million (ppm). In the absence of a true clock, the *relative clock offset* and *relative clock skew* can be defined with respect to a reference clock.

Timing Model. As illustrated in Figure 2, we let t_i be the transmit time of message i (assuming $t_0 = 0$) and $\Delta t_i = t_i - t_{i-1}$ be the inter-transmission time according to the transmitter's clock. If messages are transmitted every T seconds, we have $\Delta t_i = T$ and $t_i = iT$. The receiver's clock is considered as the reference clock. In practice, there exists a clock skew in the transmitter's clock relative to the reference clock, which introduces an offset O_i between the two clocks. Therefore, the actual transmit time is $t'_i = t_i - O_i$ according to the reference clock.

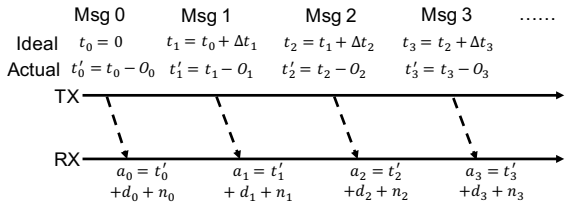

Figure 2: Illustration of our timing model.

While the clock skew may be slowly varying due to factors including the temperature, the clock skew is almost constant over a short duration. Given a constant clock skew S, the relationship between the elapsed time t_i at the transmitter and the elapsed time t'_i at the receiver is given by $S = (t_i - t'_i)/t'_i$. Hence, we have $t'_i = t_i/(1+S)$, and $O_i = t_i - t'_i = \frac{S}{1+S}t_i$. To account for offset deviations due to jitters, we model the actual clock offset $O_i = \frac{S}{1+S}t_i + \epsilon_i$, where ϵ_i's are assumed to be i.i.d. zero-mean random variables.

After a network delay of d_i (due to message transmission, propagation, arbitration, and reception) and the zero-mean quantization noise n_i [37], the arrival timestamp of message i is

$$a_i = t_i - O_i + d_i + n_i = t_i - \frac{S}{1+S}t_i + \eta_i = \frac{1}{1+S}t_i + \eta_i, \quad (1)$$

where $\eta_i = -\epsilon_i + d_i + n_i$ is the overall noise. Since periodic CAN messages have constant data lengths over time, it is reasonable to assume constant-mean network delays, i.e., $\mathbb{E}[d_i] = d$. Hence, η_i's can be modeled as i.i.d. random variables with a mean of d and a variance of σ_η^2.

3.2 Adversary Model

We consider an adversary who aims to infiltrate the CAN bus and launch stealthy attacks without being detected. We assume that the adversary can passively monitor and observe all ongoing CAN transmissions. In addition, it has full knowledge of the deployed covert channels and can also observe all authentication messages that are being transmitted. In reality, there are usually two ways of gaining unauthorized access to the CAN bus: 1) compromise an in-vehicle ECU either physically or remotely [3], or 2) plug an external device (a malicious ECU) into the CAN bus [18]. We assume that the adversary has no access to the keys stored in the TPM of the compromised ECU and other legitimate ECUs.

The adversary can use the compromised or malicious ECU to perform three basic attacks: 1) suspension, 2) injection, and 3) masquerade attacks, as considered in [5, 21, 28]. As illustrated in Figure 3(a), a suspension attacker prevents the compromised ECU from transmitting certain messages, while an injection attacker can fabricate and inject CAN messages of arbitrary choices of message ID, content, and timing, as sketched in Figure 3(b). Injection attacks can lead to more sophisticated attacks such as the DoS attack [15] and the bus-off attack [4]. In the (stealthy) masquerade

[1]In the UW EcoCAR (Chevrolet Camaro), all of the 89 messages with distinct IDs are periodic with periods ranging from 10 ms to 5 sec. In the Toyota Camry 2010, 39 of the 42 distinct messages are periodic. In the Dodge Ram Pickup 2010 in [5], all of the 55 distinct messages are periodic. While CAN message periodicity depends on the manufacturer and the model, the above examples suggest that periodic CAN messages are very common and even dominant on the CAN bus of commercial automobiles.

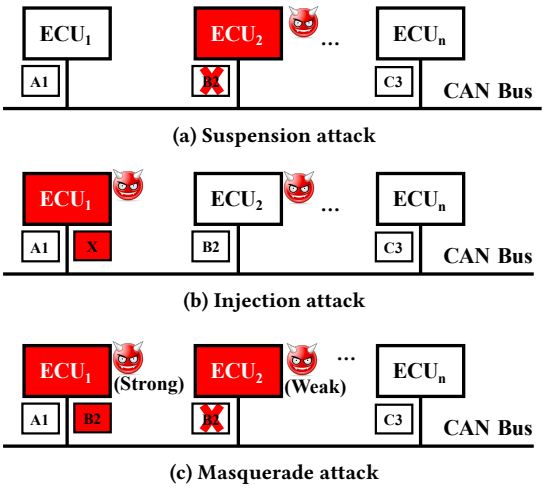

Figure 3: Three representative attacks on the CAN bus.

attack, the adversary will need to compromise two ECUs – one is weakly compromised (acting as the strong attacker who can only launch suspension attacks) and the other one is fully compromised (acting as the strong attacker who can both launch suspension and injection attacks). In the example in Figure 3(c), the adversary suspends the weakly compromised ECU_2 from transmitting message 0xB2 and uses the fully compromised ECU_1 to inject messages 0xB2 claiming to originate from ECU_2. As compared to the suspension and injection attack, it is much more difficult to detect the stealthy masquerade attack.

4 TACAN

In this section, we provide details on the TACAN architecture (Section 4.1) and introduce our transmitter authentication protocol (Section 4.2). We then present three covert channels for transmitting authentication messages: 1) IAT-based (Section 4.3), 2) offset-based (Section 4.4), and 3) LSB-based (Section 4.5).

4.1 TACAN architecture

As shown in Figure 4, TACAN consists of in-vehicle ECUs and a trusted MN connected to the same CAN bus. We assume that the MN is pre-installed by the manufacturer during production and requires direct physical access by authorized parties (e.g., an authorized repairs shop) to prevent potential tampering and compromises. We further assume that the deployed covert channels are pre-configured during production or re-configured during maintenance, so that the one-way communication of authentication information from ECUs to the MN can be successful established.

Similar to [14, 26, 33], a master key (MK) is assumed to be pre-shared between each ECU and the MN, which is stored in the TPM. Updating of MKs (e.g., when adding or replacing an ECU) should again require direct physical access by authorized parties to the involved ECUs. The procedure of key updating is outside the scope of this paper. During operation, Each ECU will generate a session key (SK) from the MK and a global counter and further use it to generate authentication messages. We describe the transmitter authentication protocol in more detail in the following section.

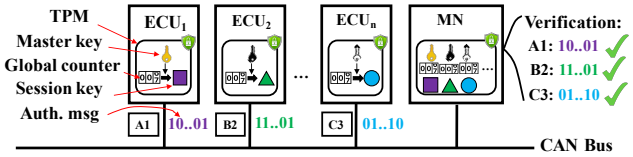

Figure 4: Illustration of TACAN architecture and the transmitter authentication protocol.

4.2 Transmitter Authentication Protocol

Inspired by the work in [26], the MN performs unidirectional authentication of each transmitting ECU. The parameters used by TACAN for the n ECUs are summarized as follows:

- The master key MK_i with $i \in \{1, ..., n\}$ is a pre-shared key between ECU_i and the MN, which is securely stored in the TPMs of both parties.
- The session key SK_i with $i \in \{1, ..., n\}$ is used to generate authentication messages for ECU_i.
- The local counter l_i with $i \in \{1, ..., n\}$ is an incremental value that stores the number of transmitted authentication messages for ECU_i. This value is contained in the authentication message.
- The global counter g_i with $i \in \{1, ..., n\}$ represents a value updated on specific circumstances (e.g., car ignition, l_i overflow) and it is used to generate SK_i.

We assume that both SK_i and g_i are stored in the TPM so that an attacker cannot tamper with them and launch replay attacks.

Session Key Generation. Each ECU_i stores its own master key MK_i, and the MN stores the master keys of all ECUs. The session key for the n ECUs is generated from MK_i and g_i as follows:

$$SK_i = HMAC(MK_i, g_i), \text{ for } i \in \{1, ..., n\},$$

where $HMAC(\cdot)$ refers to the Hash-based Message Authentication Code algorithm [19]. It is possible to use different hashing algorithms for HMAC (e.g., HMAC-SHA256) to meet the desired security requirements.

Every time SK_i is updated, l_i will be reset to zero. On the receiver side, the MN uses the same MK_i and g_i to compute the corresponding SK_i for ECU_i. Counter synchronization is performed when g_i is incremented. Since authentication message verification failures can be caused by de-synchronization of counters, a re-synchronization procedure may be performed. More details are provided in [26].

Authentication Message Generation. ECU_i first increments l_i and then computes the authentication message A_m as follows:

$$A_m = l_i || HMAC(SK_i, l_i),$$

where "||" denotes bit string concatenation. For the scope of our work, we assume that all the parameters (keys and counters) conceived for TACAN are binary values. As we can see, A_m is not related to any normal CAN message and merely serves as an identifier for ECU_i.

As for l_i, a 24-bit counter can last for 46+ hours for a 10-ms message even in per-message authentication, which is sufficient for our transmitter authentication protocol that transmits A_m at a much smaller frequency. Implementations of TACAN are free to use whichever hashing algorithm for HMAC and sizes of keys

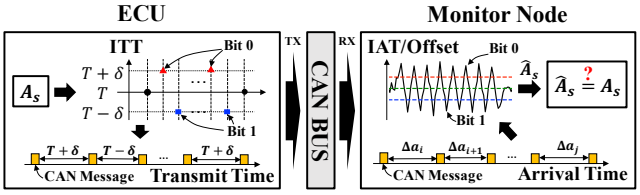

Figure 5: Illustration of timing-based covert channels.

that are deemed strong enough. Instead of transmitting the entire digest that are usually hundreds of bits long, the transmitter may truncate each digest to several bits to reduce the transmission time (e.g., using the least significant 8 bits or XORing all bytes together to create a condensed 8-bit version of the digest, as in [20, 31]).

4.3 IAT-Based Covert Channel

Figure 5 illustrates two timing-based covert channels for periodic CAN messages, in which the transmitting ECU embeds the authentication message into the ITTs of CAN messages, which can be extracted from the IATs or offsets by the MN. By verifying the received authentication message, the MN can authenticate the transmitter. In this section, we present the IAT-based covert channel.

Observations. According to our timing model in Eq. (1), the observed IAT between messages $(i-1)$ and i is given by

$$\Delta a_i = a_i - a_{i-1} = \frac{1}{1+S}\Delta t_i + (\eta_i - \eta_{i-1}). \tag{2}$$

Since almost all CAN messages are periodic (i.e., $\Delta t_i = T$), the IATs have a mean of $\mathbb{E}[\Delta a_i] = T/(1+S)$ and a variance of $\mathrm{Var}(\Delta a_i) = 2\sigma_\eta^2$. In practice, clock skews are usually very small (in the order of 100s of ppm). Hence, we have $\kappa = \mathbb{E}[\Delta a_i] \approx T$, where κ is considered as the reference level for IATs[2].

From Eq. (2), we can see that an amount of deviation δ in ITTs will lead to a corresponding change of $\delta/(1+S)$ in IATs, which may be easily observed by the receiver when variances in IATs are small. In the case of large variances in IATs, the receiver may compute the running average to smooth out the noise, that is,

$$\Delta\bar{a}[i] = \frac{1}{L}\sum_{j=0}^{L-1}\Delta a_{i+j} = \frac{1}{1+S}\left(\frac{1}{L}\sum_{j=0}^{L-1}\Delta t_{i+j}\right) + \frac{1}{L}(\eta_{i+L-1} - \eta_{i-1}),$$

where L is the window length. Through the running average, the variance of IATs can be reduced significantly by a factor of L^2. Note that due to the existence of $1/L$ in the bracket, the amount of deviation δ needs to be added to L consecutive ITTs in order to maintain the same level of changes in observed IATs.

As an example, we plot the IAT distributions of message 0x020 ($T = 0.01$ sec) from the Toyota dataset [8] in Figure 6. When $\delta = 0.02T = 2 \cdot 10^{-4}$ sec is added to or subtracted from IATs (to simulate operations at the transmitter assuming negligible clock skew effects), the three clusters (representing a bit 0 or a bit 1 or neither) cannot be separated from each other (Figure 6(a)), which may lead to bit errors at the receiver side. In contrast, with the running average of $L = 4$, the clusters are clearly distinguishable

[2]In the case of very large clock skews, the reference level κ needs to be calibrated offline from the training dataset.

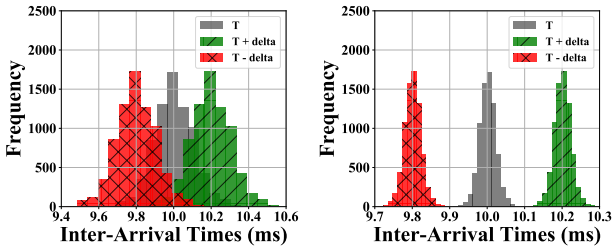

Figure 6: Example of IAT distributions of message ID=0x020 from the Toyota dataset (a) without running average ($L = 1$) and (b) with running average ($L = 4$).

(Figure 6(b)) and can be separated through thresholding. The above observations motivate our design of IAT-based covert channels.

Embedding A_m into ITTs. Given A_m of n_m bits, the transmitter first constructs an authentication frame A_f by inserting $n_s/2$ silence bits before and after A_m. The main purpose of silence bits (setting $\delta = 0$) is to maintain the reference level of IATs and signal the start and end of A_m. Hence, the total length of A_f is $n_f = n_m + n_s$. For instance, if $A_m = \texttt{0101}$ and $n_s = 2$, we have $A_f = \texttt{\_0101\_}$. Each bit i is modulated into L consecutive ITTs as follows,

$$\Delta t_j = \begin{cases} T+\delta, & \text{if } b_i = 0, \\ T-\delta, & \text{if } b_i = 1, \\ T, & \text{else,} \end{cases}$$

where $j \in [iL, (i+1)L)$.

Extracting A_m from IATs. On the receiver side, the MN records the arrival timestamps for the targeted messages and computes the IATs. It then performs running average with the right choice of L to obtain $\{\Delta\bar{a}[i]\}$. Since each bit is repeated for L consecutive ITTs, the receiver needs to sample $\{\Delta\bar{a}[i]\}$ every L values. Let the sampling offset be τ and the j-th sample be $\Delta\bar{a}[jL+\tau]$. If τ is correctly chosen, the total distance between each modulated sample and the reference level (κ) should be maximized, i.e.,

$$\tau^* = \arg\max_\tau \sum_j \left|\Delta\bar{a}[jL+\tau] - \kappa\right|.$$

Then the receiver can convert the sampled values to bits through thresholding as follows,

$$\hat{b}_j = \begin{cases} 0, & \text{if } \Delta\bar{a}[jL+\tau^*] > \Gamma_u, \\ 1, & \text{if } \Delta\bar{a}[jL+\tau^*] < \Gamma_l, \\ \_, & \text{else,} \end{cases} \tag{3}$$

where $\Gamma_l = \kappa - \delta/2$ and $\Gamma_u = \kappa + \delta/2$ are the lower and upper thresholds, respectively, $\kappa \approx T$, and "_" is the silence bit. After that, the output bits are concatenated and then split at the silence bits to obtain A_m.

Note that in order to establish a reliable IAT-based covert channel, the parameters including L and δ need to be pre-shared between the transmitting ECU and the MN, as assumed in Section 3.1. We leave online estimation of the covert channel parameters from the MN side as future work.

4.4 Offset-Based Covert Channel

The idea of the offset-based covert channel is very similar to the IAT-based covert channel (Figure 5). The main difference is that the former performs running average on IATs (including the added deviations) to smooth out the noise, while the later aims to accumulate the deviations to differentiate the modulated samples from noise. Our observations about clock offsets are as follows.

Observations. According to our timing model in Section 3.1 and Eq. (1), we have $t_i = \sum_{j=1}^{i} \Delta t_j$, $a_0 = \eta_0$, and $a_i = t_i/(1+S) + \eta_i$. Since the receiver only knows the nominal period (T) of the targeted message, it computes the observed clock offset as the difference between the expected elapsed time (up to message i) at the transmitter and the actual elapsed time at the receiver, i.e.,

$$\hat{O}_i = iT - (a_i - a_0) = iT - \frac{1}{1+S}\sum_{j=1}^{i} \Delta t_j + \eta_0 - \eta_i.$$

As we can see, if the transmitter adds δ to L consecutive ITTs ($\Delta t_j = T + \delta$ for $j = 1, ..., L$), the deviations will accumulate and lead to a decrease of $\delta L/(1+S)$ in $\hat{O}_i$. Hence, by monitoring the changes in $\hat{O}_i$, the receiver can determine the transmitted bits and extract A_m from clock offsets. The above observations motivate our design of offset-based covert channels.

Embedding A_m into ITTs. Different from the IAT-based covert channel (Section 4.3), the transmitter embeds each bit of A_f into L consecutive ITTs as follows,

$$\Delta t_j = \begin{cases} T - \delta, & \text{if } b_i = 0, \\ T + \delta, & \text{if } b_i = 1, \\ T, & \text{else,} \end{cases}$$

for $j \in [iL, (i+1)L)$, and

$$\Delta t_j = \begin{cases} T + \delta, & \text{if } b_i = 0, \\ T - \delta, & \text{if } b_i = 1, \\ T, & \text{else,} \end{cases}$$

for $j \in [iL+L/2, (i+1)L)$, where L is assumed to be an even integer. In other words, in order to transmit a bit 0/1, the transmitter adds $-\delta/\delta$ to the first $L/2$ ITTs and then subtracts $-\delta/\delta$ from the following $L/2$ ITTs so that the observed clock offset returns to the reference level after the transmission of each bit.

Extracting A_m from Offsets. On the receiver side, the monitor node records the arrival timestamps and processes the IATs in batches. Since each A_f has n_f bits and each bit is modulated into L consecutive ITTs, a total number of $N = n_f L$ consecutive IATs belong to the same A_f, where N is referred to as the batch size.

Denote the i-th IAT in the k-batch as $\Delta a_{k,i}$, where $1 \le i \le N$. Then the observed clock offset with respect to the beginning of the current batch up to the i-th IAT is

$$\hat{O}_k[i] = iT - \sum_{j=1}^{i} \Delta a_{k,j}.$$

Let $\kappa = (\max(\hat{O}_k[i]) + \min(\hat{O}_k[i]))/2$ be the midpoint (assuming that at least a bit 0 and a bit 1 are transmitted), which is considered as the reference level of clock offsets. Since clock skew is usually very small (100s of ppm) and we are computing batch-wise clock offsets, the impact of clock skew is small, and thus we assume that the κ is constant for the duration of a batch[3]. Since each bit affects L IATs, the receiver needs to sample $\{\hat{O}_k[i]\}$ every L values with a sampling offset of τ and obtain the j-th sample as $\hat{O}_k[jL+\tau]$. If τ is correctly chosen, then the total distance between each sample and the reference level should be maximized, i.e.,

$$\tau^* = \arg\max_{\tau} \sum_{j} |\hat{O}_k[jL+\tau] - \kappa|.$$

Then the receiver converts the sampled values to bits through the following threhsolding-based scheme,

$$\hat{b}_j = \begin{cases} 0, & \text{if } \hat{O}_k[jL+\tau^*] > \Gamma_u, \\ 1, & \text{if } \hat{O}_k[jL+\tau^*] < \Gamma_l, \\ \_, & \text{else,} \end{cases}$$

where $\Gamma_l = \kappa - \frac{1}{4}\delta L$ and $\Gamma_u = \kappa + \frac{1}{4}\delta L$ are the lower and upper thresholds, respectively, and "_" is the silence bit. The term of $\frac{1}{4}\delta L$ is due to the fact that δ is added to (or subtracted from) $L/2$ consecutive ITTs, and thus the maximum total deviation is $\pm\frac{1}{2}\delta L$. The midpoints between κ and $\kappa \pm \frac{1}{2}\delta L$ are chosen as thresholds. Eventually, A_m is extracted by concatenating all decoded bits and splitting them at the silence bits.

Impact on CAN bus schedulability. As we have shown, in both the IAT-based and offset-based covert channels, a certain amount of deviation is added to the ITTs. As a result, it may increase the worst-case response time of CAN messages, which is defined as the longest time from the initiating event (that puts the message in the transmission queue) occurring to the message being received by the nodes that require it. If we apply the schedulability analysis in [9] to TACAN, we can show that the effect of TACAN is equivalent to increasing the blocking delay by a constant amount of time (hundreds of μs) and increasing the message transmission time by a small percentage. Therefore, to achieve effective use of covert channels, TACAN parameters need to be experimentally obtained and fine tuned prior to deployment to ensure the schedulability of the CAN bus. A more detailed discussion is provided in Appendix A.2.

4.5 LSB-Based Covert Channel

In this section, we introduce a storage-based covert channel that embeds the authentication messages inside the LSBs of the data payload of normal CAN messages transmitted by an ECU, referred to as the LSB-based covert channel (Figure 7). Unlike the timing-based covert channels, the LSB-based covert channel can be applied to aperiodic CAN messages. For the scope of this work, we use the CAN data frames to develop our methodology.

Observations. In order to transmit an authentication message over the CAN bus, it is very common to make use of the existing fields of a CAN message, such as the data field [13, 14, 20] (at least one byte) and the extended ID field [14], or simply introduce additional CAN messages [13, 14]. In practice, however, if all bytes in the data field have been used or the CAN bus is already heavily loaded, then the existing approaches of exploiting CAN messages may disrupt the ECU's functionality or increase arbitration delays.

[3] In the case of very large clock skew (e.g., 1000s of ppm), there will be a linear trend due to clock skew in batch-wise clock offsets and thus the reference level may not be constant. We leave the detrending process of clock offsets as future work.

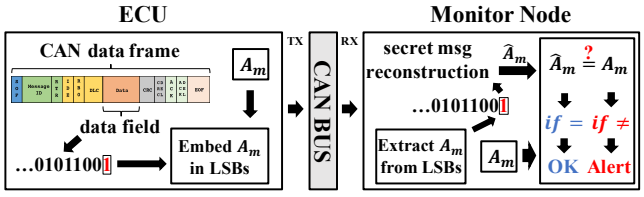

Figure 7: Illustration of LSB-based covert channel. The transmitting ECU embeds the authentication message into the LSBs of multiple normal CAN messages (with the same ID), which can be extracted and verified by the Monitor Node.

Different from existing schemes that attempt to authenticate messages, our transmitter authentication scheme aims to authenticate the transmitter instead of each message. Hence, authentication messages are transmitted much less frequently, which means that authentication bits may be spread over multiple CAN messages, using only a few bits from each CAN message. Moreover, having realized that certain CAN messages are used to carry sensor values and most of them are floating numbers, we may use the L LSBs (e.g., $L = 1$ or 2) for authentication purposes without causing significant degradation in accuracy. Taking the Toyota Camry as an example, there are at least 7 messages out of 42 that carry sensor values (e.g., wheel speeds, engine speed, vehicle speed, odometer, brake pressure, steering angle) [8]. We would expect more CAN messages that carry sensor values in newer automobiles. The above observations motivate our design of LSB-based covert channels.

Embedding A_m to LSBs. The embedding process is implemented as a sub-layer between the Application and the Data Link layers. For each message A_m (plus a known preamble to indicate the start and end of A_m), it substitutes the least significant L bits of the CAN message with the next L bits in A_m. No modification is needed if the L bits happen to be the same.

Extracting A_m from LSBs. On the receiver side, the MN extracts the L LSBs from the received CAN messages and reconstructs the authentication message. If the MN fails to verify the authentication message, it will raise an alert that indicates possible compromise of the transmitting ECU or malicious exploitation of the CAN bus.

As in the case of timing-based covert channels, we assume that the settings for the LSB-based covert channel are pre-shared between each ECU and the MN during production and updated during maintenance if necessary. More details about the embedding and extraction processes are provided in Appendix A.3.

5 SECURITY DISCUSSION

Compared to the existing message authentication schemes [14, 20, 26, 33] that attempt to verify the authenticity of each message, TACAN extracts and verifies authentication message embedded in the timing or LSBs of normal messages to authenticate the transmitting ECU and also serves the purpose of intrusion detection. Hence, TACAN can detect attacks that interrupt the transmission of normal CAN messages (e.g., suspension, injection, and DoS attacks), as well as attacks in which attackers fail to generate valid authentication messages (e.g., forgery, replay, and masquerade attacks). The above security properties provided by TACAN are independent of the CAN protocol and the contents of CAN messages.

Security Features. We summarize a list of features provided by TACAN as follows: 1) TACAN securely stores both master and session keys in the ECU's TPM to prevent being compromised by the adversary, 2) TACAN employs monotonic counters for generating session keys and authentication messages for each ECU, and 3) TACAN requires each ECU to continuously transmit unique authentication messages to enable real-time transmitter authentication.

Note that due to the third feature, if the attacker compromises an in-vehicle ECU but does not attempt to deceive TACAN (i.e., stop generating authentication messages), it will interrupt the continuous transmission of authentication messages and thus will be detected immediately. In addition, since basic attacks like suspension, injection, and DoS attacks can be easily detected, we focus on more sophisticated attacks in the rest of this section that actively attempt to evade TACAN, including the forgery attack, the replay attack, and the masquerade attack.

Detecting Forgery Attacks. In the forgery attack, the adversary has already compromised an in-vehicle ECU that is protected by TACAN and attempts to generate valid authentication messages that can be verified by the MN in order to evade the detection of TACAN. Since our adversary model (Section 3.2) assumes that the attacker has no access to the TPM of the compromised ECU, the attacker has to forge a valid digest for each local counter value without the session key. With a condensed digest of M bits, the probability of a successful forgery is $1/2^M$. For example, when $M = 8$, this probability is $1/2^8 \approx 0.4\%$. Repeated forgeries will be prevented due to the use of monotonic counters.

Detecting Replay Attacks. A replay attacker has infiltrated the CAN bus and attempts to replay previously transmitted authentication messages of the targeted ECU with the hope of passing the verification process at the MN. It is easy to see that such attempts will be detected by TACAN due to the use of monotonic counters.

Detecting Masquerade Attacks. As mentioned in Section 3.2, a masquerade attack (including the more sophisticated cloaking attack [28]) requires in-vehicle ECUs to be weakly and/or fully compromised. As a result, TACAN will force the attacker to perform a forgery or replay attack, not only for the compromised ECU itself, but also for the ECU the attacker attempts to masquerade as. Therefore, a masquerade attack will be detected by TACAN.

6 EVALUATION

In this section, we implement the proposed covert channels on the testbed and report their performance in terms of the throughput and bit error ratio using the datasets collected from two real vehicles, a 2010 Toyota Camry [8] and a 2016 Chevrolet Camaro (University of Washington EcoCAR) [10]. Testbed validation is in Section 6.1, then the evaluations of covert channels IAT-based, offset-based, and LSB-based are in Section 6.2, Section 6.3, and Section 6.4, respectively.

6.1 Testbed Validation

As shown in Figure 8, our EcoCAR testbed consists of the UW EcoCAR and two testbed ECUs, which are connected via the On-Board Diagnostics (OBD-II) port. The UW EcoCAR hosts 8 stock

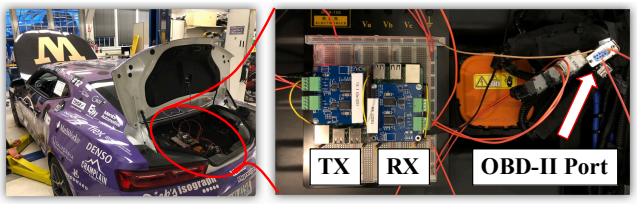

Figure 8: EcoCAR testbed. Two Raspberry Pi-based ECUs are connected to the OBD-II port at the back of the EcoCAR, one as the transmitter and the other one as the receiver.

ECus and two experimental ECUs. There are a total of 2500+ messages using 89 different messages IDs are exchanged on the CAN bus every second. Each ECU consists of a Raspberry Pi 3 and a PiCAN 2 board (using a MCP2515 CAN controller and a MCP2551 CAN transceiver). The SocketCAN [29] library is used to enable the interaction between the added ECUs and the UW EcoCAR.

In order to demonstrate the proposed covert channels, we implemented them in Python in the transmitter and collected the CAN data traces using the receiver. Since the LSB-based covert channel is expected to have a very small bit error ratio, we focus on the timing-based (IAT-based and offset-based) covert channels. As an example, we set $n_m = 36$ bits (length of authentication message) with alternating bit values and $n_s = 4$ bits (number of silence bits). Hence, each authentication frame is $n_f = 40$ bits long. In order to avoid any conflict with existing messages, we chose a non-existent message ID of 0x180 and set the message period $T = 10$ ms and $\delta = 0.2$ ms (2% of T). While a non-existent message ID was used in our experiments, we assumed the Raspberry Pi-based ECU as a stock ECU that transmits messages with ID=0x180 and that is capable of modifying the timing of message ITTs. We would like to emphasize that TACAN does not require a new message ID or additional CAN messages to implement the proposed covert channels.

Figure 9 provides an example of observed IATs in the IAT-based covert channel. As we can see, without running average, the modulated IATs are noisy, and thresholding at the receiver side may lead to many possible bit errors (Figure 9(a)). In contrast, the running average process can effectively reduce IAT variations, making the peaks and valleys of the modulated IATs clearly stand out, which indicates a smaller probability of bit errors (Figure 9(b)).

An example of observed offsets in the offset-based covert channel is provided in Figure 10. As we can see, unlike the IAT-based covert channel, increasing L in the offset-based covert channel effectively increases the accumulated amount of deviations in offsets at the receiver side and thus reduces the bit error probability. In the rest of this section, we evaluate each covert channel in more detail.

6.2 Evaluation of IAT-Based Covert Channel

In this section, we evaluate the performance of IAT-based covert channels in terms of throughput and bit error ratio.

Throughput. Given n_m bits for A_m, n_s silence bits, and a window length of L, the time it takes for a CAN message with period T to transmit an authentication frame A_f is $T_f = (n_m + n_s)LT$, which increases linearly as function of L and T. Then the throughput or rate for transmitting A_m is $r_m = n_m/T_f$ bits per second (bps). For

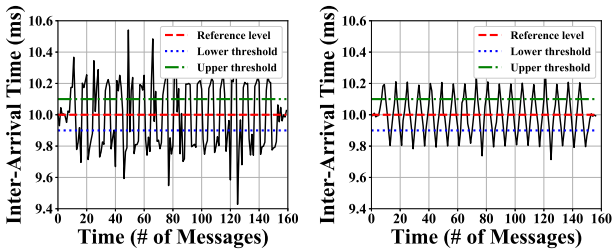

Figure 9: Example of observed IATs in IAT-based covert channels (a) without and (b) with running average ($L = 4$).

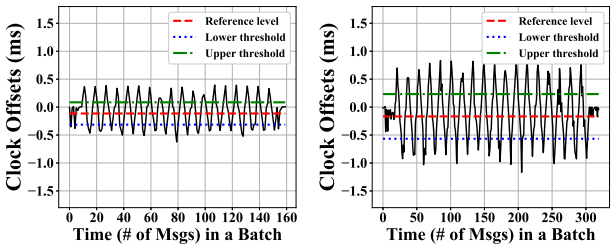

Figure 10: Example of observed offsets in offset-based covert channels with (a) $L = 4$ and (b) $L = 8$.

example, suppose that $n_m = 36$, $n_s = 4$, and $L = 4$. Then we have $T_f = 1.6$ sec for $T = 0.01$ sec and $r_m = 36/1.6 = 22.5$ bps.

Bit Error Ratio. The bit error ratio is defined as the number of bit errors divided by the total number of bits transmitted in a given time interval. In order to evaluate the bit error ratio of IAT-based covert channels on real vehicles, we collected data for six representative messages (including message 0x180 that was transmitted from the testbed ECU) with different ID levels, periods, and noise levels from the Toyota Camry [8] and the UW EcoCAR [10], as shown in Table 2. The same EcoCAR dataset was used in [35]. Note that IAT noise is quantified in terms of the standard deviation (normalized by the period) and the range (the difference between the maximum IAT and the minimum IAT, normalized by the period).

Table 2: Selected set of representative messages.

Msg ID	Period (ms)	Standard dev. (Normalized)	Range (Normalized)	Source
0x020	10	1.1%	10.2%	Toyota
0x224	30	0.9%	4.8%	Toyota
0x0D1	10	2.7%	51.5%	EcoCAR
0x180	10	1.7%	30.1%	EcoCAR
0x185	20	1.3%	22.6%	EcoCAR
0x22A	100	1.2%	6.4%	EcoCAR

In this experiment, we first preprocess the data trace of each message to fill in any missing messages. We set $n_m = 36$ bits and $n_s = 4$ bits, and thus each frame is 40 bits long. Note that the choices of n_m and n_s are not critical here, since we focus on the bit error ratio instead of the frame error ratio. Out of 100 frames ($100n_m$ bits in total), the number of bit errors ($\hat{b}_j \neq b_j$) is recorded and divided

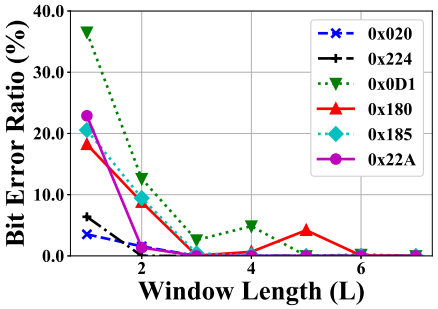

Figure 11: Bit error ratios of IAT-based covert channel for different CAN messages as a function of window length (L).

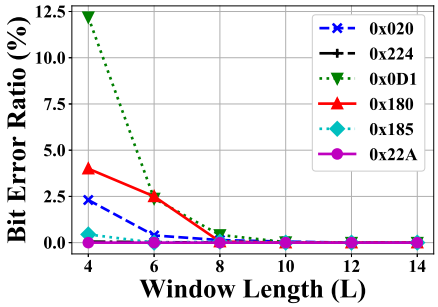

Figure 12: Bit error ratios of offset-based covert channels for different CAN messages as a function of window length (L).

by the number of transmitted bits to obtain the bit error ratio. The amount of deviation δ is fixed to 2% of the message period (T). Since clock skews are small, we directly add δ to IATs to simulate the effect of adding the same amount to ITTs at the transmitter side.

As illustrated in Figure 11, the performance of IAT-based covert channel varies a lot among different messages, depending on their noise levels. In general, the bit error ratios are very high with $L = 1$, but they quickly drop to 0 as L increases for all messages, which demonstrates the effectiveness of running average. Moreover, we observe that messages with a large IAT range (e.g., 0x180, 0x0D1 and 0x185) tend to require a large L value to establish a reliable IAT-based covert channel. With $L = 6$, all messages have a bit error ratio of less than 0.1%.

6.3 Evaluation of Offset-Based Covert Channel

Since the throughput analysis of offset-based covert channels is the same with that of IAT-based covert channels, we focus on the bit error performance in this experiment using the same setup in Section 6.2. Results are provided in Figure 12.

As we can see, the performance of offset-based covert channels again depends on the characteristics of individual messages. When $L = 4$, the bit error ratio is less than 1% for messages 0x224, 0x185, and 0x22A. As L increases, added deviations accumulate at the receiver side, thus effectively reducing the bit error ratio. With $L = 8$, the bit error ratio has dropped to less than 0.42% for all messages. As compared to the IAT-based covert channel, the offset-based covert channel is less efficient and requires a larger L value. This is mainly because it adds deviations to only the first half of L consecutive ITTs, and the second half is used only for the purpose of maintaining the reference level for clock offsets, which contributes very little in differentiating the modulated samples from the noise.

6.4 Evaluation of LSB-Based Covert Channel

In this section, we evaluate the performance of the LSB-based covert channel in terms of throughput, bit error ratio, and accuracy loss.

Throughput. Given n_m bits for A_m and n_s bits for the start sequence, it takes $T_f = n_f T/L = (n_m + n_s)T/L$ sec to transmit an authentication frame A_f using L LSBs of a CAN message with period T. Then the throughputs for transmitting A_f and A_m are $r_f = n_f/T_f$ and $r_m = n_m/T_f$, respectively. For example, suppose that $n_s = 36$, $n_s = 4$, $T = 0.01$ sec and $L = 2$. Then we have $T_f = 0.2$ sec, $r_f = 40/0.2 = 200$ bps, and $r_m = 36/0.2 = 180$ bps.

Bit Error Ratio. Since the LSB-based covert channel embeds the authentication message into the data payload of normal CAN messages, its bit error ratio will be as small as that of the CAN bus itself. According to [11], the bit error ratios of CAN in normal environments (factory production line) and aggressive environments (two meters away from a high-frequency arc-welding machine) are 3.1×10^{-7}% and 2.6×10^{-5}%, respectively.

Accuracy Loss. As the LSB-based covert channel modifies the LSBs in the data fields of CAN data frames, such modification will lead to accuracy loss of sensor values. With changes of one LSB ($L = 1$), the accuracy loss of the same scale with the resolution (or the discretization error) of that sensor value. Increasing L will result in larger throughput and accuracy loss. Hence, manufacturers will need to assess the impact of accuracy loss in CAN data on the functionality and safety, as well as trade off the accuracy loss against throughput when deploying the LSB-based covert channel. It is important to highlight that for periodic messages that cannot tolerate accuracy loss, manufacturers may deploy IAT-based and offset-based covert channels instead.

In order to demonstrate the feasibility of the proposed LSB-based covert channel, we consider two CAN messages: 1) the wheel velocity value based on the publicly available Toyota dataset [8] and 2) the engine coolant temperature value that we identified through reverse engineering from the EcoCAR dataset [10]. In our experiments, we set L to 1 or 2 and quantify the accuracy loss in terms of *the maximum error* (using the original values as ground truth). Note that we intentionally keep $L \leq 2$ in order to avoid significant distortions to the underlying sensor values or jeopardize the functionality of the receiving ECU.

As illustrated in Figure 13 and highlighted in the magnified box, the maximum error introduced to wheel velocity is 0.01 km/h for $L = 1$ and 0.03 km/h for $L = 2$, which is very insignificant. As for the engine coolant temperature as shown in Figure 14, the maximum error is 1 $^{\circ}C$ for $L = 1$ and 3 $^{\circ}C$ for $L = 2$, which are still moderate.

7 CONCLUSION AND FUTURE WORK

In this paper, we introduced TACAN, a transmitter authentication scheme using covert channels for the CAN bus, that allows a MN

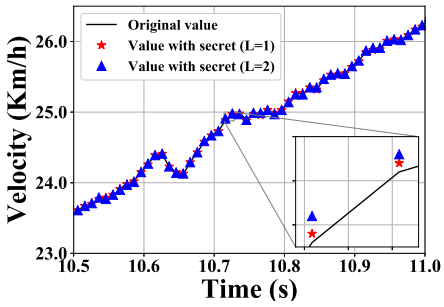

Figure 13: Authentication message embedded in Toyota wheel velocity data with $L = 1$ and with $L = 2$.

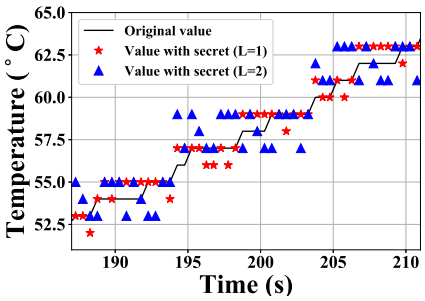

Figure 14: Authentication message embedded in EcoCAR engine coolant temperature data with $L = 1$ and with $L = 2$.

to verify the authenticity of the transmitting ECU. We developed IAT-based, offset-based, and LSB-based covert channels to communicate the authentication information between ECUs and the MN without introducing protocol modifications or traffic overheads. We provided a security discussion for TACAN and demonstrated the proposed covert channels through testbed validation and experimental evaluation. Our future work will include experimental evaluation of TACAN under various attacks against the CAN bus. In addition, we will also improve the throughput of the proposed covert channels and explore hybrid covert channel schemes.

ACKNOWLEDGMENTS

We thank the EcoCAR 3 project student lead Aman V. Kalia for helping us with the EcoCAR testbed. This work was supported in part by NSF grant CNS-1446866, ONR grants N00014-16-1-2710 and N00014-17-1-2946, and ARO grant W911NF-16-1-0485. Views and conclusions expressed are that of the authors and not be interpreted as that of the NSF, ONR or ARO.

REFERENCES

[1] M. Bojarski et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).
[2] Bosch. 1991. CAN Specification Version 2.0. (1991).
[3] S. Checkoway et al. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *Proc. of the 20th USENIX Conf. on Security (SEC'11)*. USENIX Association, Berkeley, CA, USA, 77–92.
[4] K-T Cho and K. G Shin. 2016. Error handling of in-vehicle networks makes them vulnerable. In *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 1044–1055.
[5] K-T Cho and K. G. Shin. 2016. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In *Proc. of 25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX, 911–927.
[6] K-T Cho and K. G. Shin. 2017. Viden: Attacker Identification on In-Vehicle Networks. In *Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. 1109–1123.
[7] W. Choi et al. 2018. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Trans. Inf. Forensics Security* (2018).
[8] J. Daily. 2010. Interpreting the CAN data for a 2010 Toyota Camry. http://tucrrc.utulsa.edu/ToyotaCAN.html. (2010). Accessed: 2019-1-22.
[9] R. I Davis et al. 2007. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems* 35, 3 (2007), 239–272.
[10] UW EcoCAR3. 2019. University of Washington EcoCAR 3. http://ecocar3.org/washington/about-us/. (2019). Accessed: 2019-1-22.
[11] J. Ferreira et al. 2004. An experiment to assess bit error rate in CAN. In *Proc. of 3rd Int. Workshop of Real-Time Networks (RTN2004)*. 15–18.
[12] Trusted Computing Group. 2018. *TCG TPM 2.0 Automotive Thin Profile. Specification Version 1.01. Revision 15.* Technical Report.
[13] B. Groza et al. 2012. Libra-can: a lightweight broadcast authentication protocol for controller area networks. In *Proc. of Int. Conf. on Cryptography and Network Security*. Springer, 185–200.
[14] A. Hazem and H. Fahmy. 2012. LCAP – A lightweight CAN authentication protocol for securing in-vehicle networks. In *Proc. of 10th Embedded Security in Cars Conference (ESCAR), Berlin, Germany*, Vol. 6.
[15] T. Hoppe et al. 2008. Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures. In *Proc. of the 27th Int. Conf. on Computer Safety, Reliability, and Security (SAFECOMP '08)*. 235–248.
[16] ISO. 2015. *International Standard ISO 11898-1 Road Vehicles-Controller Area Network (CAN), Part 1 Data Link Layer and Physical Signaling.*
[17] M. Kneib and C. Huth. 2018. Scission: Signal Characteristic-Based Sender Identification and Intrusion Detection in Automotive Networks. In *Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security*. ACM, 787–800.
[18] K. Koscher et al. 2010. Experimental Security Analysis of a Modern Automobile. In *Proc. of the 2010 IEEE Security Privacy (SP '10)*. 447–462.
[19] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. 1997. *HMAC: Keyed-hashing for message authentication.* Technical Report.
[20] R. Kurachi et al. 2014. CaCAN-centralized authentication system in CAN (Controller Area Network). In *Proc. of Embedded Security in Cars (ESCAR 2014)*.
[21] C.-W. Lin et al. 2012. Cyber-security for the Controller Area Network (CAN) communication protocol. In *Proc. of 2012 Int. Conf. on Cyber Security*. IEEE.
[22] C. Miller and C. Valasek. 2015. Remote exploitation of an unaltered passenger vehicle. In *Black Hat USA*.
[23] D. L. Mills. 1992. *Network Time Protocol (Version 3): Specification, Implementation and Analysis.* Technical Report. RFC 1305.
[24] P. S. Murvay and B. Groza. 2014. Source Identification Using Signal Characteristics in Controller Area Networks. In *IEEE Signal Processing Letters*, Vol. 21. 395–399.
[25] M. Müter and N. Asaj. 2011. Entropy-based anomaly detection for in-vehicle networks. In *2011 IEEE Intelligent Vehicles Symposium (IV)*. 1110–1115.
[26] A.-I. Radu and F. D Garcia. 2016. LeiA: A lightweight authentication protocol for CAN. In *European Symp. on Research in Computer Security*. Springer, 283–300.
[27] S. U. Sagong, X. Ying, L. Bushnell, and R. Poovendran. 2018. Exploring Attack Surfaces of Voltage-Based Intrusion Detection Systems in Controller Area Networks. In *ESCAR Europe 2018*.
[28] S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran. 2018. Cloaking the clock: emulating clock skew in controller area networks. In *Proc. of the 9th ACM/IEEE Int. Conf. on Cyber-Physical Systems (ICCPS'18)*. 32–42.
[29] SocketCAN. 2018. Linux-CAN/SocketCAN user space applictions. https://github.com/linux-can/can-utils. (2018). Accessed: 2018-10-13.
[30] Olivia Solon. 2016. Team of hackers take remote control of Tesla Model S from 12 miles away. https://www.theguardian.com. (2016). Accessed: 2018-10-06.
[31] C. Szilagy and P. Koopman. 2008. A flexible approach to embedded network multicast authentication. (2008).
[32] J. M Taylor et al. 2017. Enhancing integrity of modbus TCP through covert channels. In *Proc. of Inf. Conf. on Signal Process. and Commun. Syst. (ICSPCS)*.
[33] A. Van Herrewege et al. 2011. CANAuth – a simple, backward compatible broadcast authentication protocol for CAN bus. In *ECRYPT Workshop on Lightweight Cryptography*.
[34] A. M Wyglinski et al. 2013. Security of autonomous systems employing embedded computing and sensors. *IEEE micro* 33, 1 (2013), 80–86.
[35] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, and R. Poovendran. 2019. Shape of the Cloak: Formal Analysis of Clock Skew-Based Intrusion Detection System in Controller Area Networks. *IEEE Trans. Inf. Forensics Security* (2019).
[36] S. Zander et al. 2007. A survey of covert channels and countermeasures in computer network protocols. *IEEE commun. Surveys Tut.* 9, 3 (2007), 44–57.
[37] S. Zander et al. 2008. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *Proc. of the 17th Conf. on Security Symp.* 211–225.

A APPENDIX

A.1 CAN Frame

As illustrated in Figure 15, each CAN frame or message has a set of predefined fields, including the Start of Frame (SOF) field, the Arbitration field (including a 11-bit message ID for the base frame format or a 29-bit message ID for the extended frame format), the Control field, the Data field (8-64 bits), the CRC field, the ACK field, and the End of Frame (EOF) field.

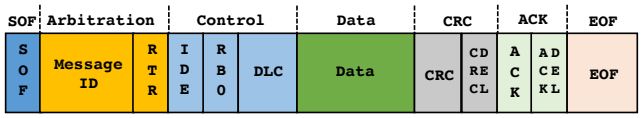

Figure 15: Illustration of CAN frame structure.

A.2 Impact on CAN Bus Schedulability

In order to understand the impact of TACAN on CAN bus schedulability, we apply the schedulability analysis in [9]. For the ease of discussion, we provide a summary of definitions as below. The same notations in [9] are used for consistency.

- m: Message priority (used interchangeably with message ID).
- C_m: Transmission time, defined as the longest time that the message can take to be transmitted.
- J_m: Queuing jitter, defined as the longest time between the initiating event and the message being queued, ready to be transmitted on the bus.
- w_m: Queuing delay, defined as the longest time that the message can remain in the CAN controller slot or device driver queue, before commencing successful transmission on the bus.
- T_m: Message period, defined as the minimum inter-arrival time of the event that triggers queueing of the message. Such events may occur strictly periodically with a period of T_m or sporadically with a minimum separation of T_m.
- R_m: Worst-case response time of message m, defined as the longest time from the initiating event occurring to the message being received by the nodes that require it.
- D_m: Hard deadline, defined as the maximum permitted time from occurrence of the initiating event to the end of successful transmission of the message.

A message is *schedulable* if and only if its worst-case response time is no greater than its deadline ($R_m \leq D_m$). A CAN bus is schedulable if and only if all messages on the CAN bus are schedulable.

As per [9], we have $C_m = (80 + 10s_m)\tau_{bit}$ (including bit stuffing), where s_m is the number of data bytes and τ_{bit} is the transmission time of a single bit. For a 8-byte message on a 500 kbps CAN bus, we have $\tau_{bit} = 2\ \mu$s and $C_m = 320\ \mu$s. While CAN nodes typically have separate clock sources, all the timing quantities (e.g., message jitters, bit times, message periods, and deadlines) that derived from node clocks will be converted to real-time.

The message m's worst-case response time R_m is given by

$$R_m = J_m + w_m + C_m, \tag{4}$$

and the queuing delay w_m consists of two elements:

- *Blocking* B_m, due to lower priority messages being transmitted when message m is queued, and
- *Interference* due to higher priority messages which may win arbitration and be transmitted in preference to message m.

The blocking delay B_m is given by $B_m = \max_{k \in lp(m)}(C_k)$, where $lp(m)$ is the set of messages with lower priority than m.

In order to analyze the worst-case response times, it is important to characterize the *busy period*, in which all messages of priority m or higher, queued strictly before the end of busy period, are transmitted during the busy period. The *maximal* busy period begins with a so-called *critical instant* where message m is queued simultaneously with all higher priority messages and then each of these higher priority messages is subsequently queued again after the shortest possible time interval.

For simplicity, we assume that only one instance of message m is transmitted during a priority level-m busy period. In this case, the worst-case queuing delay is given by:

$$w_m = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m + J_k + \tau_{bit}}{T_k} \right\rceil C_k. \tag{5}$$

Since the right hand side is a monotonic non-decreasing function of w_m, Eq. (5) can be solved using the following recurrence relation,

$$w_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \right\rceil C_k. \tag{6}$$

A suitable starting value is $w_m^0 = B_m$, and the recurrence relation iterates until, either $J_m + w_m^{n+1} + C_m > D_m$, i.e., the message is not schedulable, or $w_m^{n+1} = w_m^n$, in which case the worst-case response of message m is given by $J_m + w_m^{n+1} + C_m$.

In order to apply the above schedulability analysis to TACAN, we define $T'_m = T_m - \delta$. Since TACAN adds at most δ to each ITT, which can be considered as part of the queuing delay, we have $J'_m = J_m + \delta$. Assume that $\delta = 0.02T_m$ and all messages employ the timing-based covert channel. By substituting $T'_m = 0.98T_m$ and $J'_m = J_m + 0.02T_m$ into Eq. (6), we have

$$\begin{aligned} w_m'^{n+1} &= B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m'^n + (J_k + 0.02T_k) + \tau_{bit}}{0.98T_k} \right\rceil C_k \\ &\approx B'_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m'^n + J_k + \tau_{bit}}{T_k} \right\rceil C'_k, \end{aligned} \tag{7}$$

where

$$B'_m = B_m + \sum_{\forall k \in hp(m)} \left(\frac{0.02}{0.98} C_k \right), \tag{8}$$

and

$$C'_k = \frac{1}{0.98} C_k = 1.02C_k. \tag{9}$$

Therefore, the effect of TACAN on schedulability is equivalent to increasing the blocking delay by a constant amount of time and increasing the message transmission time by a certain percentage. For example, if we assume 45 messages with higher priority (half messages in the EcoCAR), the increase in the blocking delay and message transmission time is 294 μs and 6.5 μs ($C_m = 320\ \mu$s), respectively. By solving Eq. (7), we can compute the corresponding worst-case response time of message m with TACAN. Hence, to achieve effective use of covert channels, TACAN parameters need

Algorithm 2: Extracting A_m from LSBs

Input : A_m, L
$A_f \leftarrow \alpha||A_m$; $i \leftarrow 0$;
while $i < length(A_f)$ **do**
  Receive d_{field} from the Data Layer layer;
  // Process d_{field} in the Application layer;
  $\hat{A}_f[i : i + L - 1] = bin(d_{field}(\gamma))_L$;
  if $\hat{A}_f[i : i + L - 1] \neq A_f[i : i + L - 1]$ **then**
    Alert = True;
  $i \leftarrow i + L$;
return;

to be experimentally obtained and fine tuned prior to deployment to ensure the schedulability of the CAN bus.

A.3 LSB-Based Covert Channel Algorithms

In this section, we describe the authentication message embedding and extraction algorithms for the LSB-Based covert channel methodology. The related notations are summarized in Table 3.

Table 3: Notation of LSB-based covert channel algorithms.

Notation	Description
α	Start sequence
A_m	Authentication (secret) message
A_f	Authentication frame
$d_{field}(\cdot)$	Data field of CAN data frame
γ	Selected bytes (value)
$bin(\cdot)_L$	Least significant L bits

The embedding procedure is described in Algorithm 1. For each A_m, a known preamble (a particular bit sequence) is first appended to the beginning of A_m that allows the receiver to determine the start of A_m. We assume that the authentication messages are generated and transmitted continuously. Hence, the start of the next A_m indicates the end of the previous one. Whenever new data field content d_{field} is received from the upper (Application) layer, the least significant L bits of selected values in d_{field} (denoted as $bin(d_{field}(\gamma))_L$) are compared with the next L bits of A_f: if the bits of interest are different then the substitution occurs, otherwise d_{field} is not modified. The same process is then repeated for each new authentication sequence.

Algorithm 1: Embedding A_m to LSBs

Input : A_m, L
$A_f \leftarrow \alpha||A_m$; $i \leftarrow 0$;
while $i < length(A_f)$ **do**
  Receive d_{field} from the upper (Application) layer;
  if $bin(d_{field}(\gamma))_L \neq A_f[i : i + L - 1]$ **then**
    $bin(d_{field}(\gamma))_L = A_f[i : i + L - 1]$;
  $i \leftarrow i + L$;
  Send d_{field} to the lower (Data Link) layer;
return;

The extracting procedure is described in Algorithm 2. When the MN receives a targeted CAN message, it filters for the Data frame and extracts from the d_{field} the L LSBs. Once a valid start sequence is detected, the MN starts storing the bits in $\hat{A}_f$ and comparing the newly arrived bits against the expected A_f. In this way, it is possible to check possible inconsistency of the authentication sequence received also before the complete reception of A_f to shorten the verification time. If the actual value stored in $\hat{A}_f$ is not equal to part of the bits in A_f, an alert is raised highlighting a possible compromise of specific ECU or malicious exploitation of the CAN bus. Once the A_f is entirely received, the MN starts to extract the next authentication message.