Cloaking the Clock: Emulating Clock Skew in Controller Area Networks

Sang Uk Sagong*, Xuhang Ying*, Andrew Clark†, Linda Bushnell*, and Radha Poovendran*
* Department of Electrical Engineering, University of Washington, Seattle, WA 98195.
† Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609.
Email: {sagong, xhying, lb2, rp3}@uw.edu, aclark@wpi.edu

Abstract— **Automobiles are equipped with Electronic Control Units (ECUs) that communicate via in-vehicle network protocol standards such as the Controller Area Network (CAN). These protocols were designed under the assumption that separating in-vehicle communications from external networks is sufficient for protection against cyber attacks. This assumption, however, has been shown to be invalid by recent attacks in which adversaries were able to infiltrate the in-vehicle network. Motivated by these attacks, intrusion detection systems (IDSs) have been proposed for in-vehicle networks that attempt to detect attacks by exploiting physical properties such as clock skew of an ECU. In this paper, we propose the cloaking attack, an intelligent masquerade attack in which an adversary modifies the timing of transmitted messages to match the clock skew of a targeted ECU. The attack leverages the fact that, while the clock skew is a physical property of each ECU that cannot be changed by the adversary, the estimation of the clock skew by other ECUs is based on the timing of network traffic, which, being a cyber component only, can be modified by an adversary. We implement the proposed cloaking attack and test it on two IDSs, namely, the current state-of-the-art IDS and its adaptation to the widely-used Network Time Protocol (NTP). We implement the cloaking attack on two hardware testbeds, a prototype and a real vehicle, and show that it is able to deceive both IDSs. We also introduce a new metric called the Maximum Slackness Index to quantify the effectiveness of a clock skew-based IDS in detecting masquerade attacks when the adversary is unable to precisely match the clock skew of the targeted ECU.**

Index Terms—**CPS Security, Controller Area Network, Intrusion Detection System, Masquerade Attack, Clock Skew**

I. Introduction

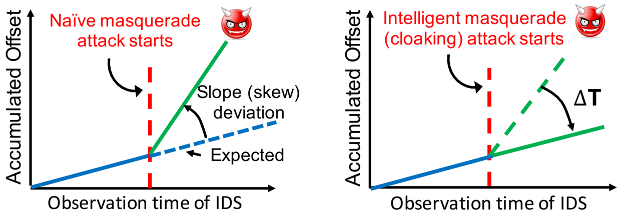
Contemporary automobiles are equipped with electronic control units (ECUs) for various functionalities such as vehicle maneuverability, fuel efficiency, and heat, ventilation, and air conditioning. In order to operate these ECUs properly, the information among ECUs is exchanged via in-vehicle network protocols. In-vehicle network protocols are based on standards such as the Controller Area Network (CAN), which were developed for closed networks that are isolated from the external environment. Based on the closed network assumption, in-vehicle protocols were not designed for cyber security, and in particular do not provide encryption or message authentication.

Connected vehicles, however, have an increasingly large and diverse array of outward-facing components in order to provide safety, navigation, and entertainment, which violate the assumption of a closed operating environment. These externally accessible interfaces leave connected vehicles vulnerable to attacks in which an adversary compromises outward-facing ECUs (e.g., CD players or cellular radio), gains access to the CAN bus, and then blocks messages sent by other ECUs (denial-of-service attack) or sends spoofed messages that claim to be originated from legitimate ECUs such as steering or engine control (masquerade attack) [1]. Such attacks can create spurious alarms to the driver, disable brakes, or cause the vehicle to accelerate uncontrollably, causing serious safety risks to passengers, pedestrians, and other vehicles [2]–[4].

The cyber vulnerabilities of connected vehicles have motivated development of intrusion detection systems (IDSs) for in-vehicle networks [3], [5]–[7]. Due to the lack of cryptographic integrity checks, such IDSs rely on physical invariants of the system. For instance, ECUs usually transmit messages of fixed length at fixed frequencies, and the message contents do not vary drastically over time. In [3], [6], mechanisms for detecting DoS attacks by exploiting message periodicity were proposed. Detection mechanisms based on network entropy were proposed in [7]. As pointed out in [5], however, entropy-based IDSs may be ineffective against intelligent adversaries who mimic the structure and frequency of legitimate traffic.

An IDS for detecting such intelligent attacks was proposed in [5], based on the following principles. Each ECU on the CAN bus has a different hardware clock, which has a distinct clock speed due to variations in the clock's hardware crystal, a property referred to as clock skew [8], [9]. Since all process clocks in an ECU are derived from the hardware clock, they are affected by the clock skew as a consequence. In particular, the inter-transmission times of messages that are periodically transmitted by an ECU will be impacted by its clock skew. If a naive adversary injects the spoofed periodic message from an ECU that is different from the spoofed ECU, the difference in the clock skew will affect the inter-transmission times. Hence, an ECU that receives periodically transmitted messages can estimate the clock skew of the transmitting ECU based on message inter-arrival times. The IDS located at the receiving ECU then detects an attack when an unexpected change in the estimated clock skew occurs (Fig. 1(a)).

In this paper, we analyze IDSs that use the clock skew for detection. Our key observation is that an intelligent adversary who realizes that the IDS at the receiver ECU computes the clock skew using message inter-arrival times can manipulate the inter-transmission times to match the clock skew of the targeted ECU and avoid detection. We refer to this intelligent



(a) (b)

Fig. 1: Clock skew estimated by the IDS at the receiver. (a) An IDS tracks the clock skew of the transmitter and detects deviations due to naive masquerade attacks. (b) An intelligent masquerading adversary adds a delay ΔT to message inter-transmission times, in order to emulate the clock skew of the targeted ECU and bypass the IDS.

masquerade attack as the *cloaking attack*, as illustrated in Fig. 1(b). These results show that, while physical system properties such as clock skew may be helpful in providing security assurances and detecting attacks, intelligent adversaries may still evade detection when physical properties are filtered or mediated through compromised cyber components. Throughout this paper, we make the following specific contributions:

- We propose the cloaking attack, in which an adversary adjusts message timing and cloaks its clock to match the targeted ECU's clock skew in order to avoid detection.
- We analyze the effectiveness of the proposed cloaking attack against two IDSs, including a state-of-the-art IDS and its adaptation to the Network Time Protocol (NTP).
- We introduce a new metric called Maximum Slackness Index (MSI) to quantify the effectiveness of a clock skew-based IDS in detecting masquerade attacks.
- We evaluate our attack on hardware testbeds, including a CAN bus prototype and a real vehicle (the University of Washington EcoCar). Our hardware evaluations show that the cloaking attack is successful against both IDSs during all hardware trials. We show that the NTP-based IDS has a smaller MSI than the state-of-the-art IDS, and is more effective at detecting masquerade attacks.

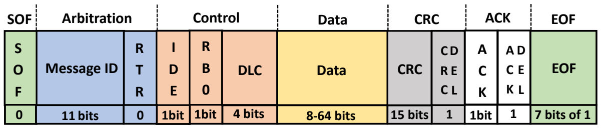
The rest of the paper is organized as follows. Section II explains the adversary model as well as clock-related concepts, and reviews the state-of-the-art IDS. The NTP-based IDS is introduced in Section III, and the cloaking attack is proposed in Section IV. Section V presents the experimental results. Section VI presents our conclusions and future work.

II. Overview of CAN and IDS

Below, we review the CAN protocol and relevant clock related concepts. We then present the adversary model, introduce attack scenarios, and review the state-of-the-art IDS [5].

A. CAN Background

The CAN protocol [10], [11] is one of the most widely used in-vehicle networking standards. CAN is a broadcast bus network, which means that ECUs on the same bus are able to transmit any messages to any ECU and observe all ongoing transmissions. The CAN frame structure is illustrated in Fig. 2. It does not include encryption, authentication, or timestamps.



SOF	Arbitration		Control			Data	CRC		ACK		EOF
SOF	Message ID	RTR	IDE	RB0	DLC	Data	CRC	CRC DEL	ACK	ACK DEL	EOF
0	11 bits	0	1bit	1bit	4 bits	8-64 bits	15 bits	1	1bit	1	7 bits of 1

Fig. 2: Structure of CAN frame. Each frame consists of Start of Frame (SOF) field, Arbitration field, Control field, Data field, CRC field, ACK field, and End of Frame (EOF) field.

The CAN bus acts as a logical AND gate, that is, if two ECUs transmit simultaneously, the message with a smaller ID (higher priority) will be transmitted, through a process known as arbitration. For example, if messages 0x100 and 0x010 are transmitted simultaneously, the ECU that attempts to transmit its message ID 0x100 one bit at a time (starting from the most significant bit) will observe a 0 bit on the CAN bus although it had transmitted a 1, recognize that another ECU is transmitting a higher priority message, and stop its transmission.

B. Clock-Related Concepts

In this section, we follow the NTP definitions of clocks [8], [9], [12]. Let us first define C_{true} as the "true" clock that runs at a constant rate, i.e., $C_{true}(t) = t$. Let $C_A(t)$ denote the time kept by clock A. The *clock offset* of C_A, denoted as $O_A(t)$, is the difference between the time reported by C_A and the "true" time, i.e.,

$$O_A(t) = C_A(t) - C_{true}(t). \tag{1}$$

The *frequency* of C_A at time t is given by $C'_A(t)$. The *clock skew* of C_A, denoted as $S_A(t)$, is the difference in the frequencies (or first derivatives) of C_A and C_{true}, i.e.,

$$S_A(t) = C'_A(t) - C'_{true}(t). \tag{2}$$

A positive clock skew means that C_A runs faster than the true clock, while a negative clock skew implies that C_A runs slower than the true clock. The unit of clock skew is microseconds per second (μs/s) or parts per million (ppm). For example, if C_A is faster by 5μs every 10ms according to C_{true}, then its clock skew relative to C_{true} is 500ppm.

In-vehicle ECUs typically have constant clock skews [5]. Suppose that C_A has a constant clock skew S_A. If Δt is the time duration measured by C_{true}, the amount of time that has passed according to C_A is $\Delta t_A = (1 + S_A)\Delta t$, and $\Delta t = \Delta t_A/(1 + S_A)$. Similarly, if there is a second non-true clock B with a constant clock skew S_B that reports a time duration of Δt_B, we have $\Delta t_B = (1 + S_B)\Delta t$. Then the clock skew of C_B relative to C_A, denoted as S_{BA}, is given by

$$S_{BA} = \frac{\Delta t_B - \Delta t_A}{\Delta t_A} = \frac{S_B - S_A}{1 + S_A} \tag{3}$$

and the relationship between S_{BA} and S_{AB}, that is, the clock skew of C_A relative to C_B, is given by

$$S_{AB} = \frac{-S_{BA}}{1 + S_{BA}}. \tag{4}$$

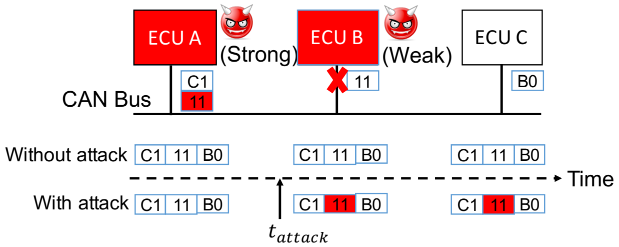


Fig. 3: Illustration of a masquerade attack. In this example, ECU A is fully compromised by the strong attacker, and ECU B is weakly compromised by the weak attacker. Before the attack, ECU B transmits message 0x11 every T sec. At $t = t_{attack}$, the weak attacker suspends ECU B's transmission of message 0x11, and the strong attacker starts fabricating and injecting spoofed messages with ID=0x11 every T sec.

When such a "true" clock does not exist, a non-true clock is chosen as the *reference* clock. Then *relative offset* and *relative skew* are defined for other clocks with respect to the reference clock. Two clocks are said to be *synchronized* at a particular moment if both relative offset and relative skew are zero.

C. Adversary Model and Attack Scenarios

Adversaries can compromise in-vehicle ECUs physically or remotely by exploiting various attack surfaces [1]. As in [5], we consider two types of attackers with different capabilities: 1) *weak attacker*, who is assumed to be able to suspend the transmission of messages of the weakly compromised ECU, but cannot inject any messages, and 2) *strong attacker*, who is assumed to be able to suspend messages of the fully compromised ECU and inject arbitrary attack messages.

The two types of attackers naturally lead to three attack scenarios: *suspension*, *fabrication*, and *masquerade* attacks. In a suspension attack, a weakly compromised ECU is prevented from transmitting certain messages, whereas in a fabrication attack, a fully compromised ECU injects fabricated messages with legitimate IDs. Since most in-vehicle CAN messages are periodic, the above two attacks would significantly change the frequency of certain messages, and thus can be easily detected by state-of-the-art IDSs [3], [6], [7], [13].

Masquerade attacks combine suspension and fabrication attacks. In a masquerade attack, two ECUs A and B are compromised by strong and weak attackers respectively (Fig. 3). The goal of the attack is to impersonate ECU B by injecting periodic messages with spoofed IDs. During the attack, the weak attacker who has compromised ECU B suspends certain messages from ECU B, while the strong attacker uses the fully compromised ECU A to inject messages claiming to originate from ECU B. It has been shown that the masquerade attack can potentially cause severe problems to the vehicle [4], [14]. Although the previously mentioned IDSs actively monitor the CAN bus traffic, the masquerade attack does not change the frequency of the spoofed message, and thus is more difficult to detect than the suspension and fabrication attacks.

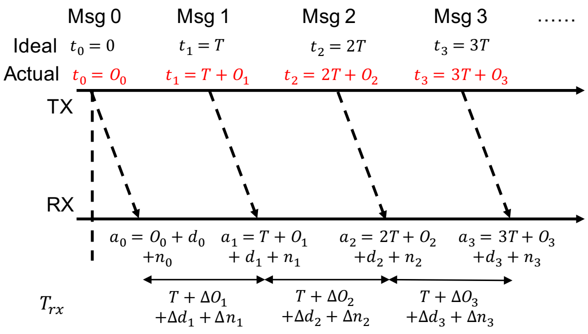

Msg 0
Msg 1
Msg 2
Msg 3
......
Ideal
$t_0 = 0$
$t_1 = T$
$t_2 = 2T$
$t_3 = 3T$
Actual
$t_0 = O_0$
$t_1 = T + O_1$
$t_2 = 2T + O_2$
$t_3 = 3T + O_3$
TX
RX
$a_0 = O_0 + d_0 + n_0$
$a_1 = T + O_1 + d_1 + n_1$
$a_2 = 2T + O_2 + d_2 + n_2$
$a_3 = 3T + O_3 + d_3 + n_3$
T_{rx}
$T + \Delta O_1 + \Delta d_1 + \Delta n_1$
$T + \Delta O_2 + \Delta d_2 + \Delta n_2$
$T + \Delta O_3 + \Delta d_3 + \Delta n_3$

Fig. 4: Timing analysis of message arrivals in CAN.

D. Clock Skew-Based Detection

In-vehicle ECUs operate according to their local clocks with distinct clock skews, which can be exploited for fingerprinting. Methods proposed in [15]–[17], however, are not applicable to CAN since there are no transmit timestamps in CAN messages. To bypass this issue, the state-of-the-art IDS in [5] exploits the fact that almost all CAN traffic is periodic and uses message periodicity to extract and estimate the transmitters' clock skews for detecting masquerade attacks and identifying the compromised ECU. We now review the IDS of [5].

1) Timing model for CAN: Fig. 4 illustrates the timing of a periodic message from the perspective of a receiving ECU R. Since only R's timestamps are available, we consider its clock as the reference, and refer to the relative offset and relative skew of the transmitter's clock as offset and skew, respectively.

Suppose that the transmitter transmits a message every T sec according to its local clock. In the ideal case where the two clocks are synchronized, message i will be transmitted at $t_i = iT$ in R's clock[1]. Due to clock skew, however, the actual transmission time is $t_i = iT + O_i$ in R's clock, where O_i is the *accumulated offset*[2] since message 0. After a network delay of d_i (due to message transmission, propagation and reception), the message arrives at the incoming buffer of R, and has a timestamp $a_i = iT + O_i + d_i + n_i$, where n_i is zero-mean noise introduced by R's timestamp quantization process [17]. Denote the inter-arrival time between messages $(i-1)$ and i as $T_{rx,i}$, which is given by

$$\begin{aligned} T_{rx,i} &= T + (O_i - O_{i-1}) + (d_i - d_{i-1}) + (n_i - n_{i-1}) \\ &= T + \Delta O_i + \Delta d_i + \Delta n_i, \end{aligned}$$

where ΔO_i is the clock offset in period i, and Δd_i and Δn_i are the differences in network delay and quantization noise, respectively, between periods i and $(i-1)$. Since messages with the same ID usually have the same length and experience a very similar network environment, it is reasonable to assume $\mathbb{E}[\Delta d_i] = 0$, as in [5]. Since $\mathbb{E}[n_i] = 0$ and hence $\mathbb{E}[\Delta n_i] = 0$, we have $\mathbb{E}[T_{rx,i}] = T + \mathbb{E}[\Delta O_i]$.

[1] Strictly speaking, t_i is the time when the transmitter puts the first bit of message i into the outgoing buffer.

[2] For consistency, we adopt the version of formula $t_i = iT + O_i$ from [5], instead of $t_i = iT - O_i$, which is the version that is consistent with the NTP specifications. Nevertheless, it does not affect our following analysis.

2) Clock Skew Detector : To estimate the clock skew, the IDS processes incoming messages in batches of size N (e.g., 20), and computes the "average offset" in the k-th batch,

$$O_{avg}[k] = \frac{1}{N-1} \sum_{i=2}^{N} \left[a_i - (a_1 + (i-1)\mu_T[k-1])\right], \quad (5)$$

where $\mu_T[k-1]$ is the average inter-arrival time of the previous batch, and the quantity in the square brackets is the difference between the measured arrival time and the estimated arrival time for the i-th message.

When an average offset value is computed from the current batch, its *absolute* value is added to the accumulated offset,

$$O_{acc}[k] = O_{acc}[k-1] + |O_{avg}[k]|. \quad (6)$$

It is then modeled as $O_{acc}[k] = S[k]t[k] + e[k]$, where $S[k]$ is the regression parameter, $t[k]$ the elapsed time, and $e[k]$ the identification error. To estimate the unknown parameter S, the Recursive Least Squares (RLS) algorithm is adopted, which minimizes the sum of squares of the modeling errors [18].

In a naive masquerade attack, the impersonating ECU has a clock skew different from the targeted ECU's, which would cause significant identification errors. Hence, the identification error is considered as an indicator of an attack. The IDS tracks the normal clock behavior for messages with the target ID by tracking the mean and standard deviation of the identification errors (denoted as e), μ_e and σ_e. To be robust against noise, μ_e and σ_e are updated only if $|(e-\mu_e)/(\sigma_e)| < \gamma$, where γ is a preset update threshold. For detection, the Cumulative Sum (CUSUM) method, which derives the cumulative sums of deviations from the norm behavior [19], is implemented. Letting $\theta_e = (e-\mu_e)/\sigma_e$, the upper and lower control limits L^+ and L^- are updated for each new error sample as:

$$L^+ = \max(0, L^+ + \theta_e - \kappa), L^- = \max(0, L^- - \theta_e - \kappa),$$

where κ is a sensitivity parameter. If either the control limit exceeds a preset detection threshold Γ, an unexpected change in the estimated clock skew is detected, and the IDS declares an attack. The values of γ, κ, and Γ chosen in [5] are 3, 5, and 5, respectively. A more detailed workflow of the state-of-the-art IDS is provided in Appendix A.

3) Correlation Detector: It is pointed out in [5] that if two messages are from the same transmitter, their average offsets are likely to be equivalent and show high correlation (i.e., the correlation coefficient ρ is close to 1), whereas two messages from different ECUs would have low correlation. Hence, the correlation detector keeps track of the correlation of two highly correlated messages, and declares a masquerade attack if ρ is less than a detection threshold (e.g., 0.8). As a result, in cases where the impersonating ECU happens to have a similar clock skew with the targeted ECU, the masquerade attack may bypass the clock skew detector, but would still be detected by the correlation detector. It is important to note that the clock skew detector applies to any periodic message, but the correlation detector is only applicable to a pair of periodic messages with highly correlated average offsets.

We will use analytical and experimental analyses in Sections IV and V to show that not all pairs of messages from the same transmitter show high correlation. Specifically, we find that high correlation is more likely to exist between two messages that are *consecutively transmitted* by the same ECU and also *consecutively received* by the receiver.

III. NTP-BASED IDS

In this section, we present an adapted IDS that computes clock offsets and clock skews according to the NTP specifications, which is referred to as the NTP-based IDS.

The motivation for our NTP-based IDS is two-fold. First, we note that the the metric in Eq. (5) is not consistent with the NTP definition in Eq. (1), since it does not calculate the time difference between the transmitter's clock and the reference clock. In addition, it is assumed that O_i is a random variable and $\mathbb{E}[\Delta O_i] = 0$, which implies that $\mathbb{E}[O_i] = \mathbb{E}[O_j]$ for $i \neq j$, which does not hold in general since offsets accumulate over time (e.g., if $i \gg j$, $\mathbb{E}[O_i] \gg \mathbb{E}[O_j]$). Our second motivation is the widespread use and acceptance of NTP as a timing mechanism for real-time systems, which raises the question of whether the NTP can be used for intrusion detection as well. The main difference between the state-of-the-art IDS and the NTP-based IDS is clock skew estimation, as described below.

A. Clock Skew Estimation in NTP

In the NTP-based IDS, the accumulated offset up to message i in Fig. 4 is modeled as a random variable $O_i = iO + \epsilon_i$, where O is the constant clock offset in each period T due to the constant clock skew, and ϵ_i is the offset deviation due to ECU jitters. We assume that ϵ_i's are independent and identically distributed. Hence, the expected accumulated offset increases linearly in general as more messages are transmitted.

Consider two consecutively received messages with timestamps a_{i-1} and a_i. From the receiver's perspective, the message period is T in the transmitter's clock, which corresponds to $T_{rx,i} = a_i - a_{i-1}$ (i.e., the observed period) in the receiver's clock. According to Eq. (1), the observed clock offset is

$$\hat{O}_i = T - (a_i - a_{i-1}) = -(O + \Delta\epsilon_i + \Delta d_i + \Delta n_i),$$

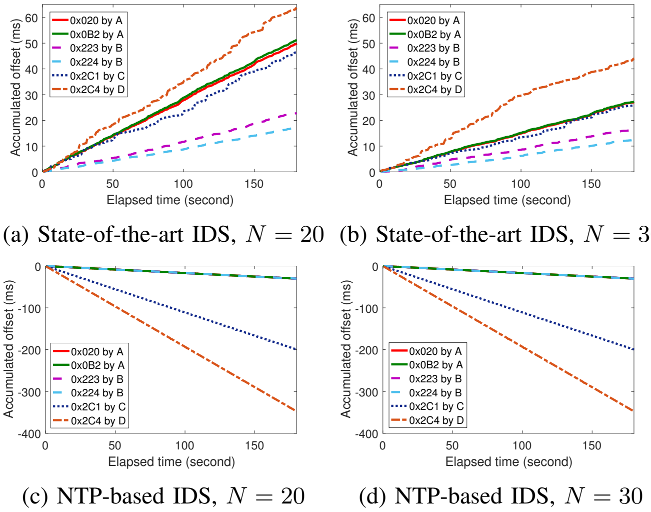
where $\Delta\epsilon_i = \epsilon_i - \epsilon_{i-1}$ and $\mathbb{E}[\Delta\epsilon_i] = 0$. A batch of N messages is used to compute the average offset of the k-th batch $O_{avg}[k]$,

$$O_{avg}[k] = \frac{1}{N} \sum_{i=1}^{N} \hat{O}_i = \frac{1}{N} \sum_{i=1}^{N} [T - (a_i - a_{i-1})] = T - \frac{a_N - a_0}{N}, \quad (7)$$

where a_0 is the timestamp of the last message in the previous batch. The accumulated offset up to the last message of the k-th batch is given by:

$$O_{acc}[k] = O_{acc}[k-1] + N \cdot O_{avg}[k]. \quad (8)$$

Note that the original value of $O_{avg}[k]$ is used, instead of the absolute value as in the state-of-the-art IDS. The other components of the NTP-based IDS remains the same as the state-of-the-art IDS. More details are available in Appendix A.



(a) State-of-the-art IDS, $N = 20$ (b) State-of-the-art IDS, $N = 30$

(c) NTP-based IDS, $N = 20$ (d) NTP-based IDS, $N = 30$

Fig. 5: Accumulated offsets provided by the state-of-the-art IDS and the NTP-based IDS with batch sizes of 20 and 30. The same portion of the data trace (with ID=25) from the Toyota dataset is used. Significant differences in slopes (i.e., estimated clock skew) are observed for the same message using the state-of-the-art IDS, whereas the clock skew estimated by the NTP-based IDS is almost identical with different batch sizes.

B. Estimation Consistency

As a physical property of an ECU, clock skew is considered to be stable over time, and thus the estimated values should be *consistent*, across 1) different batch sizes used by an IDS, 2) different portions of the same trace, and 3) different traces of the same ECU. Hence, we use the Toyota Camry dataset [20] that was used in [5] to compare the NTP-based IDS against the state-of-the-art IDS in terms of estimation consistency.

Fig. 5 illustrates the accumulated offsets estimated by the two IDSs with different batch sizes[3]. Significant differences in slopes for the same message are observed for the state-of-the-art IDS. For example, the estimated clock skew (based on the end point) of message 0x020 is around 273 ppm with $N = 20$, but dropped to around 151 ppm with $N = 30$. In contrast, the NTP-based IDS provides consistent estimation.

To further quantify estimation consistency, we consider the following three cases: 1) use the same portion of the same trace, and vary N from 20 to 100 with a step of 20, 2) set $N = 20$, and use different portions of the same trace by omitting the first m messages, where m is varied from 1 to 19, and 3) set $N = 20$, and use 14 different traces from the Toyota dataset. The standard deviation (σ) of estimated clock skews are adopted as the metric, and a smaller σ value implies more consistent estimation. As shown in Table I, the NTP-based IDS has a significantly smaller σ than the state-of-the-art IDS for all messages in all cases.

TABLE I: Standard deviations (σ_1, σ_2, σ_3) of clock skews estimated by IDSs in three different cases. The NTP-based IDS has a significantly smaller σ than the state-of-the-art IDS, which demonstrates its consistency in clock skew estimation.

Message ID	State-of-the-art IDS			NTP-based IDS		
	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
0x020	92.3682	12.0589	20.1727	0.3706	0.2000	1.7716
0x0B2	94.4480	11.7543	19.4549	0.4252	0.2045	1.7929
0x223	41.6631	16.2060	25.4885	0.3083	0.4429	1.6437
0x224	29.0736	17.0442	32.6059	0.8348	0.5491	2.3660
0x2C1	85.3753	7.9963	26.2618	0.0866	1.2191	3.2977
0x2C4	116.5630	13.0896	53.7820	1.0763	1.1599	3.3602

IV. Proposed Cloaking Attack

In this section, we propose the *cloaking attack*, an intelligent masquerade attack, in which the adversary adjusts the inter-transmission time of spoofed messages to manipulate the estimated clock skew as well as correlation to bypass an IDS.

A. Cloaking Attack on Clock Skew Detector

Consider a message transmitted by the targeted ECU B every T seconds (e.g., 20 ms) in its own clock, which corresponds to every $\hat{T} = T/(1 + S_B)$ seconds in the receiver R's clock, where S_B is B's clock skew. For the ease of discussion, we ignore offset deviations and the noise in arrival timestamps due to network delay and quantization. Then B's clock skew as estimated by R is given by $\hat{S} = (T - \hat{T})/\hat{T} = S_B$.

In a masquerade attack, the weak attacker prevents the targeted message from being transmitted by ECU B. The strong attacker controlling ECU A transmits the false message every T seconds, as measured by C_A (Fig. 3). Hence, ECU R receives the message every $\hat{T}' = T/(1 + S_A)$ seconds, as measured by C_R, where S_A is the clock skew between C_R and C_A. The clock skew measured by ECU R for the messages injected by the attacker will then be $\hat{S}' = S_A$. Therefore, if $S_A \neq S_B$, then the IDS will detect a change in the estimated clock skew after the adversary begins transmitting.

The insight underlying our attack is that while the clock skew is a physical property, clock skew estimation in any IDS is based entirely on message inter-arrival times, which can be easily manipulated by the transmitter (i.e., the strong attacker controlling ECU A) adjusting the message inter-transmission times. Effectively, the adversary *cloaks* the skew of its hardware clock, thus motivating the term *cloaking attack*. Under the cloaking attack, instead of transmitting every T seconds, the attacker-controlled ECU A transmits every $\tilde{T} = T + \Delta T$ seconds, in order to match the clock skew observed at R.

The choice of ΔT is discussed as follows. Under the cloaking attack, the inter-arrival time observed by R is

$$\hat{T}'' = \frac{\tilde{T}}{1 + S_A} = \frac{T + \Delta T}{1 + S_A}$$

and the transmitter's clock skew estimated by R is

$$\hat{S}'' = \frac{T - \hat{T}''}{\hat{T}''} = \frac{S_A \cdot T - \Delta T}{T + \Delta T}. \qquad (9)$$

[3] Due to the lack of ground truth, the authors in [5] empirically identified that 0x020, 0x0B2, 0x223 and 0x224 are transmitted by two different ECUs. However, based on our NTP-based clock skew estimation results, we believe that the four messages come from the same ECU.

Hence, to bypass the IDS, the adversary needs to choose ΔT such that $\hat{S}'' = \hat{S}$, or equivalently $\hat{T}'' = \hat{T}$, which means

$$\Delta T = \frac{(S_A - S_B)}{1 + S_B} \cdot T = S_{AB} \cdot T = \frac{-S_{BA}}{1 + S_{BA}} \cdot T, \qquad (10)$$

where S_{AB} is A's clock skew relative to B's clock, and the last two equalities are due to Eq. (3) and Eq. (4), respectively.

Therefore, the message inter-transmission time $\tilde{T}$ would be

$$\tilde{T} = T + \Delta T = T - \frac{S_{BA}}{1 + S_{BA}} T = \frac{T}{1 + S_{BA}},$$

which is the period of the message from B (i.e., weak attacker) measured by the local clock of A (i.e., strong attacker).

To summarize, the cloaking attack is performed as follows. After the adversary compromises two ECUs as strong and weaker attackers, the strong attacker estimates the period of the target message $\tilde{T}$ as measured by its local clock. During the cloaking attack, the strong attacker transmits spoofed messages every $\tilde{T}$ seconds. While the preceding analysis ignores the noise in the system, our results in Section V show that the cloaking attack is effective in a realistic environment.

B. Maximum Slackness Index (MSI)

In practice, the adversary will be unable to precisely match the clock skew of the targeted ECU due to hardware limitations. Deviations between the clock skew of the attacker and the targeted ECU, however, may still be mistaken for random delays and quantization errors by the IDS. These sources of randomness create an interval of ΔT that an adversary can introduce while remaining undetected; the more effective the detector, the smaller the interval of ΔT will be. We introduce a metric that formalizes this notion as follows. We first let $P_s(\Delta T)$ denote the probability of a successful cloaking attack when the added delay is ΔT. We define the upper and lower limits of ΔT for a successful attack as

$$(\Delta T)_{\max}(\epsilon) = \max\{\Delta T : P_s(\Delta T) > 1 - \epsilon\}$$
$$(\Delta T)_{\min}(\epsilon) = \min\{\Delta T : P_s(\Delta T) > 1 - \epsilon\}.$$

We define the *ϵ-Maximum Slackness Index* (ϵ-MSI) as ϵ-MSI $= (\Delta T)_{max}(\epsilon) - (\Delta T)_{min}(\epsilon)$. The normalized ϵ-MSI is the ratio between of ϵ-MSI (in μs) and the message period (in seconds), and its unit is ppm. Intuitively, a smaller value of ϵ-MSI signifies a more effective detector and less freedom for the attacker, since the adversary's clock skew must closely match the targeted ECU's in order to remain undetected.

C. Cloaking Attack on Correlation Detector

In practice, it is not uncommon for an ECU to transmit multiple messages with the same or different periods and priorities (i.e., sibling messages). If the spoofed message has a sibling message with the same period and highly correlated offsets, the correlation detector can be deployed as the secondary countermeasure. Before introducing the cloaking attack on the correlation detector, let us discuss why two messages consecutively transmitted and consecutively received are more likely to have high correlation in average offsets. Due to space constraints, we focus on the NTP-based IDS, but the same logic is applicable to the state-of-the-art IDS.

Denote the i-th message in the k-th batch for messages v and w as $v_{k,i}$ and $w_{k,i}$, which are transmitted at $t_{k,i}^{(v)}$ and $t_{k,i}^{(w)}$, respectively.[4] Without loss of generality, suppose that $w_{k,i}$ is transmitted right after $v_{k,i}$. Let Δt be the transmission duration of each message v, which is constant, given the fixed message length and CAN bus speed. Hence, we have $t_{k,i}^{(w)} = t_{k,i}^{(v)} + \Delta t$.

Let us consider the first case where $v_{k,i}$ and $w_{k,i}$ are received consecutively at $a_{k,i}^{(v)}$ and $a_{k,i}^{(w)}$, which means no other messages with higher priority IDs are received between $a_{k,i}^{(v)}$ and $a_{k,i}^{(w)}$ due to arbitration. For simplicity, we assume constant network delays for both messages (denoted as d_v and d_w, respectively), and ignore quantization noise at the receiver. Therefore we have $a_{k,i}^{(w)} = a_{k,i}^{(v)} + \Delta t + (d_w - d_v)$.

In the NTP-based IDS, the estimated average offset for messages v and w in the k-th batch are

$$\begin{aligned} O_{avg}^{(v)}[k] &= T - \frac{1}{N}\left(a_{k,N}^{(v)} - a_{k,0}^{(v)}\right) \\ &= -O^{(v)} - \frac{1}{N}\left(\epsilon_{k,N}^{(v)} - \epsilon_{k,0}^{(v)}\right) \\ O_{avg}^{(w)}[k] &= T - \frac{1}{N}\left(a_{k,N}^{(w)} - a_{k,0}^{(w)}\right) = O_{avg}^{(v)}[k]. \qquad (11) \end{aligned}$$

Since $O_{avg}^{(v)}[k]$ and $O_{avg}^{(w)}[k]$ are the k-th realizations of the random variables $O_{avg}^{(v)}$ and $O_{avg}^{(w)}$, respectively, Eq. (11) implies $O_{avg}^{(w)} = O_{avg}^{(v)}$, and thus their correlation coefficient ρ is as high as 1. In general, as along as the two messages are received with a constant delay (consecutive reception is a special case), they will have high correlation. In practice, however, the correlation would slightly decrease due to network delay variations and quantization noise at the receiver.

Next we examine the second case in which messages with higher priority IDs are received in between the two messages. Let the arbitration delay be $d_{k,i} \geq 0$, and thus $a_{k,i}^{(w)} = a_{k,i}^{(v)} + \Delta t + (d_w - d_v) + d_{k,i}$. Then we have

$$O_{avg}^{(w)}[k] = O_{avg}^{(v)}[k] - \frac{1}{N}(d_{k,N} - d_{k,0}), \qquad (12)$$

where the second term may be considered as the k-th realization of a random variable D, independent of $O_{avg}^{(v)}$ and $O_{avg}^{(w)}$. Therefore, we have $O_{avg}^{(w)} = O_{avg}^{(v)} + D$, and

$$\rho\left(O_{avg}^{(v)}, O_{avg}^{(w)}\right) = \frac{\sqrt{Var(O_{avg}^{(v)})}}{\sqrt{Var(O_{avg}^{(v)}) + Var(D)}} < 1.$$

As a result, depending on the variance of arbitration delay, the correlation in the second case may be much smaller than 1.

On the other hand, if two messages are transmitted from different ECUs, we have $O_{avg}^{(w)}[k] = -O^{(w)} - \frac{1}{N}\left(\epsilon_{k,N}^{(w)} - \epsilon_{k,0}^{(w)}\right)$.

[4] This is another requirement for two messages to be highly correlated: the two consecutively transmitted messages needs to be processed as simultaneously as the i-th message in the k-th batch.

Since $\{\epsilon_{k,i}^{(v)}\}$ and $\{\epsilon_{k,i}^{(w)}\}$ are independent, $O_{avg}^{(w)}$ is also independent of $O_{avg}^{(v)}$, which implies $\rho \approx 0$. The above analysis is supported by our hardware evaluation (Section V-D).

Hence, the attacker adopts the following strategy to thwart the correlation detector. Before the attack, the attacker observes the targeted message for a certain duration and identifies any sibling messages. During the attack, the strong attacker-controlled ECU A transmits a spoofed message immediately after a sibling message is received. Since the spoofed and sibling messages are transmitted almost consecutively, their average offsets will be equivalent and highly correlated (Eq. (11)). Note that Eq. (11) also implies that their estimated accumulated offsets and clock skews will be equivalent, thus bypassing the clock skew detector at the same time.

V. Evaluation

In this section, we evaluate the performance of the proposed cloaking attack on two CAN bus testbeds, and demonstrate that the cloaking attack is able to bypass both the state-of-the-art and the NTP-based IDSs. We first describe our testbeds, followed by an illustration of a single trial run of our proposed attack. We then give detailed results for the cloaking attack against both the clock skew and correlation detectors.

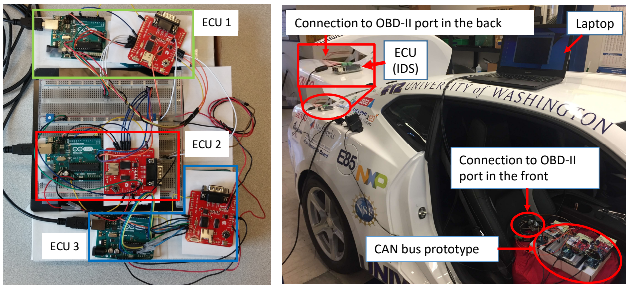
A. Testbeds

We built two CAN bus testbeds: a CAN bus prototype and a CAN testbed on a real vehicle (University of Washington EcoCar, a 2016 Chevrolet Camaro [21]). Compared with the prototype with three ECUs, the EcoCar has 8 stock ECUs and 2 experimental ECUs. There are a total of 89 messages with different IDs, and 2500+ messages are being exchanged every second. The EcoCar testbed provides a real CAN environment to evaluate and demonstrate the proposed cloaking attack.

1) CAN Bus Prototype: As shown in Fig. 6(a), our CAN bus prototype consists of three ECUs. Each ECU is composed of an Arduino UNO board and a Sparkfun CAN bus shield. The CAN bus shield uses a Microchip MCP2515 CAN controller, a Microchip MCP2551 CAN transceiver, and 120Ω terminator resistors. The bus speed of the prototype is set to 500 Kbps as in typical CAN buses. ECU 1 is the receiving ECU (i.e., the IDS) that log all messages. ECU 2 is the targeted ECU controlled by the weak attacker that transmits messages 0x11 every 100 ms (i.e., 10 Hz). ECU 3 is the strong attacker that impersonates ECU 2 in a masquerade or cloaking attack.

2) EcoCar CAN testbed: As shown in Fig. 6(b), the CAN bus prototype is connected to the in-vehicle CAN bus of the EcoCar via the On-Board Diagnostics (OBD-II) port to build the EcoCar testbed. During our experiments, the EcoCar is in the park mode in an isolated and controlled environment, but all ECUs are functional and actively exchange CAN messages, and ECUs in the park mode have almost the same constant clock skews as in the drive mode.

Due to large CAN traffic and limited computing capability, the Arduino-based ECU is not able to log all messages on the CAN bus. Hence, we build a fourth ECU that consists of a Raspberry Pi 3 and a PiCAN 2 board (which has the same CAN controller and transceiver as in the CAN bus shield) as the receiving ECU. A stock ECU is considered as the targeted ECU (the weak attacker) which transmits message 0x184 every 100 ms (i.e., 10 Hz), and the Arduino-based ECU 3 acts as the strong attacker that injects spoofed messages.


ECU 1
ECU 2
ECU 3
Connection to OBD-II port in the back
ECU (IDS)
Laptop
Connection to OBD-II port in the front
CAN bus prototype

(a) CAN bus prototype (b) EcoCar testbed

Fig. 6: CAN bus testbeds. The CAN bus prototype is connected to the CAN bus inside the EcoCar via the OBD-II port to build the EcoCar testbed.

B. Example of NTP-based IDS

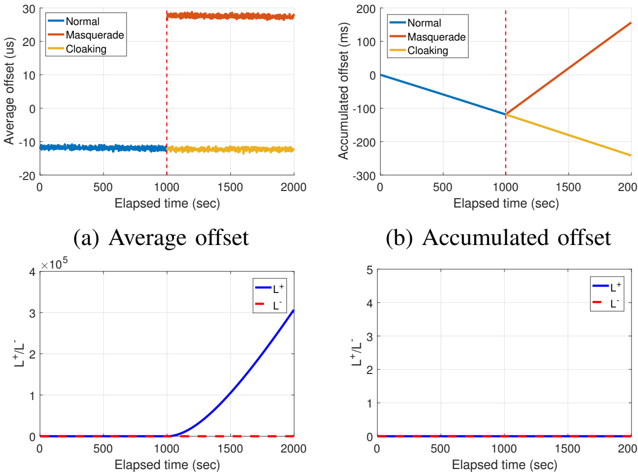
For illustration, we first describe a single execution of the masquerade attack and the behavior of the NTP-based IDS. We compare the masquerade attack without cloaking and our proposed cloaking attack. In the example, we set the update threshold γ to 4 and the detection threshold Γ to 5 for the NTP-based IDS. For the data collected from the CAN bus prototype, the sensitivity parameter κ is set to 5.

The IDS first tracks the clock skew of message 0x11 from the targeted ECU for 1000 seconds, before the attack happens. Then the IDS is fed with the timestamps of attack messages. For the masquerade attack, the strong attacker transmits every $T = 100$ ms as per its local clock. For the cloaking attack, the strong attacker observes the inter-arrival time of message 0x11 to be around 100040 μs, and then sets the message inter-transmission time to $\tilde{T} = 100040$ μs, where $\Delta T = 40$ μs.

As shown in Fig. 7, when the masquerade attack happens, the average offset immediately jumps from around -12 μs to around 28 μs (Fig. 7(a)), and the slope changes from -118.9 ppm to 275.3 ppm (Fig. 7(b)), because of the very distinct clock skews between targeted and impersonating ECUs. As a result, such deviations add up and cause the control limits of the IDS to increase (Fig. 7(c)). In contrast, under the cloaking attack, the average offset stays almost the same as the original curve, as does the slope of the accumulated offset. Since the deviations are so small, the control limits are always zero, and thus the IDS is unable to detect the cloaking attack (Fig. 7(d)). Tests on the EcoCar testbed lead to similar observations.

C. Performance of Cloaking Attack on Clock Skew Detector

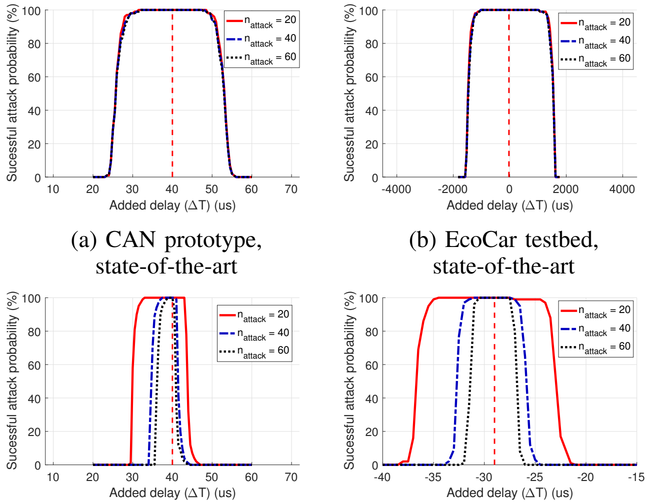
When launching the cloaking attack, the impersonating ECU (Arduino-based) transmits every 100040 μs ($\Delta T = 40$ μs) on the CAN bus prototype to spoof the 10 Hz message 0x11, and



(a) Average offset (b) Accumulated offset

(c) Control limits under the masquerade attack (d) Control limits under the cloaking attack

Fig. 7: Behavior of the NTP-based IDS under the masquerade and cloaking attacks on the CAN bus prototype, in terms of average offset, accumulated offset and control limits. In the masquerade attack, the accumulated offset grows over time and is detected by both IDS. Under the cloaking attack, the clock skews before and after the attack are indistinguishable.



(a) CAN prototype, state-of-the-art (b) EcoCar testbed, state-of-the-art

(c) CAN prototype, NTP-based (d) EcoCar testbed, NTP-based

Fig. 8: Successful attack probability on the state-of-the-art IDS and the NTP-based IDS on the CAN bus prototype and EcoCar testbed with message period 100 ms. For the value of $\Delta T = 40\ \mu s$ achieved in our hardware experiments (red dashed line), the attack was successful in all test cases. The width of each curve is equal to the ϵ-MSI for the given detector.

every 99971 μs ($\Delta T = -29\ \mu$s)[5] to spoof message 0x184 on the EcoCar testbed. We collected 8.5 hours of attack data from the CAN bus prototype and the EcoCar testbed separately.

To simulate the cloaking attack, the IDS is fed with 1000 batches of normal data, followed by n_{attack} batches of attack data in each experiment. We assume perfect timing for the cloaking attack, i.e., the first attack message is received at the next expected time instant of the targeted message. The impact of mistiming on the cloaking attack is studied in Appendix B. An attack is successful if it is undetected by the IDS, and failed otherwise. A total of 100 non-overlapping segments of size n_{attack} are prepared from the attack data to simulate 100 independent attacks. To measure the attack performance, we compute *successful attack probability*, denoted as P_s, which is the percentage of experiments where the attack is successful.

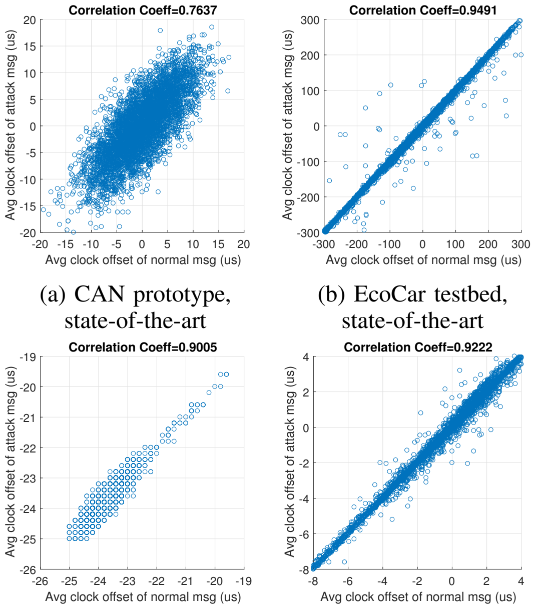
We consider the state-of-the-art IDS and the NTP-based IDS with batch size equal to 20. For the state-of-the-art IDS, the update threshold γ is set to 3 and the detection threshold Γ is 5 [5]. For the NTP-based IDS, we use $\gamma = 4$ and $\Gamma = 5$. For the data collected from the CAN bus prototype, the sensitivity parameter κ is set to 5 for both IDSs. It is set to 8 for the data collected from the EcoCar testbed to avoid false alarms.

For the value of ΔT achieved in our evaluation, the probability of successful attack was 1 against both the NTP-based IDS and the state-of-the-art IDS (Fig. 8, dashed line). In order to gain additional insight into the performance of each IDS under cloaking attack, we generated additional data sets by adding different values of ΔT to the message inter-arrival times, and then analyzed the new datasets using both IDSs.

On the CAN bus prototype, with $n_{attack} = 20$ and $\epsilon = 0.05$, the ϵ-MSI value for the state-of-the-art IDS is 22.5 μs (Fig. 8(a)), but only 11.5 μs for the NTP-based IDS (Fig. 8(c)). Hence, it is much easier for the cloaking attack to bypass the state-of-the-art IDS than the NTP-based IDS. We also found that increasing n_{attack} has little impact on ϵ-MSI for the state-of-the-art IDS, which is 20.5 μs for $n_{attack} = 40$ or 60, but significantly impacts ϵ-MSI of the NTP-based IDS, which varies from 11.5 μs to 2.5 μs as n_{attack} is increased from 20 to 60. This result suggests that the performance of the NTP-based IDS improves over the attack duration. Another interesting observation is that the P_s curves are skewed instead of symmetric. This is because when the Arduino-based ECU starts operating, its clock skew slowly decreases due to the temperature change in hardware. As a result, the IDS tends to overestimate the clock skew, and is more sensitive to a larger positive delay (that would further decrease the clock skew).

ϵ-MSI for the state-of-the-art IDS increases significantly for a real vehicle, as shown in Fig. 8(b), due to the significantly heavier CAN traffic compared to the prototype, which reduces the effectiveness of the detection. As an example, a cloaking attack with ΔT between $-1029\ \mu$s and 1021 μs can bypass the state-of-the-art IDS with 100% probability regardless of n_{attack}. For the NTP-based IDS with $\epsilon = 0.01$, ϵ-MSI is 10.5 μs for $n_{attack} = 20$, and 3 μs for $n_{attack} = 60$. Hence, in the real vehicle, as in the CAN prototype, the NTP-based IDS is more effective in detecting masquerade attacks than the state-of-the-art IDS. The proposed cloaking attack, however, is still able to thwart both detection schemes when ΔT is chosen to be within the interval $[(\Delta T)_{min}(\epsilon), (\Delta T)_{max}(\epsilon)]$.

[5] While Arduino's time resolution is 4 μs , we set ΔT to $-28\ \mu$s and changed it to $-32\ \mu$s every five messages so that $\Delta T \approx -29\ \mu$s on average.



(a) CAN prototype, state-of-the-art

(b) EcoCar testbed, state-of-the-art

(c) CAN prototype, NTP-based

(d) EcoCar testbed, NTP-based

Fig. 9: Scatter plot of the average offsets of sibling messages and attack messages under cloaking attack. The correlation coefficient is above 0.9 in the EcoCar testbed, and hence is comparable to the coefficient for consecutive messages.

D. Performance of Cloaking Attack on Correlation Detector

In this section, we demonstrate and evaluate the cloaking attack on the correlation detector. On the CAN bus prototype, the targeted message is 5 Hz. When launching the cloaking attack, the Arduino-based impersonating ECU transmits a spoofed message 0x11 after it observes a sibling message of the targeted message, with a constant delay of 100 ms[6]. On the EcoCar testbed, two 100 Hz messages 0xC1 and 0xC5 from a stock ECU are identified to be highly correlated. We choose 0xC5 as the target, and 0xC1 as its sibling message. Due to limited computing capabilities, the Arduino-based ECU is not able to receive all messages on the CAN bus, filter for the sibling message, and transmit the spoofed message. Hence, we use the Raspberry-Pi-based ECU as the impersonating ECU. It injects messages with a non-conflicting ID 0xC0, instead of 0xC5, in order to avoid any undesirable impact on the EcoCar. A total of 14 hours and 1.2 hours of attack data were collected from the CAN bus prototype and the EcoCar testbed, respectively. As a baseline, we collected 4.7 hours of normal data with one ECU transmitting two messages consecutively on the CAN bus prototype. For the EcoCar testbed, since the targeted message is not suspended (for safety), the data we collected also contains the normal data. The same settings in Section V-C are used for state-of-the-art and NTP-based IDSs.

Fig. 9 shows a typical scatter plot of average offsets of the sibling message and the attack message, when the cloaking attack is mounted. For the CAN bus prototype, the correlation is 0.76 and 0.90 for state-of-the-art and NTP-based IDSs, respectively. This is mainly because an Arduino-based ECU is dedicated to transmission, which implies a smaller jitter and offset deviation, while the quantization error is quite significant due to the Arduino's 4 μs time resolution. On the EcoCar testbed, the cloaking attack can achieve correlation up to 0.95 and 0.92 for state-of-the-art and NTP-based IDSs.

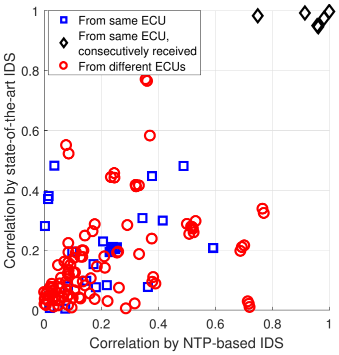


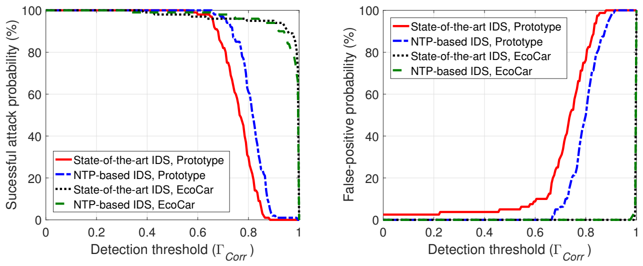
Fig. 10: Correlation relationship between pairwise messages on the EcoCar testbed. Consecutive messages from the same ECU are highly correlated, while others are less correlated.

To understand the correlation relationship between pairwise messages on the EcoCar testbed, we examine 17 messages from 5 ECUs with periods of 10 ms, 12 ms or 100 ms, based on the ground truth provided by the manufacturer. All pairs of messages are classified into the following three categories: 1) from the same ECU and (almost always) received consecutively, 2) from the same ECU but not received consecutively, or 3) from different ECUs. Correlation values are computed using 200 batches. As illustrated in Fig. 10, for two messages from different ECUs, their correlation is generally low (e.g., less than 0.2) for both state-of-the-art and NTP-based IDSs. In addition, not all pairs of messages from the same ECU have high correlation: 81% of them have correlation less than 0.6, and there are only 5 pairs with correlation higher than 0.9 for both IDSs. We checked such pairs and confirmed that their messages are always consecutively received. This result is indeed consistent with our analysis in Section IV-C.

Next we evaluate the performance of the cloaking attack. An attack on the correlation detector is successful if the resulting correlation is higher than or equal to the detection threshold Γ_{corr}, and failed otherwise. A total of 100 experiments using the attack data are conducted, each consisting of 50 batches, to compute the successful attack probability P_s. Intuitively, a higher Γ_{corr} may cause a IDS to report a false alarm, i.e., declaring an attack when there is actually none. The false alarm probability P_{fa} is equal to the percentage of experiments where the IDS reports a false alarm. A total of 80 experiments using the normal data are conducted to compute P_{fa}.

Fig. 11 illustrates P_s and P_{fa} as a function of the detection threshold Γ_{corr}. As we can see, a larger Γ_{corr} decreases P_s, making the attack more difficult, but also leads to more false alarms. On the CAN bus prototype, the state-of-the-art IDS

[6] As mentioned in Section IV-C, as long as two messages are received with a constant delay, they will be highly correlated. To validate this, we programmed the strong attacker to transmit after a constant delay on the CAN bus prototype.



(a) Successful attack probability (b) False alarm probability

Fig. 11: Successful attack probability and false-positive probability of the cloaking attack on the correlation detector under changing detection threshold. In the CAN prototype, if the detector is chosen to achieve the probability of false alarm $P_{fa} \leq 0.05$, then the attack succeeds with probability at least 0.95. In the EcoCar, the probability of success for the attack is 0.8 when the detector parameters are chosen so that $P_{fa} = 0$.

needs to set Γ_{corr} to 0.54 to ensure $P_{fa} \leq 5\%$, at which point we have $P_s = 100\%$. For the NTP-based IDS, when Γ_{corr} is 0.68, we have $P_{fa} \leq 5\%$ and $P_s = 98\%$. It demonstrates that the cloaking attack is able to effectively bypass both IDSs.

On the EcoCar testbed, when Γ_{corr} is 0.8, P_s is 95% and 96% for state-of-the-art and NTP-based IDS, respectively. Since the two messages have very high correlation under the normal condition, Γ_{corr} may be set to 0.975 without any false alarm. At this point, P_s for state-of-the-art and NTP-based IDSs is 89% and 80%, respectively. It is important to note that such attack performance is already achieved with a lower-end ECU based on Raspberry Pi, and we would expect P_s to increase when the cloaking attack is mounted by the strong attacker inside a vehicle, which is left as our future work.

VI. Conclusion

This paper investigated masquerade attacks on in-vehicle networks, in which an adversary compromises ECUs to inject spoofed messages claiming to be from a targeted ECU. Recent works have proposed using the ECU's clock skew as a fingerprint to detect attacks and identify the compromised ECU, resulting in clock skew-based IDSs that make use of first- and second-order moments. In this paper, we proposed the cloaking attack on such IDSs, in which an adversary manipulates the inter-transmission times of spoofed messages in order to match the clock skew of the targeted ECU. We evaluated the cloaking attack on a CAN bus prototype and a real vehicle, and showed that the state-of-the-art IDS was deceived in all test cases. We also proposed and evaluated an adapted IDS based on the NTP. In order to quantify the effectiveness of an IDS, we presented a new security metric, the Maximum Slackness Index, which is the range of added delay that the adversary can introduce without being detected. This work makes the case that the impact of coupling between cyber and physical components in CPS security needs to be understood, especially when attempting to leverage physical invariants arising from physical components to provide security assurances.

Acknowledgment

We would like to thank Drs. Sukarno Mertoguno and David Corman for discussions on this problem, and Dr. Kang G. Shin, for his pioneering work that introduced the clock skew-based IDS for automotive CAN networks. This work was supported by NSF grants CNS-1446866 and CNS-1656981, 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] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Conference on Security*. Berkeley, CA, USA: USENIX Association, 2011, pp. 77–92.

[2] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. of the 2010 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 447–462.

[3] C. Miller and C. Valasek, "Adventures in automotive networks and control units," in *DEF CON21*, 2013.

[4] ——, "Remote exploitation of an unaltered passenger vehicle," in *Black Hat USA*, 2015.

[5] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium*. Austin, TX: USENIX Association, 2016, pp. 911–927.

[6] T. Hoppe *et al.*, "Security threats to automotive CAN networks - practical examples and selected short-term countermeasures," in *Proceedings of the 27th International Conference on Computer Safety, Reliability, and Security*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 235–248.

[7] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, June 2011, pp. 1110–1115.

[8] S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, Mar 1999, pp. 227–234.

[9] D. L. Mills, "Network time protocol (version 3): Specification, Implementation and Analysis," RFC 1305, Tech. Rep., 1992.

[10] ISO, *International Standard ISO 11898-1 Road Vehicles-Controller Area Network (CAN), Part 1 Data Link Layer and Physical Signaling*, 2015.

[11] Bosch, "CAN Specification Version 2.0," 1991.

[12] V. Paxson, "On calibrating measurements of packet transit times," in *Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: ACM, 1998, pp. 11–21.

[13] M. Müter *et al.*, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security*, Aug 2010, pp. 92–98.

[14] "Hackers remotely kill a jeep on the highway - with me in it." https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/, July 2015, accessed: 2018-02-03.

[15] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, ser. MobiCom '08. New York, NY, USA: ACM, 2008, pp. 104–115.

[16] T. Kohno *et al.*, "Remote physical device fingerprinting," in *Proc. of the 2005 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 211–225.

[17] S. Zander and S. J. Murdoch, "An improved clock-skew measurement technique for revealing hidden services," in *Proceedings of the 17th Conference on Security Symposium*, ser. SS'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 211–225.

[18] S. Haykin, *Adaptive filter theory 2nd edition*. Prentice Hall, 2011.

[19] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs, 1993, vol. 104.

[20] R. Ruth, W. Bartlett, and J. Daily, "Accuracy of event data in the 2010 and 2011 Toyota Camry during steady state and braking conditions," in *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 5, 2012, pp. 358–372.

[21] "UW EcoCar," http://uwecocar.com/, accessed: 2017-09-26.

Appendix

A. *Workflow of Clock Skew-Based IDS*

Fig. 12 describes the workflow of a clock skew-based IDS. The following steps are applicable to both the state-of-the-art IDS in [5] and the NTP-based IDS that we propose. An IDS consists of two blocks: clock skew estimation and Cumulative Sum (CUSUM). The clock skew estimation block takes the timestamps of N newly arrived messages as input. For the k-th batch, the state-of-the-art IDS computes the average offset $O_{avg}[k]$ and the accumulated offset $O_{acc}[k]$ according to Eq. (5) and Eq. (6), respectively, whereas the NTP-based IDS follows Eq. (7) and Eq. (8). Then the identification error $e[k]$ is computed, and used to obtain the updated clock skew $S[k]$ using the Recursive Least Square algorithm [5], [18].

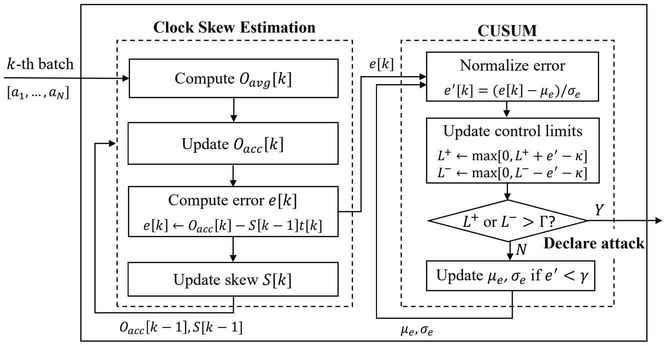


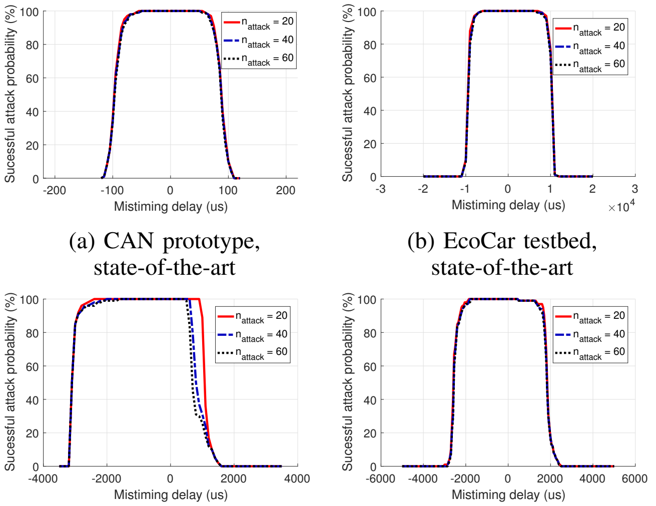
Fig. 12: Workflow of the state-of-the-art IDS [5] and the NTP-based IDS. There are two main blocks: clock skew estimation and CUSUM. The clock skew estimation block updates the clock skew using the most recent batch. CUSUM takes the identification error and update control limits for detection.

The CUSUM block takes the identification error as input. It starts to maintain the statistics of all past identification errors, i.e., mean and standard deviation, after n_{init} (e.g., 50) error samples are received. Then the new error sample is first normalized, and used to update the control limits. If either the upper or lower control limit (L^+/L^-) exceeds the detection threshold Γ, the IDS declares an attack. In order to be robust against noise, if the normalized error $e'[k]$ is less than the threshold γ, the error statistics will be updated using the new error sample $e[k]$ and all past error samples; otherwise, $e[k]$ will be dropped and error statistics will not be updated.

B. *Impact of Mistiming on Cloaking Attack*

In a masquerade or cloaking attack, the strong attacker needs to start transmitting the spoofed message at the time constant at which the targeted message should have been transmitted, if it had not been suspended. It naturally raises the question whether mistiming affects the cloaking attack performance. In this simulation, we introduce a mistiming delay (either positive or negative) between the last message of normal data and the first message of attack data, in addition to the message period.

The IDS is fed with 1000 batches of normal data, followed by n_{attack} batches of attack data with a batch size of 20 in each experiment. Γ is 5 for both IDSs, and γ is set to 3 and 4 for state-of-the-art and NTP-based IDSs, respectively. Also, κ is set to 5 for the CAN bus prototype and to 8 for the EcoCar testbed, respectively.



(a) CAN prototype, state-of-the-art

(b) EcoCar testbed, state-of-the-art

(c) CAN prototype, NTP-based

(d) EcoCar testbed, NTP-based

Fig. 13: Impact of the mistimed cloaking attack on the state-of-the-art IDS and the NTP-based IDS. If the strong attacker can inject the fist attack message on the proper time, the cloaking attack can bypass both IDSs.

Fig. 13 shows the impact of the mistiming of the cloaking attack on state-of-the-art and NTP-based IDSs. In general, larger mistiming causes the attack performance to decrease. On the CAN bus prototype, any amount of mistiming between -55 μs and 55 μs does not affect the attack performance (i.e., P_s is 100% with $n_{attack} = 60$) against the state-of-the-art IDS, whereas the allowed mistiming is much larger for the NTP-based IDS, mainly due to the difference in clock skew estimation. Since the clock skew of the Arduino-based ECU slowly decreases due to the temperature change in hardware as it warms up, the estimator tends to overestimate the clock skew, and thus is more sensitive to larger positive mistiming (that would further decrease the clock skew), which explains the skewness of the curves in Fig. 13(c).

On the EcoCar tested, the allowed mistiming is increased significantly, which is between -6 ms to 7 ms for the state-of-the-art IDS, and between -1.8 ms and 0.5 ms for the NTP-based IDS, due to much heavier CAN traffic in a real vehicle. The above observations imply that the timing is hardly a strict requirement for the adversary to launch a clocking attack in a real vehicle.