CANCloak: Deceiving Two ECUs with One Frame

Li Yue
Tsinghua University
yuel18@mails.tsinghua.edu.cn

Zheming Li
Tsinghua University
lizm20@mails.tsinghua.edu.cn

Tingting Yin
Tsinghua University
ytt18@mails.tsinghua.edu.cn

Chao Zhang✉
Tsinghua University
chaoz@tsinghua.edu.cn

***Abstract*— Modern vehicles have many electronic control units (ECUs) connected to the Controller Area Network (CAN) bus, which have few security features in design and are vulnerable to cyber attacks. Researchers have proposed solutions like intrusion detection systems (IDS) to mitigate such threats. We presented a novel attack, CANCloak, which can deceive two ECUs with one CAN data frame, and therefore can bypass IDS detection or cause vehicle malfunction. In this attack, assuming a malicious transmitter is controlled by the adversary, one crafted CAN data frame can be transmitted to a target receiver, while other ECUs shall not receive that frame nor raise any error. We have setup a physical test environment and evaluated the effectiveness of this attack. Evaluation results showed that success rate of CANCloak reaches up to 99.7%, while the performance depends on the attack payload and sample point settings of victim receivers, independent from bus bit rate.**

I. Introduction

The number of electronic devices in vehicles keeps increasing over the past decade. A modern vehicle could have tens of ECUs as well as hundreds of sensors connected to the CAN bus, which are used by engine units, braking system, transmission system, and infotainment module etc. to improve both safety and comfort of driving. However, due to the lack of security features in design, vehicles are prone to cyber attacks. Several research groups have demonstrated successful attacks against modern vehicles [1]. For instance, attackers could exploit vulnerabilities in WiFi or Bluetooth modules to access vehicles remotely, and then exploit vulnerabilities in infotainment modules to compromise reachable ECUs that are connected to the CAN bus, and finally issue malicious commands through the compromised ECU to control other ECUs and therefore vehicles.

CAN bus is the de-facto standard used by all vehicle manufacturers to provide reliable communication between different in-vehicle ECUs. However, CAN protocol in design lacks modern security features, e.g., end-to-end encryption, message authentication or node authorization. It therefore opens the door to attackers, i.e., a malicious or compromised ECU that connects to the CAN bus could compromise the whole network and take control of a vehicle.

Researchers have proposed several mitigations to address such threats. The first type of solutions is message authentication, which relies on either symmetric keys shared among legitimate ECU nodes [2], or key establishment schemes derived from pre-shared keys [3], or cryptography support with special hardware [4]. These solutions are hard to deploy in practice, since they have to extend the standard CAN protocol and require collaboration from all ECU devices. Another type of solutions provided by researchers are intrusion detection systems (IDS), e.g., [5, 6], which in general attach a detector ECU to the CAN bus and listen all CAN messages to perform anomaly detection, signature or pattern matching. This type of solutions is attractive and easy to deploy in practice.

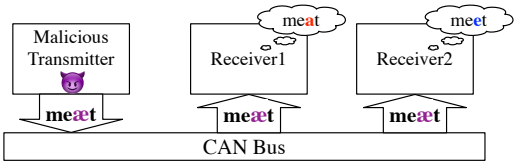

Fig. 1: CANCloak Attack: A malicious transmitter sends an ambitious waveform, different receivers read it differently.

In this paper, we point out a new attack, CANCloak, which can deceive IDS to make wrong decisions, or confuse ECUs to work collaboratively. More specifically, attackers can transmit specially-crafted ambiguous waveform on CAN bus (via a compromised ECU, or a maliciously ECU attached to the On-Board Diagnostics (OBD) port), and lead two receivers to interpret it as two different data frames, as illustrated in Figure 1. In this way, if one of the receivers is an IDS, it may fail to detect command received by the other victim ECU. Besides, if none of the deceived ECUs is an IDS, then attackers may issue two instructions at the same time. The vehicle may have abnormal behavior if these instructions are mutually exclusive(e.g., full-throttle while full-braking).

We studied the data link layer CAN standard [7], and pointed out the root cause is the ambiguous settings of ECU nodes connected to the CAN bus. Specifically, each ECU has a sample-point setting in physical layer, which controls when this ECU reads state of CAN bus and then determines the logic bit value. ECUs with different sample-point settings can communicate on a same CAN bus, which is a common case in practice. Attackers could exploit this inconsistent setting to launch the CANCloak attack.

We make the following contributions in this paper:

- We found a security issue in the physical layer of CAN, which could cause inconsistency to ECU nodes.
- We proposed a CANCloak attack to exploit this issue, which is able to deceive two ECUs with one CAN frame.
- We developed a test environment consisted of 3 ECUs, including a transmitter capable of launch CANCloak attack.

Workshop on Automotive and Autonomous Vehicle Security (AutoSec) 2021
25 February 2021, Virtual
ISBN 1-891562-68-1
https://dx.doi.org/10.14722/autosec.2021.23024
www.ndss-symposium.org

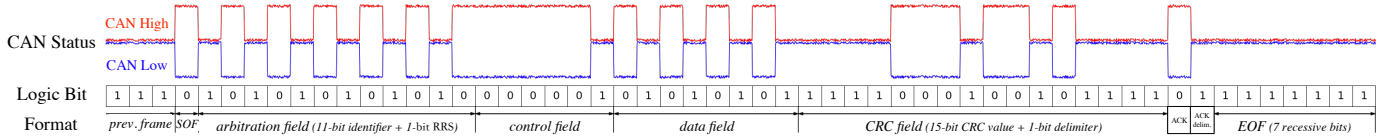

Fig. 2: A Sample CAN Data Frame Signal

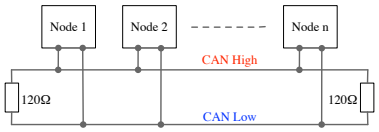

Fig. 3: CAN Bus Topology

– We evaluated the performance of CANCloak, showing an attack success rate up to 99.7% under certain experiment parameter.

II. Background

Over the past few decades, automobile manufacturers equip their new cars with more and more high-tech features, like lane departure warning (LDW) system, blind-spot warning (BSW) systems, and automatic emergency braking (AEB) systems to improve both safety and convenience of driving. Autonomous driving is coming to reality with the help of those devices. The structure of modern cars has transformed from machinery to complex in-vehicle electronics networks. For example, modern vehicles are running with up to 2500 signals exchanged by 70 electronic control units (ECUs) [8].

A. Controller Area Network (CAN)

The controller area network (CAN) is widely used to provide a communication channel for different ECUs within a car. Bosch released the CAN protocol specification [9] in 1986 and then updated it to CAN 2.0 [10] in 1991. After that, the CAN specification became an ISO standard [7] and was widely used by all automobile manufacturers.

Figure 3 shows a typical CAN bus topology. Any node could act as a master (*multi-master*) to send a message over the bus, while other nodes will receive and act on the message (*multi-cast*). Modern vehicles may have multiple CAN buses.

CAN uses a two-wire twisted cable as bus to connect all embedded control units (ECUs). These 2 cables, i.e., CAN High (CANH) and CAN Low (CANL), hold differential signal to achieve high reliability and noise tolerance. As shown in Figure 2, the voltage difference between these two signals turns the bus to one of two states, i.e., the *dominant* state (logic "0") when $V_{CANH} > V_{CANL}$, and the *recessive* state (logic "1") when $V_{CANH} \leq V_{CANL}$. For convenience, we use **negative edge** to represent logic "1" to "0" transition, and use **positive edge** to represent logic "0" to "1" transition.

B. CAN Frame

CAN frame, or "message", is the basic data unit sent through CAN bus. Though there are other type of frames, in this paper, we will only focus on data frames.

As Figure 2 shows, a CAN data frame consists of several fields, including the start of frame (SoF), arbitration field, control field, data field, cyclical redundancy check (CRC) field, acknowledge (ACK) field, and the end of frame (EoF).

SoF and EoF mark the beginning and the end of a data frame, consisting of 1 dominant bit and 7 recessive bits respectively. The arbitration field consists of an 11-bit unique identifier (which represents the message priority) and one Remote Transmission Request (RTR) bit. The control field has several control flags, including IDentifier Extension bit (IDE) and Data Length Code (DLC). The data field contains data to be transmitted with a maximum length of 8 bytes. CRC is applied to detect any transmission error in the whole data frame, followed by acknowledge bits.

C. Arbitration

CAN uses Carrier Sense Multiple Access/Collision Detection (CSMA/CD) as an arbitration scheme. Following this scheme, every ECU must check whether there are no activities for a while on the bus before sending messages, and then monitor collisions during transmission. If an ECU sends a recessive bit and then reads back a dominant bit, it learns that another ECU is sending a message with a higher priority, and thus will stop transmitting immediately.

D. ERROR Handling

To keep robust and functional even under high electrical disturbances, the CAN protocol adopts several error detections.

a) Bit Monitoring: ECU keeps tracking the actual bus status while sending a message. If any inconsistency happens (e.g., sending a recessive bit but reading back a dominant bit), a transmit error shall be raised.

b) CRC Check: Every frame includes a 15-bit CRC. All receiver nodes will re-calculate the CRC and compare it with the one in the received message. If two CRC values mismatch, a receive error shall be raised.

c) Stuff Bit Check: The CAN bus does not have an independent clock wire, and ECU nodes rely on monitoring *negative edges* to synchronize. However, the CAN protocol uses the Non-Return-to-Zero (NRZ) bit encoding scheme, in which consecutive bits of the same polarity exhibit no voltage level changes on the bus. Therefore, if many consecutive bits of the same polarity are being transmitted, then there will be no negative or positive edges on the CAN bus for a period of several bits, therefore receiver nodes may get out-of-synchronization.

To address this issue, a CAN transmitter must actively insert one bit of opposite polarity (called stuff bit) after five consecutive bits of the same polarity have been transmitted in the stream. On the receiver side, if a node detects any

violation against this stuffing rule, i.e., six consecutive bits of same polarity (except EoF bits) are found in a stream, then a stuff error will be raised.

d) Acknowledge Check: Each receiver node should send a dominant bit to the bus during the ACK slot. The transmitter will signal an ACK error if no dominant bit was detected during the ACK slot.

E. CAN Sub-Bit Structure

TABLE I: Segments of CAN bit

Segment	Length	Explanation
Synchronization Segment	1 tq	To synchronize the various CAN nodes on the bus
Propagation Segment	1~8 tqs	Compensate for physical delay times within the network
phase-1 Segment	1~8 tqs	Compensate for edge phase errors, may be lengthened during re-synchronization
phase-2 Segment	2~8 tqs	Similar to phase-1, may be shortened during re-synchronization

tq: time quanta

The minimum time unit used in a CAN controller is **time quanta**, derived from the ECU's system clock. For each ECU, the time period of receiving a bit is split into four segments, i.e., synchronization segment, propagation segment, phase-1 segment, and phase-2 segment, whose time duration are all multiples of the ECU's time quanta. Table I lists the structure of a bit transmission time period, the length limit of each segment and the explanation of each segment.

The separation point between phase-1 and 2 segment is called the **sample-point**. ECU will read voltage of CAN bus at sample-point to get bus state. Note that, segments' length (in number of time quanta) are up to the ECU's settings, so does the position of the sample-point. It is not necessary, and could be physically infeasible, to synchronize all ECUs' sample-points. To make multiple ECUs work on the same CAN bus simultaneously, the only hard requirement is the bus rate, i.e., the length of a bit time of every ECU, must be the same. It is common to find ECUs with different sample-point settings working on a same CAN bus[1].

F. Synchronization Mechanisms

CAN is a message-based protocol and does not have a specific timing wire for synchronization. Thus, CAN receivers rely on extracting timing information from signals on the bus. To receive every bits in a frame correctly, the CAN standard provides two synchronization mechanisms in different layers.

Hard synchronization is the main mechanism used in frame level synchronization. It happens when receiver detects the first negative edge occurs after the last frame and inter-frame interval on bus. This negative edge indicates the bus idle status is over, and the first bit (SoF) of a new frame is incoming. Once detected, the bit timing logic unit of the ECU will be reset to the initial state.

[1]We have confirmed it in multiple real world ECUs.

Bit re-synchronization is a bit level synchronization mechanism. Ideally, all ECUs' bit length are identical, then they will keep synchronized, receivers should observe bus state flips only in its synchronization segment. However, due to the accuracy limitation of physical and environmental factors such as unequal temperature over the bus, different ECUs may have different bit lengths. The CAN bus protocol has to be tolerate and thus allows reasonable minor clock shift among ECUs.

While receiving a frame, a receiver may observe bus state flip outside the synchronization segment, since the transmitter may have an inconsistent bit length. Specifically, the bus status flip (a negative or positive edge) may arrive slightly later than expected, i.e., in the receiver ECU's propagation or phase-1 segment. Or, the bit flip may arrive slightly earlier than expected, i.e., in the phase-2 segment of the previous bit.

The bit re-synchronization mechanism is proposed to compensate for this phenomenon. In every CAN controller, there is a variable named re-Synchronization Jump Width (SJW). If the receiver ECU finds a negative edge outside its synchronization segment, it will adjust its bit length with SJW number of time quanta, in order to keep pace with the transmitter in the following bits. Specifically, if the receiver ECU finds a negative edge in the propagation or phase-1 segment, it learns that its last bit length is shorter than the transmitter's, and thus will lengthen its bit length, by expanding the phase-1 segment with SJW time quanta. Otherwise, if receiver ECU finds a negative edge in the phase-2 segment, it learns that its bit length is longer than the transmitter's, and thus will shorten its bit length, by reducing the phase-2 segment with SJW time quanta.

III. The CANCloak Attack

In this section, we will present a security issue of CAN protocol, and present the CANCloak attack which could exploit this issue to deceive ECUs.

A. Threat model

We assume the target CAN bus follows the latest CAN data link layer and physical signalling standard [7], and at least two ECUs connected to the CAN bus have different sample-point settings. Regarding the defense, we assume no authentication mechanisms have been deployed, but an IDS may or may not be attached to the CAN. This is a common scenario in practice.

On the other hand, we assume the adversary knows sample-points of two victim CAN receivers (which may or may not include an IDS). It is feasible for attackers to get such information, e.g., by disassemble another adversary-owned vehicle of the same model, extract target ECU, then take experiments on it. Moreover, we assume the adversary could attach a malicious ECU which can send crafted waveform to the CAN bus, e.g., via the OBD-II diagnose port. In practice, this is also feasible for attackers.

Given a designated CAN frame ($Frame_A$), goal of CANCloak attack is to let a target victim receiver, $Receiver_A$, accepts $Frame_A$ as intended, and the other receiver, $Receiver_B$, cannot notice $Frame_A$ from the same bus. It is important to make sure that $Receiver_B$ would not raise any receive error during this attack, otherwise it will send an active error frame immediately to interrupt $Receiver_A$.

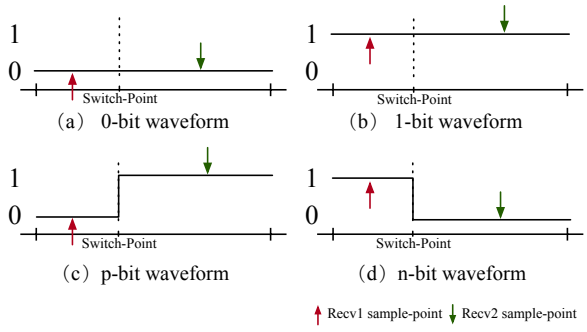

Fig. 4: The adversary transmitter could send a waveform of 0-bit, 1-bit, p-bit and n-bit, by switching bus state between sample-points of two receiver ECUs, to trick them getting different combination of bit values.

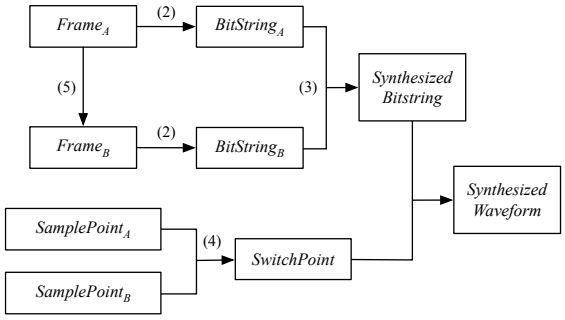

Fig. 5: Design of CANCloak

B. Intuition: One Bit Deceiving

Given that ECUs read CAN bus state only at sample-point, it is possible to craft an ambiguous signal letting receivers decoding it differently, if these receivers have different sample-points.

In a 3-ECU transmission scenario, while one ECU act as transmitter and other two as receivers, without loss of generality, assuming $Receiver_A$ has a sample-point earlier than $Receiver_B$. While these two ECUs are about to receive next bit from bus simultaneously, they would extract same polarity of result from bus, as illustrated by 4 (a) and (b) for 0-bit and 1-bit respectively.

However, if the transmitter of current waveform is malicious, intending to trick two receivers to read different bit values, Adversary could switch the bus state around a specially chosen **switch-point**, which sits in between sample-points of receivers. As shown in Figure 4 (c), the adversary maintain bus state to logic '0' before switch-point of bit, and manipulate it to logic '1' after it. Thus $Receiver_A$ would recognize this bit as '0', while $Receiver_B$ mark it as '1'. Figure 4 (d) shows the opposite scenario. For simplicity, since these abnormal bit introduces positive (logic '0' to '1') or negative (logic '1' to '0') edge to waveform, we denote them as **'p'-bit** and **'n'-bit**

C. Design of the CANCloak Attack

Since a frame is transmitted as a bit string and an adversary is able to deceive ECUs with one bit as discussed in previous sections, she/he could then synthesize an attack waveform bit by bit. Figure 5 shows the overview of the workflow.

Though, there are several issues to address in this attack.

1) Introduce $Frame_B$: As described in III-A, $Receiver_B$ should neither detects $Frame_A$ nor raises any error. Thus, we shall let $Receiver_B$ receives another frame, $Frame_B$, which meets the following basic requirements.

- Has a legitimate CAN data frame structure.
- Is different from $Frame_A$.

2) Construct Bit-Strings: After choosing both frame in application layer, we need to craft their counterpart in data link layer, bit-string of '0's and '1's. Usually, this step is automatically complete by transmitter part of CAN controller hardware. However, those bit-strings would be blind to application layer, which is needed in further phase of our attack. So, we manually finish this step (fill control field and CRC field, etc.) of both frame, resulting in $BitString_A$ and $BitString_B$ respectively.

3) Synthesize Bit-String: Given two bit-strings expected by the victim receivers, the adversary needs to synthesize a special bit string consisting of 0-bit, 1-bit, p-bit, and n-bit, in order to deceive victim ECUs bit by bit.

Table II shows an example of synthesized bit string.

TABLE II: Example Synthesized Bit-String

$BitString_A$	1 0 0 1 1 0 0 0 1 1 0 1 0
$BitString_B$	1 0 1 1 0 1 0 1 0 0 0 1 1
Synthesized BitString	1 0 p 1 n p 0 p n n 0 1 p

4) Determine Switch-Point: Theoretically, the switch-point could be placed anywhere between sample-points of the two ECUs, and would not affect the attack performance. In practice, considering the electromagnetic interference, we set the switch-point around the center of sample-points' gap. We will discuss the success rate of different switch-point settings in Section IV-B.

5) Optimize $Frame_B$: In this section, we will demonstrate how content of $Frame_B$ influence attack performance, and show a practical method of generating an acceptable $Frame_B$.

The synthesized bit-string may introduce some 'n'-bits, which introduce negative edge inside bits. These edges could easily trigger receivers' re-synchronization mechanism. For the receiver having earlier sample-point, it will consider this negative edge come later then sample-point and lay in phase-2 segment of bit, then shorten this bit for SJW. For the other receiver, this negative edge is in propagation or phase-1 segment, thus the bit time would be longer. As a result, there will be a time gap between 2 receivers while stepping into next bit. The cumulative effect of such time gap will confuse receivers and possibly result in receive errors.

To avoid this chaos, we add another requirement:

- By manipulate $Frame_B$, synthesized Bit-string should have minimum number of 'n'-bits, (i.e., 1–>0).

It is easy to eliminate 'n'-bits in most part of frame, because we can arbitrarily manipulate $Frame_B$. In contrast, CRC field of frame is not directly controllable, it is calculated from other fields of the frame. Thus, we forge a large amount of $Frame_B$

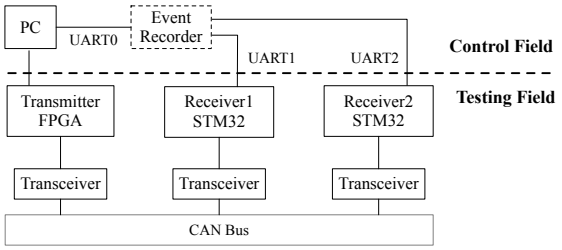

Fig. 6: Illustration of a CANCloak attack scenario. Components in the analyzing field are not needed in real world attacks.

candidates which does not create any 'n'-bit outside CRC field, and elect the one create least number of 'n'-bit in CRC field to be the actual $Frame_B$. Though there are 15 bit in CRC field, in most circumstance, we can find a solution having equal or less than 1 'n'-bit in 10 second, on a modern PC.

D. Implementation of CANCloak Test Bench

To demonstrate this attack, we setup a basic CAN bus environment connected with the malicious transmitter and two victim receiver ECUs. The structure is shown in the testing field in Figure 6.

1) Malicious Transmitter: As the core of this attack, the malicious transmitter is responsible for generating malformed waveform to deceive two receivers. An off-the-shelf CAN transceiver cannot generate crafted waveform directly, because the synthesize CAN frames do not conform the normal CAN bus data frame structure. We implemented a proof-of-concept (POC) malicious transmitter based on an ALTERA EP4CE6 FPGA. FPGA can handle complex low-level logic of CAN transmitter, while concurrently sending precise synthesized waveform.

2) Victim Receivers: To make our testing platform closer to a real-world scenario, the receivers are set up in the most common way. Core of receivers are STM32F429 micro-controllers, integrated with CAN controller peripheral module. In our implementation, a CAN time quanta is 200ns. To meet the most common bit rate of CAN bus, 500 kHz, each bit time should be exactly 2ms, so each bit takes 10 time quantas in our demonstration configuration.

3) Transceivers: Note that, since the I/O ports of the FPGA and the STM32 micro-controllers are not able to handle differential voltage signals on the CAN bus, we use three extra NCV7342 CAN transceivers to connect the FPGA and micro-controllers to the actual cable of the CAN bus.

4) Attack Verification: To verify whether the attack succeeds, we collect the messages received by victim ECUs and compare them with the intention of the adversary. Note that, this part is only for verification purpose, **not** required for real world attacks. The structure is shown in the analyzing field in Figure 6.

The work flow of event recorder is described as follow. Every time a synthesized waveform is sent to CAN bus, event recorder will wait for 50ms to let receivers process it, then ask receivers if they accepted any frame recently. An success attack is count only if $Receiver_A$ accepted designated $Frame_A$, and both receiver did not raise any error. Finally, event recorder uploads all result to PC for further process.

IV. Evaluation

We conducted several groups of experiments on our test environment to evaluate the performance of CANCloak attack. The performance might be affected by the following parameters:

- The *Switch-Point* used to synthesize the attack payload.
- The *Bit Rate* of CAN bus.
- The *Gap of Sample-Points* of victim receivers, determined by their timing configuration.
- The *Content of* $Frame_A$ expected to accept by receivers.

We only change one parameter in each experiment to reveal their influence.

A. Attack Payload Data Set

$Frame_A$ in CANCloak is designated, shall be determined by real attack purpose. In our experiment, $Frame_A$ is an eight bytes data frame whose ID and all data bytes are random. $Frame_B$ is determined by $Frame_A$. We spent 20 seconds finding the best (least 'n'-bit in synthesized frame) $Frame_B$, on a modern laptop with Intel Core i7-8550U CPU. This process was repeated 1000 times. Thus we have a data set consists of 1000 frame pairs after less than 6 hours.

B. Change the switch-point

We choose the switch-point as the variable for the first set of experiments, while all other parameters are controlled. The output port of the transmitter in CANCloak is working at a 50 MHz maximum flip rate. This rate is 100 times faster than the CAN bus bit rate (500 kbps, Bits Per Second). So we have 101 potential switch-point positions (from 0% to 100%). If the switch-point is 0%, the synthesized frame will be identical to $Frame_B$ since there is no space leaving for $Frame_A$. On the contrary, if the switch-point is set to 100%, both receivers will get $Frame_A$.

For more details, we take switch-point as the variable in this experiment and change it from 0% to 100%. The other parameters are configured as follow: bit rate is 500 kbps, sample-points of receives are 40% and 70%, i.e. their ratio of segments' length *Synchronization:Phase-1:Phase-2* are 1:3:6 and 1:6:3 respectively. We tested all 1000 frame pairs in our data set with every switch-point. So there are 101,000 tests in total.

Figure 7 shows the success rate of CANCloak attack, i.e., CANCloak alienate the $Receiver_A$ to get $Frame_A$ and $Receiver_B$ to get $Frame_B$. The valid attack window is from 30% to 70%. The highest success rate is 99.7%, appears when switch-point is 45%. And the success rate is near 100% when switch-point lies in 40% and 60%.

C. Change the Bus Rate

Although CAN standard [7] allows the rate vary from 10 kbps to 1 Mbps, there are two major CAN bus rates usually used, *High-Speed CAN Bus* at 500 kbps and *Low-Speed*

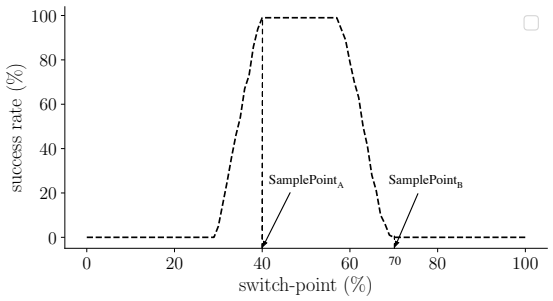

Fig. 7: Attack Success Rate Over Different Switch Point

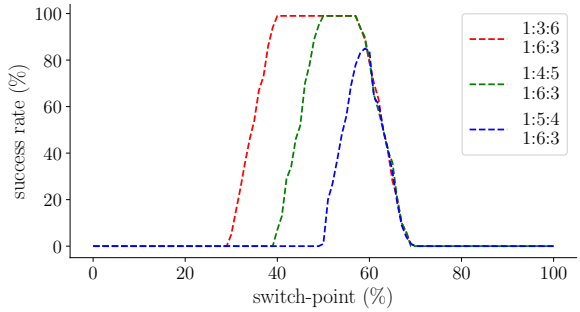

Fig. 8: Attack success rate in different receiver timing settings

CAN Bus at 250 kbps. We select four representative rates for evaluation, the maximum rate (1 Mbps) and the minimum rate (10 kbps), and two commonly used rates (250 kbps, 500 kbps).

Other parameters are the same as the previous experiment. While switch-point varies from 0% to 100% for each bit rate. sample-points of receivers is 40% and 70%, All frame pairs in our data set are tested on every switch-point, every bus rate. Thus 404,000 tests in total were performed.

The result is omitted because the success rate curve remains the same at different bus rates, indicating that CANCloak's performance stable over different bus rate settings.

D. Change the Gap Between Receivers' Sample-Points

Timing parameters of different ECUs, like the length of time quanta or segments, may different, though they are connected to the same bus. For $Receiver_A$, 3 sets of timing parameters are used, which are 1:3:6, 1:4:5 and 1:5:4, i.e., sample-point are 40%, 50% and 60% respectively. For $Receiver_B$, the sample-point is fixed at 70% with the timing parameters of 1:6:3. Other parameters are still the same as the previous experiment. The bus rate is 500 kHz, The payloads are all frame pair from our data set, The switch-point varies from 0% to 100% for each sample-point setting. So there are 303,000 tests in total.

Figure 8 shows how attack success rate varies with 3 different sample-point settings. In general, the more deviation these two receivers have in the sample-point settings, the wider attack window adversary could have. In the first configuration, there are 30% time interval between two receivers' sample-points. In this case, we have a 20% width perfect attack window, which starts at 40% (i.e., identical to sample-point of $Receiver_A$) and ends at 60% (i.e., a time quanta (10%) ahead of $Receiver_B$'s sample-point). The perfect attack window drops to 10% and 0 for the second and third settings, respectively.

In summary, the length of the perfect attack window of switch-point is one time quanta less than the gap between the sample-points. The ideal attack window starts at the sample-point of $Receiver_A$, and ends at one time quanta before sample-point of $Receiver_B$.

V. Conclusion

In this paper, we propose a new type of attack, CANCloak, which takes advantage of the deviation between two CAN ECUs. We then design and implement CANCloak attack. As far as we know, CANCloak is the first attack against CAN physical layer. Furthermore, we evaluate the overall performance of CANCloak attack by changing sample-point of CAN receivers, bit rate of bus, switch-point settings, and attack payload. The results show that CANCloak attack has up to 99.7% of success rate in certain conditions.

References

[1] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

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

[3] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "Canauth-a simple, backward compatible broadcast authentication protocol for can bus," in *ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, 2011, p. 20.

[4] E. Wang, W. Xu, S. Sastry, S. Liu, and K. Zeng, "Hardware module-based message authentication in intra-vehicle networks," in *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2017, pp. 207–216.

[5] M. Marchetti and D. Stabili, "Anomaly detection of can bus messages through analysis of id sequences," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1577–1583.

[6] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *2016 international conference on information networking (ICOIN)*. IEEE, 2016, pp. 63–68.

[7] "Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling," International Organization for Standardization, Standard, Dec. 2015.

[8] E. C. De Oliveira, "Electrical architectures and in-vehicles networks," SAE Technical Paper, Tech. Rep., 2007.

[9] U. Kiencke, S. Dais, and M. Litschel, "Automotive serial controller area network," *SAE transactions*, pp. 823–828, 1986.

[10] R. Bosch *et al.*, "Can specification version 2.0," *Rober Bousch GmbH, Postfach*, vol. 300240, p. 72, 1991.