Security Attacks to the Name Management Protocol in Vehicular Networks

Sharika Kumar
The Ohio State University and
Accelera by Cummins Inc.
kumar.918@buckeyemail.com

Imtiaz Karim
Purdue University
karim7@purdue.edu

Elisa Bertino
Purdue University
bertino@purdue.edu

Anish Arora
The Ohio State University
anish@cse.ohio-state.edu

***Abstract*—Trucks play a critical role in today's transportation system, where minor disruptions can result in a major social impact. Intra Medium and Heavy Duty (MHD) communications broadly adopt SAE-J1939 recommended practices that utilize Name Management Protocol (NMP) to associate and manage source addresses with primary functions of controller applications. This paper exposes 19 vulnerabilities in the NMP, uses them to invent various logical attacks, in some cases leveraging and in all cases validating with formal methods, and discusses their impacts. These attacks can–① stealthily deny vehicle start-up by pre-playing recorded claims in monotonically descending order; ② successfully restrain critical vehicular device participation and institute a *dead beef attack*, causing reflash failure by performing a replay attack; ③ cause stealthy address exhaustion, *Thakaavath*–exhaustion in Sanskrit, which rejects an address-capable controller application from network engagement by exhausting the usable address space via pre-playing claims in monotonically descending order; ④ poison the controller application's internally maintained source address-function association table after bypassing the imposter detection protection and execute a stealthy SA-NAME Table Poisoning Attack thereby disable radar and Anti Brake System (ABS), as well as obtain retarder braking torque dashboard warnings; ⑤ cause Denial of Service (DoS) on claim messages by predicting the delay in an address reclaim and prohibiting the associated device from participating in the SAE-J1939 network; ⑥ impersonate a working set master to alter the source addresses of controller applications to execute a Bot-Net attack; ⑦ execute birthday attack, a brute-force collision attack to command an invalid or existing name, thereby causing undesired vehicle behavior. The impact of these attacks is validated by demonstrations on real trucks in operation in a practical setting and on bench setups with a real engine controller connected to a CAN bus.**

I. Introduction

With the automotive industry expanding rapidly into vast smart mobility ecosystems, new levels of cyber sophistication have been reached today. The transformation, however, has introduced new cybersecurity risks as proven by the exponential increase in the magnitude, frequency and sophistication of cyber attacks over the last decade [1]. UNECE WP.29 R155 [2] and R156 [3] regulation enforcement linked to ISO/SAE 21434 [4] and ISO/SAE 24089 [5] standards effectively mitigate automotive-specific cyber risks and are aggressively being adopted by automotive industries. These guidelines eschew guidance on the exact processes to be followed and avoid recommending specific solutions but instead heavily emphasize risk analysis, life-long cybersecurity threats, and vulnerability management. The vehicular communication security risks emerging from ISO/SAE 21434 assessments are not mitigated on commercial vehicles due to Society of Automotive Engineers J1939 (SAE J1939) [6]–[15] security limitations. An attacker can exploit the SAE-J1939 vulnerabilities widely adopted in MHD vehicular communication to induce undesired vehicle behaviors.

Vehicle Network Attacks. The openness of the SAE J1939 recommended practices gives easy access to replicate known consumer vehicle safety-critical attacks [16]. The support of the right to repair via On-Board Diagnostics II (OBD-II) port on MHD vehicles allows for extracting vehicle diagnostic information. Efforts to limit diagnostic tool functionalities by minimum privileges by access control methods emerged as adversaries started plugging malicious devices to tamper critical vehicle functionalities, [17]. Just by attaching a cheap CAN-capable device to the OBD-II port, the adversary attains access to a critical vehicular SAE-J1939 CAN network. CAN being inherently insecure, researchers have exploited weaknesses of CAN and have demonstrated stealthy selective denial-of-service attacks [18]. Authentication defenses to secure intra-vehicle CAN communication, such as watermarking [19], freshness, and integrity headers [20], still lack industry-wide adoption.

Motivation and Goal. In view of the fact that security was not a primary design consideration in SAE-J1939 protocols and with no formal protocol verification conducted in prior research, there is a lack of systematic security analysis in SAE-J1939 protocols. Vehicle manufacturers and component makers urge following industry standards for easy integration for interoperability, and there is no industry standard method (yet) to achieve security for onboard MHD communications and no SAE-J1939-compatible way (yet) to perform it. On SAE-J1939 networks, controller applications announce their source address and primary function following SAE-J1939 81 recommended practice [14] without any authentication. In this paper, we aim to systematically analyze the SAE-J1939-81 Name Management Protocol (NMP) that defines and influences the primary functions of controller applications to identify security gaps and their possible exploits.

Contributions. In this paper, we characterize vulnerabilities

Symposium on Vehicles Security and Privacy (VehicleSec) 2024
26 February 2024, San Diego, CA, USA
ISBN 979-8-9894372-7-6
https://dx.doi.org/10.14722/vehiclesec.2024.23004
www.ndss-symposium.org

of the NMP of SAE J1939 networks. *To the best of our knowledge, we are the first to perform such an examination on MHD vehicular protocols.* With vulnerability analysis of NMP, we uncover 19 new stealthy attack vectors. Our analysis is bolstered by LTL-based formal model checking of these attack vectors on NMP. To this end, we create 3 formal models of the different forms of NMP by solving several modeling challenges and using a state-of-the-art model checker to validate and uncover the possible addressing schemes systematically. We validate and verify attacks on a bench setup with a real engine controller connected to a CAN bus as well as on a real-world truck. On the bench setup, our technique to predict the reclaim time creates repeated collisions with claim messages to suppress a controller application from network participation. On a moving truck, we disable the radar with our stealthy automated source address-function association table poisoning attack, thereby incapacitating vehicle features such as lane change assistance, parking aid, collision mitigation, blind spot detection, and rear cross-traffic alert. Disabling such features can have a catastrophic impact on the security of the vehicle. For instance, in case the ABS is disabled with our attack, the victim vehicle operator would be required to adjust their braking pressure to prevent the tire from being locked up instead of providing constant full braking or hard-breaking pressure.

Responsible Disclosure and Open-source. We have responsibly disclosed the findings of our work to the standardization body and are actively cooperating with them for mitigation. The models and related research artifacts are open-sourced at: https://github.com/MasterTigress2020/J1939_81_NMP_Attacks

II. Background

A. Name Management Protocol

ISO/OSI Layers	J1939 Standard
Application Layer	SAE J1939/71 - Vehicle Application Layer SAE J1939/73 - Application Layer Diagnostics SAE J1939/81 - Network Management
Presentation Layer	-
Session Layer	-
Transport Layer	SAE J1939/21,22 - Datalink Link Layer
Network Layer	SAE J1939/31 - Network Layer
Datalink Layer	SAE J1939/21 - Datalink Link Layer
Physical Layer	SAE J1939/11 - Physical Layer - 250Kbits/s SAE J1939/12 SAE J1939/14 SAE J1939/15 Reduced Physical Layer, 250 kbits/s

TABLE I: SAE J1939 ISO/OSI Layers

SAE J1939 is the recommended practice for communication and diagnostics between vehicle components in the car and heavy-duty truck industry. Originating in the United States, SAE J1939 is now widely used worldwide and can support up to 254 logical nodes. On a typical bit rate of 250 kbps or 500 kbps, the protocol also supports control and measurement value transfers. As shown in Table I, the SAE J1939 standard is split into multiple standards [6]–[15] according to the OSI reference model with session and presentation layers unspecified like most of the field bus protocols.

Overview. The SAE J1939/81 standard [14] defines messages that can be used by connected controller applications to acquire and maintain a network address on an SAE J1939 communication network by specifying the address arbitration process. The network is managed by associating source addresses with the primary function of the controller application. This standard also specifies the initialization process, reaction requirements to power outrages, and detection and reporting of the source address contention. For network management, a set of messages is used. Each message uses a specific Parameter Group Number (PGN), as shown in Table II. On SAE-J1939 networks, messages can be broadcast messages or destination-specific messages (P2P) as indicated by the transmitting Controller Application (CA) in the PDU-specific field of the 29-bit CAN identifier. If the value of the PDU-specific field is set to 255 the message shall be a broadcast message that can be received and processed by all connected CAs. Alternatively, if the PDU-specific field is set to a destination address of CA on the network, this message shall be detected and processed by the targetted CA. Though address announcement message PGN 60928 is recommended [14] as broadcast, P2P claims are as well processed by CAs. Various acronyms used throughout this paper are tabulated in Table III

Message Name	PGN	Payload
Request PGN	59904	PGN 60928
Address Claimed Cannot Claim Source Address	60928	NAME
Commanded Address	65240	NAME, new SA
Name Management	37632	Name fields and mode

TABLE II: Network Management Messages

Acronym	Expansion
MHD	Medium and Heavy Duty
NMP	Name Management Protocol
ABS	Anti Brake System
DOS	Denial of Service
SAE	Society of Automobile Engineers
OBD II	On-Board Diagnostics II
PGN	Parameter Group Number
CA	Controller Application
SACCA	Single Address Capable Controller Application
MSB	Most Significant Bit
DA	Digital Annex
AACCA	Arbitrary Address Capable Controller Application
RAM	Random Access Memory
TPMS	Tire Pressure Monitoring System
CIA	Confidentiality Integrity, and Availability
MAC	Machine Authentication Code
CMAC	Cipher-based Message Authentication Code
HMAC	Hash-based Message Authentication Code
FSM	Finite State Machine
SMV	Symbolic Model Verifier

TABLE III: Term Reference

Finite State Machine. A CA on the SAE-J1939 network follows the SAE-J1939-81 arbitration process to determine the source address that it can claim (SACCA initialization state transitions as shown in Fig. 2). An 8-bit source address within the CAN identifier serves as the source address. A CA uses the Address Claimed message (PGN 60928) indicating the NAME it prefers to operate on (shown in Fig. 1). A unique 64-bit NAME transmitted in data field of a claim consisting of 10 fields provides the primary function that the CA has on the network. On power-up and receipt of the request message for Address Claimed, a CA shall transmit an address

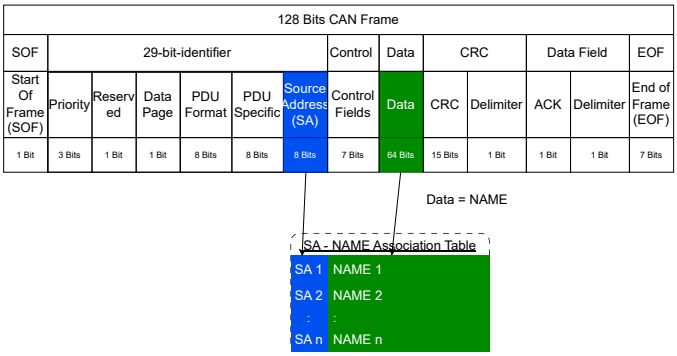


Fig. 1: SA-NAME Association Table derived from a Classic-CAN Address Claim Frame

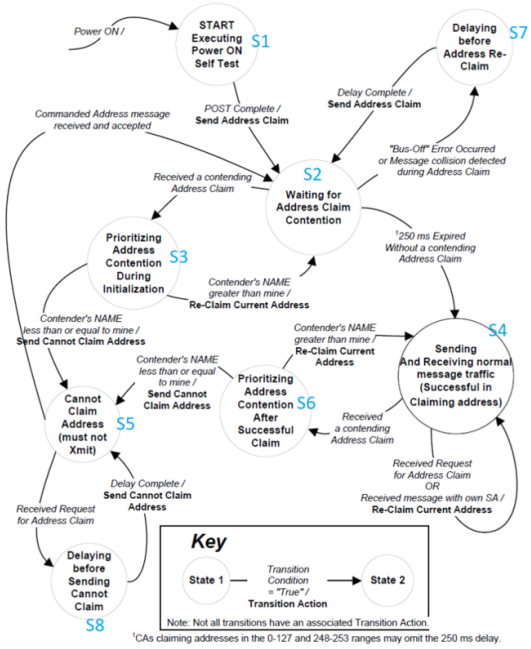


Fig. 2: SAE-J1939-81 State Transition Diagram for Initialization of Single Address Capable CAs

claim message announcing its source address and NAME, i.e., transitioning from state *S1* to *S2* as indicated in Fig. 2. An address contention is resolved by arbitrating over NAME at *S3* or at *S6*, where lower NAME has higher priority. Any CA unable to claim an address shall announce its inability on the Cannot Claim Address message and navigate from *S2*, *S3* to *S5*. If no contending claim is received, a CA shall start trans-receiving communication messages, transiting from *S2* to *S4*. Every CA constructs an SA-NAME association Table that associates its NAME and source address to correlate the critical function of a CA to an address.

Address Configuration and Capability. A particular CA determines the source address to use for Address Claim via the address configuration method. Two primary capabilities– Single Address Capable and Arbitrary Address Capable are defined for the Address Claim process. The primary capability can be distinguished as indicated in NAME's Most Significant Bit (MSB). Single Address Capable CA (SACCA) can only claim a single source address typically assigned by the SAE-J1939 committee published as J1939 Digital Annex (DA) document. SACCA can be classified as non-configurable, service configurable, commanded configurable, and self-configurable. Non-configurable CA has a fixed permanent address that can be altered only by a software update. The service tool can change Service configurable CA's source address using commanded address messages. The source address of Command configurable CA's is commanded during vehicle power-up without the intervention of a service tool. Based on the vehicle configuration at a particular position of the vehicle, self-configurable CA's determine internally which source address it can use from a limited set of source addresses. Arbitrary Address Capable CA (AACCA) can select its source address within the 128 to 247 inclusive range. AACCA can re-calculate and re-claim an unused address in case of an address conflict.

B. Vulnerabilities

Among the various communication protocols found in automotive networks, such as LIN, FlexRay, CAN-FD, Automotive Ethernet, etc., the most widely used is the CAN-based multi-master serial. The recommended vehicle bus practice SAE J1939 is found wherever a diesel engine exists. The CAN vehicle bus standard is a message-based protocol originally designed to save on copper without taking security into account. Various security risks, such as lack of authentication, inadequate integrity checks, and insecure arbitration schemes are inherently present in the CAN vehicle bus standard.

SAE J1939 over CAN was designed assuming an honest/trusted vehicular environment. Vulnerabilities that open up due to this assumption are perceptible in all SAE-J1939 recommended practices.

In the case of the SAE-81 Name Management Protocol, the protocol allows for an SA-NAME association if an interval of 250 ms has expired without a contending address claim in the absence of a strong security authentication. This permits an adversarial CA to issue fake claims, segregate CAs, poison NAME-SA association tables in CA's and make the network unusable by pre-claiming addresses. The recommended practices only address address contention and do not address NAME contention. This breaks the primary goal of NAME management protocol to corroborate that NAMEs of all CAs intended to transmit on a particular network are unique. This key responsibility earmarked on integrators of networks and manufacturers of ECUs by SAE-81 results in several adversarial possibilities of misuse that may lead to undesired/unsafe vehicle behavior. SACCAs that only have the ability to claim preferred addresses can be silenced by replaying the claim at a higher priority, as shown by the exploit reported in [21]. While AACCA can mitigate such a threat by providing the flexibility of claiming on a different address upon arbitration loss, we observe that address exhaustion by forged claims leaves AACCA with no address to claim on. The SAE-J1939-81 recommended practices prescribe including a NAME checksum byte computed on the target CA's original NAME as a security check to ensure that the correct CA receives the command message. However, as the checksum has no associated key, it can easily be matched up by an adversary

and, hence is inadequate to perform the security check required by SAE J1939-81.

C. Model Checking with Linear Temporal Logic (LTL)

Model checking is a common verification technique used to analyze control systems [22], hardware [23], and communication protocols [24]. There are two inputs: a given property ϕ (*e.g.,* a controller always receives an address) and a model $\mathcal{M}$ to be verified. Generally, the property can be expressed in linear temporal logic [25], which describes the relative or absolute order of behaviors in the verification system (*e.g.,* the next state denoted by **X**, the subsequent path denoted by **F**, and the entire path denoted by **G**). The target for LTL-based model checking is whether the property ϕ holds in $\mathcal{M}$ under the semantics of temporal logic ($\mathcal{M} \models \phi$). An LTL model-checking algorithm is a decision procedure that, given a model $\mathcal{M}$ and an LTL formula ϕ, returns the answer "yes" if ($\mathcal{M} \models \phi$), and "no", plus a counterexample, if ($\mathcal{M} \not\models \phi$). To perform model checking, the algorithm essentially negates the whole property ($\neg\phi$) and constructs the correspondence model $\mathcal{M}'$. It then searches the state space (i.e., the set of system execution paths $\mathcal{P}$) for the existence of an intersection ($\mathcal{P}(\mathcal{M}) \cup \mathcal{P}(\mathcal{M}')$). If the intersection exists, then the property is not verified, and a path from the initial state to the intersection is provided as a counterexample; otherwise, the property ϕ is satisfied. We use model-checking in this context to analyze and validate the attacks on the NMPs systematically.

III. Attacks on the Name Management Protocol

Our security analysis of NMP revealed several vulnerabilities that were carefully exploited to construct new attack vectors. This section describes the threat model and manifold attacks and their mitigation.

A. Threat Model

For the purpose of this work, we assume that the adversary has direct access to the CAN bus via a public OBD port. The adversary's capabilities are limited by computational power attributed to the number and speed of the processors on the device, the amount of Random Access Memory (RAM), and the efficiency of the software used to execute the attacks. This device that is physically attached to the bus can be a compromised onboard device, a compromised infotainment unit [26], [27], a compromised datalogger, or a diagnostic tool where a Man-In-The-Middle shim is inserted [28]. The device can also be remotely connected to wireless interfaces on the vehicle bus, such as the edge units, the telematics units, wireless capable dataloggers, Bluetooth-capable diagnostic tools, or a Tire Pressure Monitoring System (TPMS).

B. Attacks

Combining the power of human inspection and formal model checking, security vulnerabilities found are constructed into attacks. Our human security analysis focused on compromising cyber security properties such as Confidentiality, Integrity, and Availability (CIA) that can lead to vehicular damage scenarios. The uncovered attack vectors, their action sequences, and the violated security properties are tabulated in Table IV.

1) Pre-Play, Replay, and Dead Beef Attacks: In conformity with SAE J1939-81, if two CAs contend for an address, the CA with the equal or lower numerical value of the NAME shall have higher priority and win the address arbitration. In these attacks, the adversary impersonates a CA with equal NAME and wins address contention, effectively restraining a legit CA from network participation.

Attack steps. In this attack, the attacker records address claims to discover connected CAs SAs. On the power cycle, the attacker *Pre-Plays* the claims monotonically in descending order (ex: A-1, B-2, C-3) at state *S1* prior legit claims. Alternatively, attacker *Replay* claims post legit claims or during reflash to force the CA state from *S1, S2, S3* to *S5*.

Impact. The primary impact of the attack is to restrain the network participation of critical vehicular devices stealthily. Replaying claims during software download of a CA to institute *Dead Beef* attack would leave the CA in a non-operational state, requiring rebooting or re-downloading of software to recover. As NMP is excluded from SAE J1939-22 [10] and proposed SAE J1939-91C, these attacks persist even on secured CAN-FD networks as well.

Mitigation. An inter-CA authentication mechanism with a key agreement scheme would prevent unauthorized claims from affecting CA network participation. This can be realized by including NMP in SAE J1939-91C to perform key agreement and authentication. Also, if NMP is addressed in secured SAE J1939-22 networks, these attacks can be prevented on CAN-FD networks.

2) Thakaavat (Address Exhaustion) attacks: An AACCA, on address conflict, can select a source address from 128 to 247 inclusive at *S3* by re-claiming. In this attack, the adversary claims on every address in 128 to 247 inclusive, disallowing a legit AACCA to claim on. In scenarios where SACCAs and AACCAs contend, AACCAs shall lose arbitration as setting the "Arbitrary Address Capable" bit in its NAME decreases its priority for winning arbitration. An adversary can replay claims with *Bit-Flip* to deny an AACCA on the network.

Attack steps. In this attack, the attacker Pre-play claims in monotonically descending order with all addresses in the 128 to 247 inclusive range, forcing legit AACCA to lose arbitration, send a "cannot claim address" message, and move to state *S5*. As this attack exhausts all the 120 addresses, we call it *Thakaavat*, which in Sanskrit means exhaustion. Alternatively, the attacker can leave out any one address while performing *Thakaavat*, forcing the legitimate AACCA to take the left-out address and move to *S4*. We name this attack *Magical - Pick a Card, Any Card Attack*. If the AACCA is command-capable, the adversary can command an AACCA to take a specific SA. In this case, they can Replay an AACCA claim with a bit flip at the "Arbitrary Address Capable" bit by impersonating a SACCA, forcing the AACCA to lose arbitration. By latching on to the AACCA claims and performing a *Bit-Flip* attack, a *Thakaavat* attack is executed, leaving CA's at state *S5*.

Impact. The impact of this attack is a stealthy denial of an AACCA from network participation. Furthermore, adversarial control on the SA of an AACCA helps avoid the attack step of network discovery.

Mitigation. Effective mitigation of these attacks can be

achieved by restricting fraudulent claims by performing key agreement and authentication schemes.

3) SA-NAME Table Poisoning Attacks: When a CA transmits the Address Claimed message, all CAs compare or record this newly claimed address to their SA-NAME mapping table. In this attack, the adversary poisons the SA-NAME mapping table to break the SA-NAME correlation, thereby creating undesired vehicle behavior. SAE-J1939 recommends each CA detect one or multiple source address imposters and alert the network. An adversary can bypass this imposter detection mechanism by P2P claims.

Attack steps. The adversary claims on the same NAME as a legitimate node but on a different SA for this attack. We executed SA-NAME Table poisoning attacks by swapping identities using P2P poisoning also bypassing imposter alert messages.

Impact. As an effect of the attack, a CA that processes by NAME cannot distinguish between legitimate and adversarial claims, given that both have won arbitration without contention and have reached state *S4* from *S2*, creating unauthorized vehicle behavior. This creates inconsistencies between the real and the SA-NAME association table maintained in a receiver CA at state *S4*.

Mitigation. SA-NAME Table Poisoning attacks can be mitigated by authenticating the address claim messages instead of arbitrating by the NAME field. The imposter alert bypass can be prevented by restricting the processing of address claim messages to global addresses only.

4) DOS with Collision Attack: CAN is based on CSMA/CD+AMP, where collisions are resolved through a bitwise arbitration based on a pre-programmed priority of each message in the identifier field of a message. As simultaneous address claim messages shall result in collisions and further bus-off, SAE-J1939-81 specifies adding a pseudo-random transmit delay to avoid collisions by retaining a CA at *S7*. In this attack, the adversary intentionally inserts a claim aimed to collide with a legitimate claim, denying a legitimate CA to claim.

Attack Steps. The attacker exploits the predictive nature of the pseudo-random delay from weak entropy sources to intentionally create collisions by duplicating the legitimate claim at the exact time, forcing CA's to stay at state *S7*.

Impact. Due to this attack, an adversary can deny a legitimate CA from network participation.

Mitigation. Collision attacks can be mitigated using non-repeatable true random numbers within the CAs or secured pseudo-random number generators.

5) Bot-Net and Commanded Addressing Attack: A network interconnection CA, a bridge, or a diagnostic or scan tool may command another CA to use a given source address with the "commanded address" message without any security authentication. As SA commands are unauthenticated, an adversary can impersonate a commanding CA and assign SA to one or multiple CAs.

Attack Steps. In this attack, an attacker exploits the lack of authentication, impersonates a commanding CA, and alters the SA of CAS on the network. Alternatively, the adversary can override a legitimate command by message injections to an invalid address of 0xFE. Post failing a commanded addressing sequence, the adversary can act as a commanded CA to suppress fault indications at the vehicular driver. Additionally, as SAE-J1939-81 does not contend on NAME, an attacker can impersonate a "working set master" by claiming the same NAME on a different address, still complying with SAE-J1939-81 recommended practices.

Impact. The CA that understands the working set in this scenario shall update the SA-NAME association table with the attacker's SA and receive working set communications from the attacker.

Mitigation. This class of attacks can be mitigated by authenticating the "commanded address" message and adding NAME contention mechanisms in [14] standard.

6) Birthday Attack: The NAME checksum contains the arithmetic sum of the 8 bytes of the target CA's original NAME truncated to 8 least significant bits. This security check ensures that the correct CA receives the command message to protect against the possibility of changes to the SA having through the address arbitration process. In this attack, the attacker brute forces the 8-bit check iterating from 0 to 255 to assign an invalid or an existing CA NAME.

Attack Steps. Exploiting the inadequacy of using checksums for security use cases, the attacker can brute force a birthday collision attack to command multiple CAs with incorrect NAMEs. As the checksum used is truncated, it eases the attack execution by reducing the collision space.

Impact. The attack can cause undesired vehicle behavior due to an incorrect CA NAME assigned.

Mitigation. This attack can be mitigated by replacing the checksum with a Machine Authentication Code (MAC) to authenticate NAME strongly.

7) Source Address Alteration Attack: As already mentioned, checksums are not recommended for security verification as they can be easily matched up due to the lack of key association. This gap is exploited in this attack where adversary easily matches up checksum to assign a new CA NAME.

Attack Steps. The attacker impersonates a CA match-up checksum and alters the NAME of the CA.

Impact. As NAME uniquely identifies the primary function of a CA, altering NAME can create disassociation between CA's actual primary function and the NAME assigned, further resulting in confusing the receiver.

Mitigation. To prevent adversaries from altering SA's, the checksum could be replaced by a Cipher-based Message Authentication Code (CMAC) or Hash-based Message Authentication Code (HMAC) along with agreed pre-shared keys.

IV. Validation with Formal Verification

In this section, we delve into the validation of the attacks using formal verification.

TABLE IV: Catalog of New Attacks

No.	Technical Requirement	Attack	Impact	Sequence of Action	Violation
1	CA with $\leq$ NAME wins arbitration	**Pre-play** chained recorded claims prior to legit claim	Stealthily deny CA from network participation	(S1-S2-S3-S5)$^+$	Availability
2	CA with $\leq$ NAME wins arbitration	**Replay** (=NAME) recorded claims within 250ms	Stealthily deny CA from network participation	(S1-S2-S3-S5)$^+$	Availability
3	CA with $\leq$ NAME wins arbitration	**Dead Beef Attack**: Replay (=NAME) recorded claims during CA reflash	Interrupted reflash, needs reboot or reflash to recover	S1-S2-S3-S5	Availability
4	AACCA can select SA in the range of 128 to 247	**Thakaavat Attack**: Pre-play chained claims (128 to 247)	Stealthily deny legit AACCA from network participation	S1-(S2-S3)$^+$-S5	Availability
5	AACCA can select SA in the range of 128 to 247	**Pick a Card, Any Card Attack**: Pre-play chained claims with any one SA left out (128 to 247)	Attacker stealthily assigns AACCA's SA, reduces CA SA ambiguity	S1-(S2-S3)$^+$-S4	Integrity
6	SA of CCACA can be commanded	Prior legit claims, chain pre-play claims (128 to 247), command at S5	Attacker stealthily assigns AACCA's SA, reduces CA SA ambiguity	S1-(S2-S3)$^+$-S5-S2-S4	Integrity
7	SACCA wins over AACCA as "Arbitary Address Capable" bit is set	Repeat replay AACCA claim with "Arbitrary Address Capable" bit flip	Effective stealthy address exhaustion denies AACCA on network	S1-(S2-S3)$^+$-S5	Integrity
8	Each CA owns a unique NAME and unique SA	Replay recorded claims on new SA	Legit CA processes messages from new SA	S1-S2-S4	Integrity
9	Claims are sent on global address (255)	Use P2P claims to be invisible to legit CA but audible to receiver CA	SAE J1939 imposter alert mechanism bypass	S1-S2-S4	Integrity
10	Each CA owns a unique NAME and unique SA	**SA-NAME Table Poisoning Attack**: Disassociate/swap identities using P2P claims	Poison NAME-SA association table. Indirect CA silencing	S1-S2-S4	Integrity
11	Re-claim post a random delay to avoid a collision	**Collision Attack**: Exploit delay prediction to cause collision	Deny legit CA from network participation	S1-(S2-S7)$^+$	Availability
12	SA of CCACA can be commanded	Impersonate commanding CA to command CAs SAs	Undesired vehicle behavior	S1-S2-S4	Integrity
13	Change in SA of the Working Set Master shall NOT change the definition of the Set	**Bot-Net Attack**: Claim on the same NAME with new SA to impersonate as "Working Set Master"	Users update SA association to NAME via attack claim, redirecting members to adversary	S1-S2-S4	Integrity
14	SA of CCACA can be commanded	Command CCACA's SA as 0xFE(invalid), mock CCCA messages	Stealthily denial of CCCA from network participation	S1-S2	Integrity
15	NAME checksum security verify on commanded CA's	**Birthday Attack**: Brute force checksum collision to change CA NAME	Undesired vehicle behavior due to incorrect NAME assignment	S1-S2-S4	Integrity

TABLE V: Catalog of New Attacks - Contd.

No.	Technical Requirement	Attack	Impact	Sequence of Action	Violation
16	Only the most recent tool can issue adopt command successfully	Match-up checksum to change CA NAME post legit tool command	Undesired vehicle behavior due to incorrect NAME assignment	S1-S2-S4	Integrity
17	Each CA on the network may be a member of only one Working Set	Replace the Working Set definition via "Working Set Master" impersonation	Undesired vehicle behavior	S1-S2-S4	Integrity
18	Working Set Masters re-defines Working Set with "Working Set Master" message with data of zero	Transmit "Working Set Master" with all zero messages implying the purpose of the Working Set is finished	Makes the Working Set useless leading to undesired vehicle behavior	S1-S2-S4	Integrity
19	Users shall create the Working Set with a total number of members specified	Exhaust the number of members of working set	SA members can not join the Working Set anymore	S1-S2-S4	Integrity

A. Formal verification with model checking

To validate the attacks, we formally verify the addressing schemes. For this, we build formal models for the addressing schemes and identify several safety and liveness properties, including the crucial property that the addressing scheme will always reach the state where the CA will have a valid address. Violating this property means there are ways an attacker can stop a CA from getting a valid address. Then, we use the model and properties with a model checker to ensure the properties hold; otherwise, we get a counterexample trace for an attack. Model checking with the formal model guarantees that, within the state space of the model, all possible scenarios or attacks that may cause the addressing scheme to hinder are discovered. In the following, we details each such step and discuss the technical challenges we faced and how we resolved them.

B. Modeling

For the formal modeling we go through the SAE J1939-81 recommended practices [14] and model the three addressing schemes: single-addressing, arbitrary addressing, and command-configurable in the SMV [29] formal language as finite-state-machines (FSMs). Symbolic Model Verifier (SMV) is a seminal language introduced to model synchronous, asynchronous systems, and network systems formally. We model two CAs that are communicating with a shared channel. Modeling two CAs is enough for us to test the safety and liveness properties that need to be ensured. We assume the shared bus that connects the CAs is under the influence of a Dolev-Yao adversary [30]. Following this adversary model, the attacker can inject, drop, and modify any communication between the two CAs. Dolev-Yao is a very popular threat model that has been used in analyzing multiple protocols, and the adversarial capabilities also suit our modeling.

Challenges of modeling. While modeling the addressing schemes, we face several challenges.

(C1) *Modeling NAME variable:* Every CA that transmits messages on a SAE J1939 network shall have a NAME. This NAME is a 64-bit identifier for a CA and is composed of 10 fields. These fields include industry groups, vehicle systems, and others. This NAME is communicated between CAs and may be formally modeled as a stream of bits. However, for a 64-bit NAME stream, the search space of the model will be 2^{64} states, which hits the scalability of the model checker.

(C2) *Address space:* For each CA the address space is between 0 to 254. Therefore, for modeling multiple CAs this essentially implodes the states of the model.

Insights on the challenge solutions. Here, we discuss how we solve the discussed challenges. To solve (C1), we examine the design documents thoroughly and observe that though the NAME is a 64-bit variable, they are divided into dependent and independent fields. The independent fields are arbitrary address capable, industry group, function lower, and manufacturer code. Therefore, we can essentially model only these four fields with a sequence variable. This gives us a scalable solution to test the properties. To resolve (C2), we downgrade the address variables to smaller values(i.e., 5) to efficiently capture the property space without causing a state space exploration. As we do not have any property that specifically reasons about a specific address value, this downgrade does not affect the soundness and completeness of our model.

C. Properties

For properties, we consult the recommended practices and find out invariants that need to be ensured by the addressing scheme. One of the most important invariants is: *the addressing scheme should reach the designated state where the CA is assigned an address for receiving and sending traffic.* In total, we test 5 safety and liveness properties, and these properties are converted to formal SMV language as well.

D. Model checking

For modeling checking, we use NuXmv [31]. NuXmv is a popular model checker that uses state-of-the-art algorithms to verify. We utilize LTL-based model checking and encode the properties in linear temporal logic. We pass the model and property to the model checker. In case the property is violated, we get a trace of a vulnerability through a counterexample. We then modify the property to ensure the same counterexample is not generated again. This process is repeated until the model checker returns the property as true.

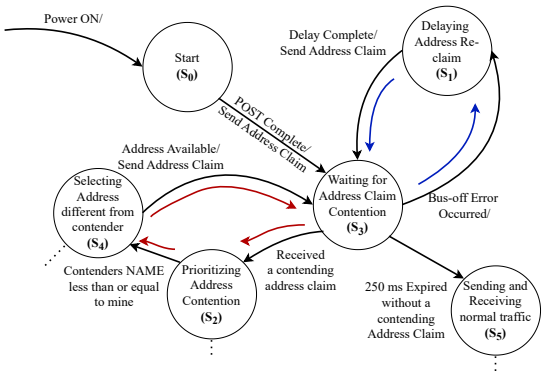


Fig. 3: Simplified FSM for Initialization of Arbitrary Address Capable CAs (AACCA); the convention for transitions is "condition/action"

To show this process in a running example, we focus on the simplified partial FSM of the arbitrary addressing scheme (shown in Figure 3). For the property, we use the previously discussed liveness property *the addressing scheme should reach the designated state where the CA is assigned an address for receiving and sending traffic.* The property is converted to a LTL property as $F(State = S_5)$. This essentially encodes that in every run of the scheme, the protocol should move the S_5 state *Sending and Receiving normal traffic*. Any violation of this property, in other words, the counterexample trace, is a sequence of messages that forces the scheme to move to a state where the CAs do not get any more addresses. Providing this property and the formal model to the model checker, in most cases, generates a counterexample, where a bus-off-error is caused repeatedly, and the addressing scheme is stuck in the $S_3 \rightarrow S_1 \rightarrow S_3$ states (shown with the blue arrow in Figure 3). We then modify the LTL property as $G(!Bus-off-error$ & $!delay-complete) \rightarrow F(State = S_5)$ so that this counterexample is not generated again. Following a second run of the model checker, we get another counterexample, now related to the case where there is a contending address claim, the contender NAME being less than equal to the CA, and there are no addresses available, the scheme will never reach state S_5 (shown with red arrows in the Figure 3). This process continues until there are no more counterexamples and issues with the addressing protocol, and the property is declared true. As shown, model checking provides a systematic way of uncovering all the possible issues of the addressing schemes one by one. Also, it provides assurance that we can exhaustively list all the weaknesses. We uncover issues 17-19 and verify all the issues shown in Table IV and Table V using nuXmv and our formal models.

V. Attack Demonstration

This section provides detailed demonstrations of some of the attacks identified in Section III.

A. Birthday Attack

Background Theory. The Name Management (NM) message PGN 37632 assigns CA's NAME fields during network configuration. This message is 8 bytes long and contains the NM Control mode indicator field, sent in the least significant 4 bits

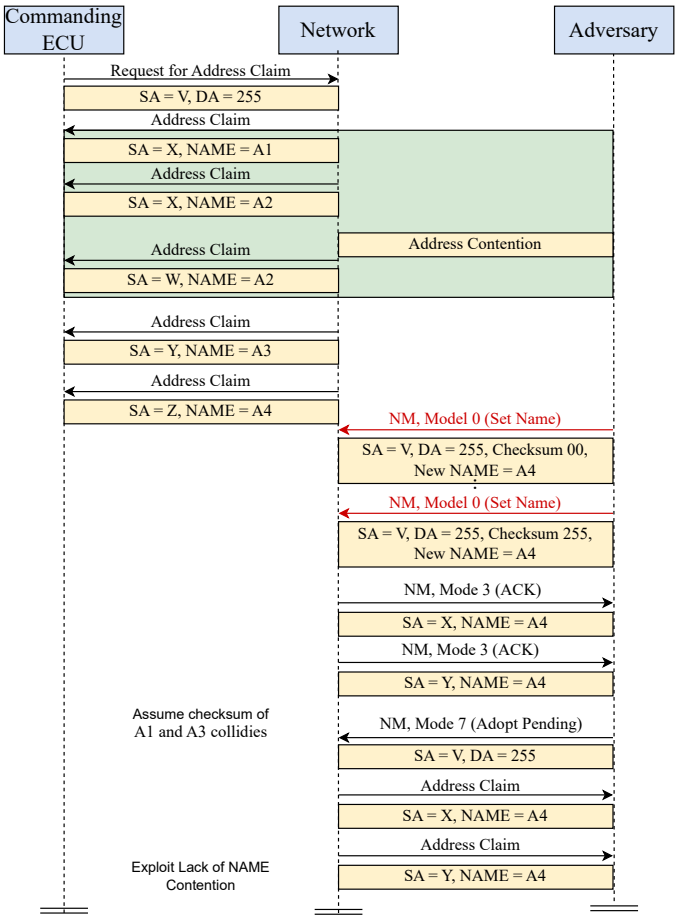


Fig. 4: Birthday Attack Demonstration on an AACCA Network

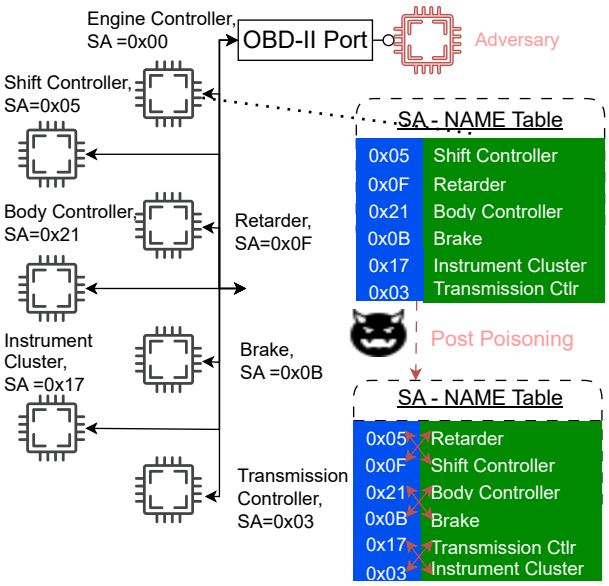


Fig. 5: SA-NAME Table Poisoning Attack

Fig. 6: Truck used for Validating SA-NAME Table Poisoning Attacks

of byte 2 that indicates various modes of usage. To ensure that the NM message has been received by the correct CA, an 8-bit NAME checksum is used.

Execution. We design a birthday attack, a brute-force collision attack that exploits the mathematics behind the birthday problem in probability theory [32].To demonstrate the attack, we impersonated the commanding ECU and exploited the lack of authentication handshake during NAME assignment and the higher likelihood of 8-bit checksum collisions to assign a legit CA with an invalid or existing NAME. The unique SA-NAME mapping on the network is now broken, as two CAs with two different source addresses have the same corrupted NAME. This scenario remains stealthy as SAE-J1939-81 does not handle the NAME contention. If the attacker assigns a NAME "A4", in effect, the original CAs A1 and A3 are disconnected from the network, as shown in Fig. 4. This brute-force birthday attack can change multiple CA's identities, leading to undesired vehicle behavior.

B. SA-NAME Table Poisoning Attack

Background Theory. Each CA records and/or compares the received address claims to their SA-NAME association table of used SAs and network addresses. Should multiple CAs claim to use the same address, an address contention is detected on the CAs using the contented address. The CA with the equal or lowest NAME wins, and the other CAs shall claim a different address or stop participation on the network. These changes on the network are also reflected in the SA-NAME association table maintained at each CA. To defend against the adversarial impersonation of legitimate CAs on the SAE-J1939 network, SAE J1939 included an imposter PG Alert message acronymed as "IPGA" which would be sent out when a CA receives a message from its SA.

Execution. Exploiting the lack of authentication on address claim messages, we impersonated several CAs to corrupt the SA-NAME association table maintained on each CA. We bypassed the imposter alert detection mechanism by using destination-specific or P2P claims to show that the attack can still be stealthily carried out. We developed Python scripts to attack the truck shown in Fig. 6 that consisted of an Engine Controller, Shift Controller, Retarder, Brake, Body Controller, Transmission Controller, and Instrument Cluster. Each CA maintains an SA-NAME association table. An Engine Controllers SA-NAME association table is shown in Fig. 5. To demonstrate the attack, we corrupted the engine controller's and headway controller's SA-NAME association table by connecting to the OBD port and sending destination-specific claims, as shown in Fig. 5. Note that we chose the NAME corruption in such a way that it swaps identities on a critical vehicular network. We confirmed the Engine SA-NAME association table corruption by inspecting the address where the SA-NAME table was stored in the Engine Controller. For a CA based on the SA-NAME association table, this corruption can cause undesired vehicle behavior due to identity swap. Messages received from the brake are now treated as from the shift controller, messages received from the body controller are now treated as received from the instrument cluster, and messages received from the transmission controller are now treated as received from the retarder. With the poisoning attack, we could disconnect the ABS on a moving truck, which led to hard/erratic braking. The vehicle warning on the dashboard is shown in Fig. 7a. We could disable the radar on an advanced truck equipped with radar while the truck was moving. Disabling of radar, in effect, disabled vehicle features such as lane change assistance, parking aid, collision mitigation, blind spot detection, and rear cross-traffic alert. The vehicle warning is shown in Fig. 7b. The actual braking torque of the retarder was indicated abnormal, as shown in Fig. 7. Since we used destination-specific messages directed to the Engine controller, our messages were undetected by the legit CAs, and hence, no imposter alert message was seen post-attack execution in the network traffic logs.

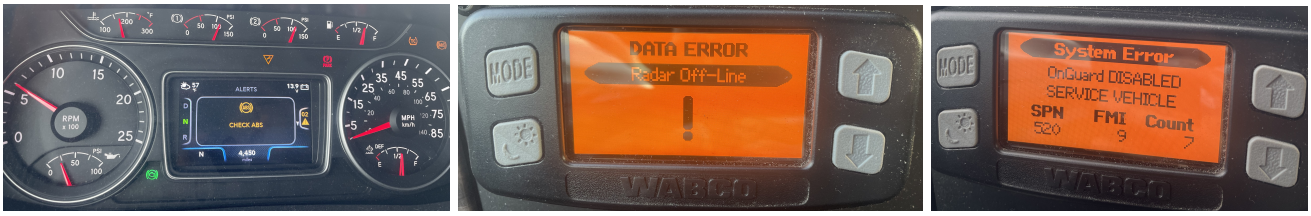
C. Collision Attacks

Background Theory. After transmitting a claim message, the transmitted CA shall monitor the network for error code information and contending claims. If an error has occurred or a CA detects an equal or lower claim, automatic retransmission of the claim message shall be scheduled after a pseudo-random transmit delay. If this pseudo-random transmit delay is predictable, an adversary can perform repeated intentional collisions, denying CAs participation in the SAE-J1939 network.

Execution. Our experimental test setup (shown in Fig. 8) consists of an adversary, an engine controller, and a PCAN logger connected to a J1939 CAN network at 250 kbps. We developed Python scripts to replay the engine claim at a time, t1, which took $T_{pystack}$ time to reach the lower layer driver and further took T_{frame} time to get transmitted on the CAN bus as shown in Fig. 9. Assuming the engine controller took $T_{enginestack}$ to reach the application layer, $T_{processing}$ time was taken to process the frame and a delay of $T_{randomdelay}$ was added to make the re-claiming time unpredictable to avoid a collision. On timeout of $T_{randomdelay}$, the engine controller reclaims. The engine controller received the frame at time t8, and reclaim was started at t7. The frame takes $T_{enginestack}$ to reach the lower layer driver and further takes T_{frame} to get transmitted on the bus and finally takes $T_{pystack}$ time for the adversary to receive the message in the application layer at timestamp t6. The PCAN logger receives the adversary's claim at timestamp t2 and receives engine controllers reclaim at timestamp t5. On visualizing these timestamps and delays as shown in Fig. 9, we could derive the below equations:

$$x = (t5 - t2) + T_{pcanstack} - (T_{frame} + T_{pcanstack})$$

$$x = (t5 - t2) - T_{frame}$$



(a) ABS Disabled on the Truck (b) Radar Disconnected (c) Braking Torque of the Retarder

Fig. 7: Real World Attack Impact

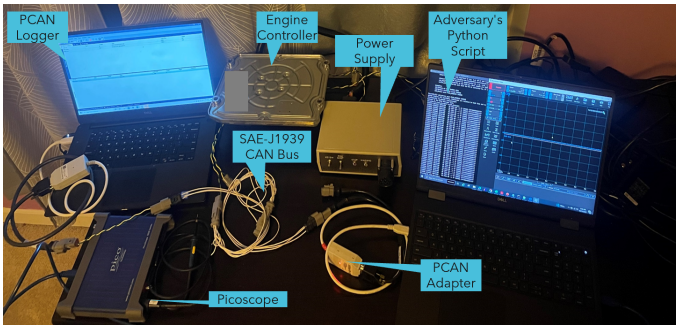


Fig. 8: Bench Setup for Validating Claim Collisions

For a baud rate of 250000 *kbps*, the bit time is $1/250000$ *seconds*, i.e., $4\mu s$. This means that 1 *bit* takes $4\mu s$ to transfer on a bus of 250000 *kbps*. Hence, the approximate time for transferring one frame consisting of $125bits$ is $500\mu s$. However, when measured with an oscilloscope, we found this value to be $552.2\mu s$

$$T_{framemeasured} = 552.2\mu s$$

$$x = (t5 - t2) - 0.5522ms$$

Since the adversary's Python script and PCAN logger are on two different devices, time synch is inaccurate. However, time differences on the device itself, such as $t6 - t1$ and $t5 - t2$, are reliable. Hence, we created equations to enhance accuracy based on time differences on one device. With time synchronization, we do not need to restrict in this manner. If the adversary tries to find x based on $pystack$ time, which is under the control of the adversary, the delay - x ms that needs to be accounted for the adversary to collide with a claim message perfectly is:

$$x = (t6 - t1) - (T_{frame} + T_{pystack}) - (T_{frame} + T_{pystack})$$

$$x = (t6 - t1) - 2 * (0.5522ms + T_{pystack})$$

With multiple logging on our experimental setup, we found that x averaged to 18.24376321 ms. We could perfectly collide with engine reclaims by adding an 18.24376321 ms delay in the adversarial Python script, as shown in Fig. 10.

VI. Discussion and Future Work

In this work, we take the first steps to perform a systematic analysis of SAE J1939 networks, starting with NMP. We uncover 19 new attack vectors, validate them using formal verification, and show their impact in real trucks. Though our attacks are extensively applicable in MHD trucks, they can be demonstrated wherever SAE J1939 standards are used, such as agricultural equipments, forklifts, and industrial equipments. As SAE J1939-81 is applied over ISO 11898 and ISO 11992 physical layers, our attacks would be relevant on both these physical layers. Although CAN FD Data Link Layer J1939-22 [10] offers security trailers for onboard messages, NMP messages were excluded. Therefore, our attacks are feasible on SAE J1939 CAN-FD networks and are not limited to classic CAN networks.

Our future work includes uncovering attacks in other SAE-J1939 protocols and deploying robust security mechanisms with real-world demonstration. We also plan to model and experiment with different inter-CA authentication handshake mechanisms and authenticated commanded messages to create an effective defense against our attacks. This will include exploring practical dynamic inter-CA key agreement schemes and identifying optimized security trailers.

VII. Acknowledgement

We thank Cummins Senior Security Advisor Chris S York and Datalink Architect Rob Frost for their review feedback. We would like to thank Internal Machine Validation Manager Nick Ouellette and Internal Machine Validation Technical Specialist Lesley Kelle for helping with truck facilitation.

References

[1] "2023 Global Automotive Cybersecurity Report," 2023. [Online]. Available: https://upstream.auto/reports/global-automotive-cybersecurity-report/
[2] "Regulation R155 E/ECE/TRANS/505/REV.3/ADD.154." [Online]. Available: https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security
[3] "Regulation R156 E/ECE/TRANS/505/REV.3/ADD.155." [Online]. Available: https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update
[4] SAE/ISO, "ISO/SAE 21434 - Road vehicles — Cybersecurity engineering," 2021.
[5] ——, "ISO/SAE 24089 - Road vehicles — Software update engineering," 2023.
[6] "Sae j1939 physical layer, 250 kbps, twisted shielded pair." [Online]. Available: https://www.sae.org/standards/content/j1939/11/
[7] "Sae j1939 physical layer, 500 kbps." [Online]. Available: https://www.sae.org/standards/content/j1939/14_202204/
[8] "SAE J1939 Physical Layer, 250 Kbps, Un-Shielded Twisted Pair (UTP)." [Online]. Available: https://www.sae.org/standards/content/j1939/11/
[9] "Sae j1939 data link layer." [Online]. Available: https://www.sae.org/standards/content/j1939/21_202205/
[10] "Sae j1939 can fd data link layer." [Online]. Available: https://www.sae.org/standards/content/j1939-22/
[11] "Sae j1939 network layer." [Online]. Available: https://www.sae.org/standards/content/j1939/31_201809/

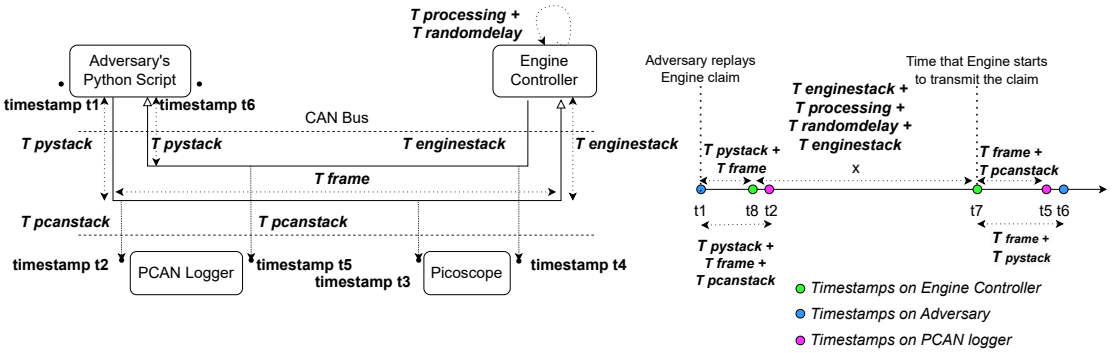


Fig. 9: Visualizing the Events in a Claim and its Response, to Derive Temporal Offset for Adversarial Collision

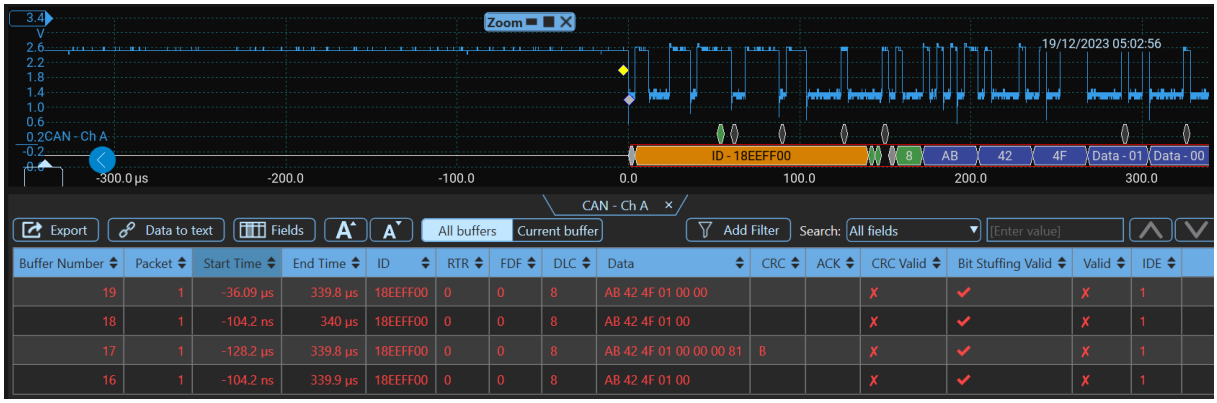


Fig. 10: Guaranteed Collision of Claims Indicated in CRC Valid Field of the CAN Frame, as seen on an Oscilloscope

[12] "Sae j1939 vehicle application layer." [Online]. Available: https://www.sae.org/standards/content/j1939/71_202002/

[13] "Sae j1939 application layer - diagnostics." [Online]. Available: https://www.sae.org/standards/content/j1939/73_202208/

[14] "Sae j1939 network management." [Online]. Available: https://www.sae.org/standards/content/j1939/81_201106/

[15] "Sae j1939 j1939 digital annex." [Online]. Available: https://www.sae.org/standards/content/j1939da_202208/

[16] Y. Burakova, B. Hass, L. Millar, and A. Weimerskirch, "Truck hacking: An experimental analysis of the SAE j1939 standard," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, Aug. 2016. [Online]. Available: https://www.usenix.org/conference/woot16/workshop-program/presentation/burakova

[17] T. R. Markham and A. Chernoguzov, "A balanced approach for securing the OBD-II port," *SAE Int. J. Passeng. Cars Electron. Electr. Syst.*, vol. 10, no. 2, pp. 390–399, Mar. 2017.

[18] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, ser. Lecture notes in computer science. Cham: Springer International Publishing, 2017, pp. 185–206.

[19] A. J. Michaels, M. Fletcher, C. Henshaw, V. S. S. Palukuru, and J. Moore, "Managing trust along the CAN bus," in *SAE Technical Paper Series*, no. 2022-01-0119. 400 Commonwealth Drive, Warrendale, PA, United States: SAE International, Mar. 2022.

[20] Q. Zou, W. K. Chan, K. C. Gui, Q. Chen, K. Scheibert, L. Heidt, and E. Seow, "The study of secure CAN communication for automotive applications," in *SAE Technical Paper Series*, no. 2017-01-1658. 400 Commonwealth Drive, Warrendale, PA, United States: SAE International, Mar. 2017.

[21] S. Mukherjee, H. Shirazi, I. Ray, J. Daily, and R. Gamble, "Practical dos attacks on embedded networks in commercial vehicles," vol. 10063, 12 2016, pp. 23–42.

[22] T. Bochot, P. Virelizier, H. Waeselynck, and V. Wiels, "Model checking flight control systems: The airbus experience," in *2009 31st International Conference on Software Engineering-Companion Volume*. IEEE, 2009, pp. 18–27.

[23] A. Biere, T. van Dijk, and K. Heljanko, "Hardware model checking competition 2017," in *2017 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2017, pp. 9–9.

[24] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino, "5greasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 669–684. [Online]. Available: https://doi.org/10.1145/3319535.3354263

[25] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. ieee, 1977, pp. 46–57.

[26] C. Miller and C. Valasek, "Adventures in automotive networks and control units." [Online]. Available: http://illmatics.com/car_hacking.pdf

[27] ——, "A survey of remote automotive attack surfaces." [Online]. Available: http://illmatics.com/remote%20attack%20surfaces.pdf

[28] S. Kumar, J. Daily, Q. Ahmed, and A. Arora, "Cybersecurity vulnerabilities for off-board commercial vehicle diagnostics," in *SAE Technical Paper Series*, no. 2023-01-0040. 400 Commonwealth Drive, Warrendale, PA, United States: SAE International, Apr. 2023.

[29] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L.-J. Hwang, "Symbolic model checking: 1020 states and beyond," *Information and computation*, vol. 98, no. 2, pp. 142–170, 1992.

[30] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[31] R. C. et al., "The nuxmv symbolic model checker," in *Computer Aided Verification: 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings 26*. Springer, 2014, pp. 334–342.

[32] J. Katz and Y. Lindell, *Introduction to modern cryptography*, 2nd ed., ser. Chapman & Hall/CRC Cryptography and Network Security Series. Boca Raton, FL: CRC Press, 2015.