CommPact: Evaluating the Feasibility of Autonomous Vehicle Contracts

Jeremy Erickson, Shibo Chen, Melisa Savich, Shengtuo Hu, Z. Morley Mao
University of Michigan, Ann Arbor
{jericks,chshibo,msavich,shengtuo,zmao}@umich.edu

Abstract—**In Autonomous Vehicle (AV) platooning, vehicles queue up with minimal following distances for improved traffic density and fuel economy. If one vehicle is compromised and suddenly brakes, these AVs will most likely be unable to prevent a collision. In this work, we propose a proactive approach to platooning security: *Autonomous Vehicle contracts*, in which AVs are architected to use secure enclaves to enforce agreed-upon driving rules, such as a restriction not to brake harder than a certain threshold while the contract is in effect. We explore whether AV contracts will be feasible in worst-case emergency situations while simultaneously under attack, when it is imperative to return full autonomy to AVs as soon as possible. Through our prototype contract implementation using Intel SGX enclaves, including measurement from real-world testing of wireless On-Board Units (OBUs), we show that AV contracts can be quickly and safely terminated in the event of an emergency while retaining a false positive rate of under 0.001% per 10 hours of use. We find that individual autonomy can be returned to the vehicles of an 8-vehicle platoon under contract within 1.5 seconds of an attack, including both detection and safe vehicle separation. Smaller platoons are even quicker. Consequently, automobile manufacturers may find the additional safety offered by AV contracts to provide a net benefit.**

I. Introduction

We are currently on the precipice of Autonomous Vehicles (AVs) becoming a reality for the general public. This technology promises improvements to safety, reliability, efficiency, and quality of life. Vehicle-to-Vehicle (V2V) and Vehicle-to-Roadside (V2R) communications, collectively called V2X, are one area of ongoing research in which vehicles communicate with entities around them to better plan their driving paths. V2X can be used to deliver critical safety and positional information to nearby vehicles, as well as help shape traffic patterns on a regional scale. One exciting promise of Connected AVs (CAVs) is Cooperative Adaptive Cruise Control (CACC), or *platooning*, in which several vehicles trail one after another with minimal following distance, behaving as a single unit.

Platooning brings many advantages, particularly for highway driving. Vehicles following closely behind one another can reduce their wind resistance, significantly increasing their fuel economy. This is particularly important for vehicles that are on the road for long periods of time with minimal speed changes, such as trucks, for which testing has shown a 5-10% decrease in fuel usage, including for the leading vehicle [1]. Additionally, by packing vehicles tighter together, traffic congestion can be reduced.

Just as humans are taught to drive safely, AVs are taught to identify and avoid dangerous situations, e.g., too little following distance with a preceding vehicle. Some AV models explicitly place safety constraints on the vehicle operation [2], within which the AV can freely navigate. For instance, an AV should retain a safe following distance between itself and any preceding vehicle so that it has time to react to any unexpected behavior. However, in the process of joining a platoon, an AV will find itself compelled to violate this safe following distance constraint, as it must maintain a close following distance for economic benefit. This puts the vehicle occupants at risk that the preceding vehicle could suddenly brake and cause a collision, potentially one involving many vehicles with limited ability to respond. Liu has demonstrated this issue [3] using the PLEXE platooning extension for the popular AV simulator, Veins. Researchers have demonstrated that existing vehicles can be exploited to remotely control the operation of the brakes, even over the network [4, 5, 6]. Given the history of malware in general, it would be foolish to expect future autonomous vehicles to be immune to compromise.

Many researchers have explored the idea of misbehavior detection for autonomous vehicles [7, 8], but these approaches are inherently reactive. A compromised vehicle must first exhibit unexpected behavior before other vehicles can detect an anomaly and respond to the situation. In a platoon, with many vehicles packed closely together, a malicious vehicle could potentially create a devastating pile-up by braking more quickly than following vehicles can react. Based on published third-party stopping distance test results, many modern passenger vehicles exhibit varying maximum braking decelerations ranging from approximately 8 m/s^2 to 11 m/s^2 on dry pavement [9, 10], although tire compound and environmental conditions can further increase this variability. Even with instantaneous detection and response, two vehicles traveling at 100 km/h (62 mph) with the preceding vehicle braking at 1 G and the trailing vehicle braking at 0.9 G will require 4.32 meters of following distance to avoid a collision. This necessary separation increases linearly by another car length (5 meters) for every 173 ms of delayed response by the trailing vehicle. Although reactive measures may be able to reduce the damage and risk of injury from such a collision, they are clearly insufficient to prevent contact at desirable following distances of 1-3 meters.

Instead, we propose a proactive solution: autonomous vehicle *contracts*. A contract is an agreement between autonomous vehicles not to violate certain driving parameters. For instance, two or more vehicles traveling close together in the same lane may form a contract ensuring both maintain a speed of 100 km/h with an allowed deviation of 2 km/h until the contract is either amended, ended, or safely terminated upon communication failure. Contracts are enforced by an *enclave*, a verifiable binary running on each vehicle that can be remotely attested through cryptographic hardware keys. The AV is architected such that driving commands must be signed by the enclave, which ensures the vehicle will not violate the agreed-upon parameters. Each vehicle's powertrain and braking Electronic Control Units (ECUs), the computers that control actuation of the gas and brakes, are restricted from taking actions not validated and signed by the enclave. By remotely attesting that platoon vehicles conform to this model and signing the platoon's contract before joining, each vehicle gains additional safety assurance that the other vehicles in the platoon, even compromised ones, are restricted from performing actions disallowed by the contract, such as suddenly braking.

Of course, contracts also have a significant downside: that vehicles restricted from making sudden velocity adjustments may consequently be restricted from responding to emergency situations, such as an obstacle appearing in the road. We cannot allow vehicles to unilaterally void a contract, as doing so would obviate this new protection against bad actors. To make matters worse, the wireless communication channels can be jammed by an adversary, even one not part of the platoon. In these worst-case scenarios, it may not be possible for the platoon vehicles to actively coordinate an emergency response. Therefore, we need a proscribed mechanism for voiding a contract safely in the presence of interference and need to minimize delay before returning autonomy to each vehicle. Human-driven vehicles are subject to the Perception Response Time (PRT) which indicates that it generally takes humans between 1 and 3 seconds to respond to unexpected circumstances [11]. If we can safely separate and return autonomy to platooning vehicles in a similar time, it may be desirable to use contracts to increase the safety factor for trailing vehicles when platooning. Our experimental results show that it should be possible to do so in under 1.5 seconds for platoon sizes up to 8.

In this work, we explore the problem space of returning autonomy to vehicles under contract as quickly as possible and how the various trade-offs affect the feasibility of introducing contracts to autonomous vehicle platooning. We present the following contributions:

- A prototype framework and messaging protocol that supports the creation and maintenance of contracts as vehicles join and leave the platoon and maintains platoon safety in the presence of adversaries.
- An analysis of the physical process of separating platooning vehicles to safe following distances and velocities with minimal delay.
- An analysis of the trade-off space for minimizing delay in detecting communications failure while maintaining connectivity in unreliable conditions.
- A simulated implementation that demonstrates contracts in action.[1]
- Measurement of the wireless latencies and cryptographic computational delays that influence the critical path for detecting communications failure.

II. Threat Model

In this work, we assume the perspective of a vehicle occupant of an autonomous vehicle forming a platoon with other autonomous vehicles. In our model, all platooning vehicles are fully autonomous. The most important assumption is that an occupant *must* trust their own vehicle. A malicious vehicle can harm its occupants, e.g., drive itself off a cliff, entirely outside of the scope of platooning or contracts. It follows then that benign vehicles will act in their occupants' best interest.

We assume that any or all *other* vehicles in the platoon can be compromised or malicious, and that the operating system and applications can be controlled arbitrarily by an adversary. Malicious vehicles can collude, may maneuver in any way within their control, and send arbitrary network traffic, including completely jamming the communication channel. We assume that in the absence of an attack, platooning vehicles can coordinate to navigate around obstacles or slow down as the situation merits, and so our work focuses on the worst-case scenario in which all communications are jammed or another Denial of Service (DOS) attack is present. We assume benign vehicles will react to anomalous behavior by separating, avoiding, and blacklisting any non-compliant vehicle, generally outside the scope of this work. In comparison, this work is primarily focused on proactively preventing sudden, stealthy attacks that other approaches cannot address. Our protocol is designed so that any attack on either the integrity or the availability of the systems and messages will devolve into a DOS, which we have designed the protocol explicitly to accommodate.

We assume each vehicle is equipped with an enclave that provides attestation capabilities. If vehicles are unable to attest one another's enclaves, or if conditions are adverse, they will not form a platoon. Our architecture assumes that the powertrain and braking ECUs for each vehicle, during manufacturing, have been paired with the enclave and exchanged keys. This allows the enclave and ECUs to validate messages from each other, and prevents another device from impersonating them, regardless of communication medium. We assume that the powertrain and braking ECUs are the final arbiters of the car's movement. That is, physical attacks between an ECU and its actuators or that otherwise modify the movement of the car are outside the scope of our model. Our model is built on the assumption that the enclave and ECUs will behave as designed. As such, we assume that reducing the ECUs' attack surface to only accept commands from the enclave and remotely attesting the integrity of the enclave binary is sufficient to protect them from adversarial control.

[1] https://github.com/jericks-umich/commpact

III. Related Work

To date, the research on autonomous vehicle platooning has primarily focused on efficient communications, support for platooning maneuvers, and the string stability problem [12, 13]. PLEXE has paved the way for researchers to evaluate different platooning strategies to meet these goals [14, 15]. More recently, security researchers have begun to evaluate the safety implications of compromised autonomous vehicles engaging in platooning scenarios. Heijden [16] uses PLEXE to demonstrate how jamming and data injection attacks against several CACC controllers can lead to vehicle collisions. Petrillo [17] proposes a new CACC controller that is resilient to adversarial attacks and uses PLEXE to evaluate it against spoofing, message falsification, and packet loss. Anomaly and misbehavior detection techniques have also been explored [7, 8]. However, as Liu [3] shows, a malicious vehicle with control over the vehicle's motion can cause a platoon collision directly. Even an instantaneous braking response cannot prevent collisions among heterogeneous vehicles.

Enclaves have typically been used in cloud and mobile applications. In 2015, with growing concerns of data compromise on public clouds [18], Intel introduced SGX [19], with stronger cryptographic root-of-trust guarantees than previous offerings. SGX was groundbreaking because of its remote attestation capability. After forming a proof of attestation, it could be sent to a remote user and verified, ensuring that the application was not altered and was running on genuine Intel hardware [20]. Applications could be designed to only deploy keys or provision sensitive data after this attestation process was complete. Despite many attacks against the current generation of enclaves [21, 22, 23], the concept is powerful and can enable many secure features once more refined and resilient iterations are available.

Several recent works have explored the use of secure enclaves for automotive applications. NXP has explored the use of enclaves to encrypt and protect sensitive data, similar to the use of a Hardware Security Module [24]. Virtual Open Systems [25] and Kim [26] have explored the use of TrustZone's secure world as a platform for segregating critical vehicle software and In-Vehicle Infotainment software while running both on the same hardware. However, fundamentally, these works follow the same model as a hypervisor restricting sandboxed software from tampering with privileged data while sharing the compute platform. The true promise of enclaves is the ability to run trusted code on *remote* hardware and receive attestation that it will faithfully execute.

To the best of our knowledge, our proposal is the first to leverage the remote attestation capabilities of enclaves to protect vehicle occupants by enforcing driving constraints on *other* autonomous vehicles.

IV. Background on Enclaves

The technical background of enclaves, upon which AV contract enforcement is built, is largely outside the scope of this paper. In this section, we aim to highlight the important features that enclaves provide, without delving too far into the mechanisms by which they provide these features. Curious readers may refer to the literature on this topic [19, 27, 28] for more information.

Enclaves can be thought of as a hardware-enforced partitioned environment for executing *trusted* code. Untrusted code, which likely includes the main operating system and most applications, runs in the *insecure world*, while trusted applications run in the *secure world*. Memory used by the secure and insecure worlds is disjoint, and execution may only change worlds using prescribed instructions that perform a secure context switch, similar to a syscall or hypercall.

Intel, AMD, and ARM each have their own enclave architectures, Software Guard Extensions (SGX), Secure Encrypted Virtualization (SEV), and TrustZone, respectively. In our prototype, we use Intel SGX, as its root of trust characteristics are more mature than those of TrustZone and hardware for SGX v1 is available for consumer purchase. Regardless, our model does not rely on the characteristics of a particular enclave technology, but rather on these key features:

- Remote cryptographic attestation of valid hardware and and an immutable, currently-executing enclave binary
- Ability to securely generate and exchange keys derived from a hardware-based random number generator

In this work we require an enclave to run on each vehicle. Since the binary for this enclave can be attested, one vehicle can trust that the enclave running on another vehicle will faithfully execute as expected. Our model expects that all vehicles will be using identical, or at least mutually trusted and compatible, enclave applications, which can be verified as such before a vehicle joins the platoon. Barring vulnerabilities in the enclave binary itself, enclaves can trust other vehicles' enclaves to behave as they would themselves.

The enclave will run a small binary with a simple purpose: to validate the commands being passed to the powertrain and braking ECUs and filter any that conflict with the current contract. The ECUs must be configured to only accept commands that are signed by the enclave which ensures that the commands conform to the agreed-upon contract. The ECUs cannot simply conform to the contract policy themselves without an enclave because without the enclave's ability to be remotely attested, other vehicles cannot trust that the code properly enforces the contract. When joining a platoon, each vehicle attests the enclave running on the other vehicles before exchanging keys. Later, when receiving a contract signed by an attested enclave of another vehicle, one can trust that the remote enclave will enforce the terms of the contract.

V. Methodology

Since our primary goal is safety, we must consider what happens in emergency situations. Each vehicle will be bound to the platoon contract and restricted in how it may react. This currently takes the form of speed and acceleration bounds. In the common case, with a working communication channel and no adversarial interference, the platoon will be able to coordinate to adjust the contract and react directly to an emergency situation. Enumerating the many optimal solutions to specific emergency situations in a *cooperative* environment is outside the scope of our work. We focus instead on the *adversarial* environment, in which communications can be severed. In this environment, we wish to safely terminate the contract as soon as possible and return full autonomy to each vehicle, allowing them to respond to the emergency situation as individuals. We cannot simply terminate the contract while the platoon is still formed, as this would effectively negate the safety restrictions the contract places upon other vehicles. Consequently, we must separate the vehicles in the platoon to a Safe Separation Distance and Velocity (SSDV) before terminating the contract. This becomes our primary form of defense against adversarial actions. The platoon contract may be cooperatively terminated by any member vehicle, or by a timeout due to lack of communication, but must always remain in effect until the platoon has safely separated and the Emergency Termination Procedure (ETP) is complete.

Let us imagine a worst-case scenario. At some time t, an emergency occurs. We cannot predict this emergency, and so cannot preload instructions for the platoon vehicles to follow to react to this specific emergency. We also cannot assume that all vehicles in the platoon are even aware that an emergency has occurred. Perhaps the leader has just discovered an obstacle in the road but it is not visible to the following vehicles. Also, at time t, an adversary jams the communication channel.

In this situation, we have two phases before each vehicle can be released from the contract and may regain full autonomy. The first phase is the **Recovery Phase**, in which the platoon attempts to recover from its communications failure. Temporary wireless communications failures are common, and so we must find a balance between robust communications recovery and minimizing the delay before initiating the second phase of the termination procedure. The second phase is the **Separation Phase**, in which the platoon vehicles begin to separate from one another. The Separation Phase ends when the vehicles have reached the SSDV, at which point the contract ends. The sum of these two phases will dictate how long it takes for the platoon vehicles to react to a worst-case emergency situation.

We start by explaining the Separation Phase and ETP, as the separation of the platoon informs the design of the communications protocol for the Recovery Phase.

A. Separation Phase

In the event that the Separation Phase is triggered, the platoon vehicles must separate and return to a safe following distance from one another. This procedure must be defined in advance, as termination may commence due to a total disruption of communications and so no orchestration between vehicles can be relied upon during the procedure itself. We cannot assume that the vehicles can separate via lane change as no lane may be available. Separation must occur only from the rear of the platoon since we cannot assume the leader has room to speed up. We wish to be able to pre-calculate the amount of time this separation will take so our total delay can stay within a safe threshold every time platoon members are added or environmental conditions change.

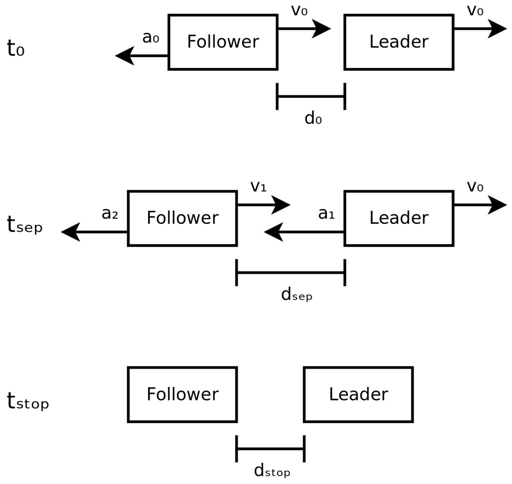

Fig. 1. Free-body diagrams showing distances, velocities, and accelerations of the Leader and Follower vehicles at the beginning of the separation procedure (t_0), the end of the separation procedure (t_{sep}), and the point at which both vehicles have come to rest (t_{stop}).

Platoon Size	v_0 m/s	a_0 m/s^2	a_1 m/s^2	a_2 m/s^2	d_0 m	d_{stop} m	t_{sep} ms
2	27.77	-8.82	-9.81	-8.82	1.0	1.0	158
3	27.77	-4.41	-9.81	-8.82	1.0	1.0	307
4	27.77	-2.94	-9.81	-8.82	1.0	1.0	451
5	27.77	-2.20	-9.81	-8.82	1.0	1.0	594
6	27.77	-1.76	-9.81	-8.82	1.0	1.0	728
7	27.77	-1.47	-9.81	-8.82	1.0	1.0	867
8	27.77	-1.26	-9.81	-8.82	1.0	1.0	982

TABLE I
SEPARATION PROCEDURE DELAYS (t_{sep}) FOR SEPARATION DECELERATIONS (a_0) CORRESPONDING TO THE RELATIVE MAXIMUM BRAKING RATE FOR PLATOON SIZES OF 2 THROUGH 8.

Each vehicle will decelerate at a proscribed fraction of the platoon's maximum deceleration rate so as to maintain equal distance between vehicles. The absolute negative acceleration A for each vehicle in the platoon is given by:

$$A_n = \frac{n}{N} M \qquad (1)$$

where N is the number of follower vehicles in the platoon, n is the vehicle's index in the platoon starting at 0 for the leader and ending at N for the tail vehicle, and M is the minimum of the platoon vehicles' maximum braking decelerations.

1) Calculating Separation Phase Delay: We first define the Safe Separation Distance and Velocity (SSDV) between the leader l and follower f as the distances and velocities such that if both apply their maximum braking acceleration they will not collide before coming to a complete stop. To simplify the scenario for visualization, we focus on a platoon made up of two vehicles, a leader and a follower, shown in Figure 1. At time t_0, both vehicles will be traveling at the platoon velocity v_0 and separated by a distance d_0, the Separation Phase will begin and the follower will decelerate at a_0.[2] At time t_{sep}, the vehicles will reach the SSDV. The leader will still have

[2] Other pairs of vehicles in a larger platoon will also separate at a_0 *relative to one another*, but will have a slower effective v_0 at t_{sep}

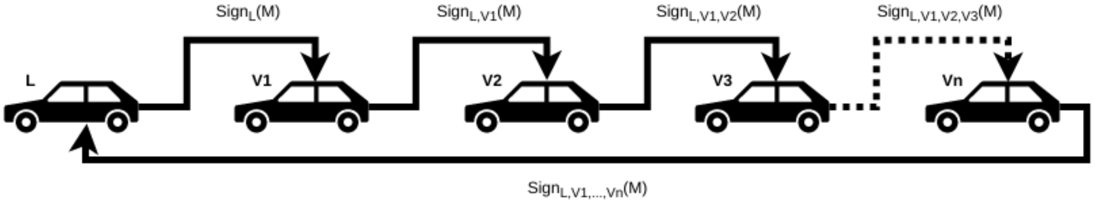

Fig. 2. A basic contract chain in which M is the contract message. The Recovery Phase timeout is extended, starting with the leader and progressing down the platoon in order. If the Emergency Termination Procedure is ever triggered, the leading vehicles are guaranteed to end the Recovery Phase no earlier than the trailing vehicles since each prior vehicle must have extended its own Recovery Phase timeout before continuing the chain.

velocity v_0 but the follower will have decelerated to v_1 and they will be separated by a greater distance d_{sep}. Both vehicles may begin to brake at their maximum rates. At time t_{stop}, the vehicles will have reached zero velocity and should remain separated by some safe distance d_{stop}.

We can solve for t_{sep} with the following equation, derived from Figure 1:

$$\begin{aligned}(a_0^2 a_1 - a_0 a_1 a_2)t_{sep}^2 + (2a_0 a_1 v_0)t_{sep} + \\ v_0^2(a_1 - a_2) + 2a_1 a_2(d_0 - d_{stop}) = 0\end{aligned} \tag{2}$$

Since at t_0 we either know or can estimate all terms besides t_{sep}, we can solve Equation 2 with the quadratic formula. Table I shows the result of using Equation 2 for heterogeneous platoons matching our opening example.

B. Recovery Phase

Platooning vehicles will spend the majority of their time in a steady state with a contract active. During this time, each vehicle will participate in continuous communication to ensure the communications channel is still up. In the worst case, an unrecoverable communication failure must lead to the emergency termination of the contract and platoon.

In this work, we use Dedicated Short Range Communications (DSRC) [29] for its low latency and availability of hardware for testing, but any wireless technology with sufficiently-low message latency could be used. We assume the wireless spectrum is adversarially-controlled and therefore each vehicle will sign its messages. Since the keys are stored within our trusted enclave, we assume forging valid messages to be infeasible within our protocol's recovery period. We pick ECDSA signatures over the NIST P-256 (secp256r1) elliptic curve as it is both one of two curves supported for message authentication in DSRC [30] and also supported by the SGX enclave cryptographic library.

So that Equation 2 holds true, we use a semi-synchronized timeout to transition from the Recovery Phase to the Separation Phase. We can take advantage of the fact that platoon vehicles are ordered to ensure that trailing vehicles always start the Separation Phase *no later than* the vehicle preceding them. We use a **contract chain** to periodically extend the timeout and ensure that preceding vehicles *must* extend their Recovery Phase before trailing vehicles are able to do so. If at any point, the contract chain fails, vehicles preceding the failure will have an extended Recovery Phase timeout whereas trailing vehicles will continue to end the Recovery Phase at the previously-agreed time. The contract chain is shown in Figure 2.

The platoon leader will continuously generate new contract chains to replace the current contract and extend the timeout, subject to the enclave-enforced *contract principles* below. Followers who object to the terms of a new contract are free to not sign it and remain under the terms of an old contract, until it eventually expires.

1) Recovery from communications failure: Since the wireless medium is known to be unreliable, we wish to be able to tolerate some number of missed contract chains before concluding that the communications channel has been severed. The precise number of failed chains before inducing the ETP is variable and dependent on the network congestion and a safety delay threshold before which the emergency termination procedure should always be triggered.

Ultimately, the number of failed contract chains to attempt before initiating the Separation Phase is dependent on the total latency for one contract chain. A lower latency means more chains can be attempted before reaching the timeout, increasing reliability. Increasing the number of vehicles in the platoon will increase the latency of the contract chain, reducing the number of chains that can be attempted before the timeout. If we wish to keep the total delay for both the Recovery and Separation Phases lower than some bound, the platoon leader will need to carefully monitor the packet loss rate and pick an appropriate number of chains to attempt before the timeout to keep the chance of a false positive below some bound. The leader can also split the platoon to keep the contract chain latency manageable.

2) Contract Principles: Contract chains need not always traverse the platoon in the same order. For instance, the Join, Leave, and Split procedures[3] allow contract chains to originate from vehicles besides the leader, traverse the platoon in reverse order, or other behaviors. Because the nuances of the communications protocol can be complex during these scenarios, we identify several principles which must always be guaranteed by the contract to maintain safety.

Principle 1. A vehicle may extend its Recovery Phase timeout to the minimum value of all vehicles in front of it.

Principle 2. A vehicle may increase its speed bounds only once all vehicles leading it have done so.

[3] These procedures are described in the extended version of this paper.

Principle 3. A vehicle may decrease its speed bounds only once all vehicles trailing it have done so.

VI. Evaluation

Manufacturers who wish to implement platooning contracts will need to decide on an appropriate maximum delay from the time an attack occurs until AVs regain autonomy. This delay may be context-dependent, e.g. a 1.5 second delay may be tolerable on an interstate highway with barriers and traditional onramps and offramps, but a rural highway with perpendicular intersections may necessitate a maximum delay of only 0.75 seconds. This delay is the sum of the Recovery Phase and the Separation Phase, and while the Separation Phase is primarily adjusted through platoon length, the Recovery Phase is dependent on several different variables. While the Separation Phase delay can be computed using Equation 2, the Recovery Phase delay is highly dependent on contract chain latency. We have chosen an arbitrary, but conservative, reliability goal of 0.001% or lower chance to invoke the ETP due to environmental packet loss per 10 hours of platooning time. We believe that if this overall delay is similar or less than the human Perception Response Time (1-3 seconds), manufacturers may find it advantageous to augment vehicle safety with contracts while platooning.

To evaluate the Recovery Phase delay, we have implemented a prototype of platooning contracts using the PLEXE platooning extension to the Veins simulator. Our simulation runs on a Supermicro server with SGX-enabled X11SSZ-QF motherboard and an Intel Core I7-6700K at 4.0 GHz. Our ECUs are emulated using a Raspberry Pi 3B+ clocked at 1.4 GHz and connected to our server via Fast Ethernet (100 Mbps).

To evaluate the additional processing latency imposed by an enclave, we have extended PLEXE to:

- Instantiate SGX enclaves and connect to an emulated powertrain/braking ECU for each simulated vehicle
- Generate and exchange enclave and ECU keys
- Sign contracts and update enclave and ECU parameters
- Validate signatures and contract parameters within each enclave
- Transmit contract chains between vehicles over the simulated DSRC communication channel

Additionally, we have augmented the simulation to incorporate the wireless transmission delays we measured using two Cohda Mk5 OBUs. The critical path for contract chain completion time is shown in Figure 3, and the results of our simulation over runs with platoons of sizes 2 through 8 are shown in Figure 4.

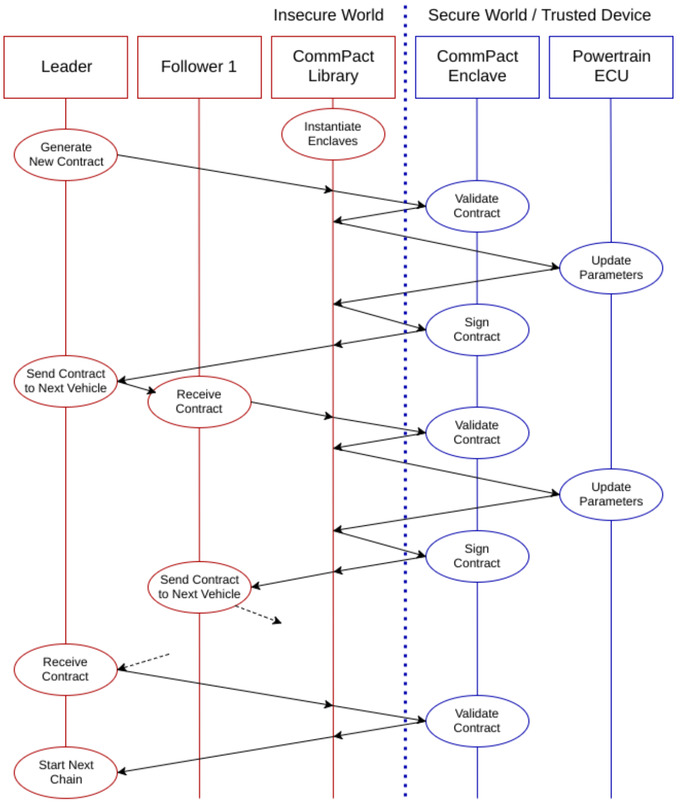

Fig. 3. Contract chain critical path from when the leader generates a new contract to when the leader verifies that the contract has been signed by all vehicles. This takes approximately 50 milliseconds for an 8-vehicle platoon.

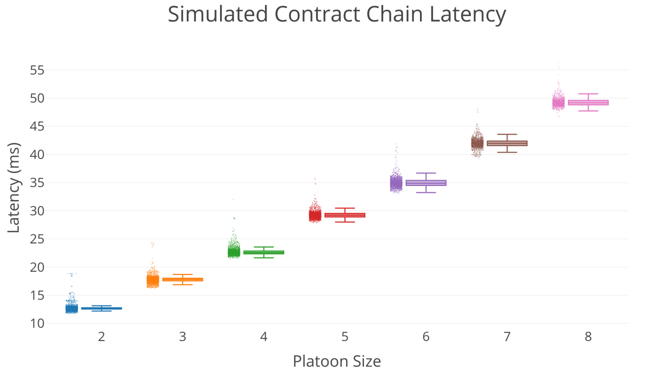

Fig. 4. Contract chain latency results for platoon sizes of 2 through 8. Mean latencies of 12.70, 17.80, 22.68, 29.26, 34.98, 42.00, and 49.27 ms, respectively.

A. DSRC Wireless Latency

Although recent works [31, 32] have shown average delays for DSRC messages to be between 1 ms and 3 ms per transmission depending on data rates and conditions, these studies measure DSRC messaging between semi-trailer trucks and over long (100m-500m) distances, which may be worse than seen in a typical passenger-vehicle platoon. To evaluate typical latencies seen in a passenger vehicle environment, we performed our own measurement study on DSRC wireless transmissions using two Cohda Mk5 OBUs with antennas mounted atop a 2013 Toyota RAV4 and 2002 Honda Civic. These vehicles were positioned at one-car-length intervals between 1 and 7 car lengths apart to evaluate the impact that distance has on packet loss and transmission latency. Our contract payload consists of one 64-byte message and between 1 and 8 64-byte ECDSA signatures (128-576 bytes). In total, we performed 13 trials of 1000 transmissions each. We tested

Operation	i7-6700K	RPi 3B+	ASIC
Sign	0.193 ms	0.709 ms	0.325 ms [33]
Verify	0.321 ms	1.321 ms	0.212* ms [34]

TABLE II

Average ECDSA Sign and Verify latencies on our experimental hardware and on reference ASICs from the literature. *We note that of the options presented by Knezevic [34] we show here the slowest configuration in their typical use category with the shortest critical path. Their fastest ASIC computes a verification in 0.037 ms on average.

each of the data rates supported by DSRC and found 18 Mbps to be most optimal.

The first 7 trials were performed one car length apart (1 meter gap) with payloads ranging from 128 bytes to 512 bytes to reflect the additional signatures appended to the contract chain as it traverses the platoon, each time traveling rearward by one car length. The remaining 6 trials were with the cars spaced 2 to 7 car lengths apart and used packet sizes ranging from 256 to 576 bytes, to reflect the messages sent from the tail vehicle to the leader. We saw mean latencies ranging from 0.949 ms for 128-byte packets at 1 car-length to 1.200 ms for 576-byte packets at 7 car-lengths, with 98th percentile latencies from 1.258 ms to 1.485 ms.

In all of the 13 trials (13,000 transmissions), we only had a single packet lost (latency greater than 10ms). Our sample size is not large enough to report a packet loss rate with a high degree of confidence, and the results may be different in alternate conditions, so we do not attempt to draw any conclusions about packet loss rate from these tests. Consequently, we use a conservative 1% packet loss rate in future calculations of false positives per 10-hour period.

B. Compute Latency

Of the total contract chain latency, the majority comes from computational delay. Of this, ECDSA sign and verify operations dominate. For each contract chain, the vehicle enclaves will each sign the contract and verify between one and N (the size of the platoon) signatures, as shown in Figure 2. Each vehicle's ECU will verify the message sent to it by the enclave and sign a response to the enclave. For an 8-vehicle platoon, there are 8 sign and 36 verify operations performed by enclaves on our reference server, and 8 sign and 8 verify operations performed by ECUs on our Raspberry Pi 3B+. Table II shows the latencies for these operations on our hardware, as well as reference latencies for these operations on low-powered ASICs presented by the literature. [33, 34]

With the exception of ECDSA sign operations on our server, all other sign and verify operations in our simulation are slower than the current state of the art in automotive ASICs for ECDSA operations. If we were to substitute these reference architecture latencies for our simulation times, we would see an improvement from 49.27 ms to 34.46 ms on average per contract chain for an 8-vehicle platoon.

C. AV Contract Configuration

Based on the results presented in Figure 4, we can determine appropriate values for the Recovery Phase timeout at different platoon sizes. We wish to reduce false positives (triggering of the ETP due to packet loss) to a manageable level while still minimizing the total Recovery Phase duration. We target a false positive rate of under 0.001% per 10 hours of platooning in an 8-vehicle platoon. Given a contract chain duration of approximately 50ms on average, we can expect to attempt 720,000 contract chains in this period. Using an estimated packet loss rate of 1%, the probability that we will see a false positive in this period is 0.00706% if we set our ETP timeout at 450 ms (9 contract chains), and 0.00055% at an ETP timeout of 500 ms (10 contract chains). Thus, we should choose a Recovery Phase of approximately 500 ms so as to keep the false positive rate below our threshold.

Platoon Size	# Chains	FP Rate per 10 hours	Recovery Phase	Separation Phase	Total Delay
2	7	0.00034%	89 ms	158 ms	247 ms
3	8	0.00012%	142 ms	307 ms	449 ms
4	8	0.00089%	181 ms	451 ms	632 ms
5	9	0.00019%	263 ms	594 ms	857 ms
6	9	0.00078%	315 ms	728 ms	1043 ms
7	10	0.00017%	420 ms	867 ms	1287 ms
8	10	0.00051%	493 ms	982 ms	1475 ms

TABLE III

Total expected delay to return autonomy to platooning vehicles at various platoon sizes, assuming a conservative 1% packet loss rate.

The total delay before autonomy can be returned to the platoon for different platoon sizes is shown in Table III. We note that these probabilities can be recalculated continuously during platooning operation and the number of contract chains attempted and the platoon size can be adjusted to compensate for periods of even higher packet loss.

VII. Conclusion

In platooning, Autonomous Vehicle contracts are likely to be beneficial, significantly increasing the difficulty for adversaries to remotely cause vehicle collisions in platoons where vehicle occupants are most at risk. Our prototype shows contracts can be adjusted to fit changing conditions and maintain a conservative emergency response delay within 1.5 seconds for platoons of 8 vehicles, or as short as 0.25 seconds for 2-vehicle platoons. During an attack, vehicles under contract can physically separate and regain full individual autonomy more quickly than many human drivers can even begin to react. While AV contracts may not be suitable for all environments or situations, their use in normal operating conditions may someday save lives and merits further investigation.

References

[1] Michael P. Lammert et al. "Effect of Platooning on Fuel Consumption of Class 8 Vehicles Over a Range of Speeds, Following Distances, and Mass". In: *SAE 2014 Commercial Vehicle Engineering Congress*. 2014.

[2] S. J. Anderson et al. "The intelligent copilot: A constraint-based approach to shared-adaptive control of ground vehicles". In: *IEEE Intelligent Transportation Systems Magazine* (2013).

[3] Jiafa Liu et al. "Secure and Safe Automated Vehicle Platooning". In: *IEEE Reliability Society* (2016): *Reliability Special Issue - 2016 August*.

[4] K. Koscher et al. "Experimental Security Analysis of a Modern Automobile". In: *IEEE Symposium on Security and Privacy*. 2010.

[5] Stephen Checkoway et al. "Comprehensive Experimental Analyses of Automotive Attack Surfaces". In: *20th USENIX Conference on Security*. 2011.

[6] Charlie Miller and Chris Valasek. *Remote Exploitation of an Unaltered Passenger Vehicle*. 2015. URL: https://goo.gl/o2rTXh.

[7] Kyong-Tak Cho and Kang G. Shin. "Viden: Attacker Identification on In-Vehicle Networks". In: *2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017.

[8] Norbert Bimeyer. "Misbehavior Detection and Attacker Identification in Vehicular Ad hoc Networks". PhD thesis. Technischen Universitt Darmstadt, 2014.

[9] Edmunds.com. https://goo.gl/JM1XZV.

[10] Mark Williams. https://goo.gl/Fgghyg.

[11] Jim Keenan. *Perception Response Time*. Ed. by Keith Borer Consultants. 2014. URL: https://goo.gl/1VBbyG.

[12] Michele Segata, Bastian Blessl, and Stefan Joerer. "Supporting platooning maneuvers through IVC: An initial protocol analysis for the JOIN maneuver". In: *Wireless On-demand Network Systems and Services*. 2014.

[13] Pedro Fernandes and Urbano Nunes. "Platooning With IVC-Enabled Autonomous Vehicles: Strategies to Mitigate Communication Delays, Improve Safety and Traffic Flow". In: *IEEE Transactions on Intelligent Transportation Systems*. 2012.

[14] Alessandro Salvi, Stefania Santini, and Antonio Valente. "Design, analysis and performance evaluation of a third order distributed protocol for platooning in the presence of time-varying delays and switching topologies". In: *Transportation Research Part C: Emerging Technologies*. 2017.

[15] D. Jia, K. Lu, and J. Wang. "A Disturbance-Adaptive Design for VANET-Enabled Vehicle Platoon". In: *IEEE Transactions on Vehicular Technology* (2014).

[16] R. van der Heijden, T. Lukaseder, and F. Kargl. "Analyzing attacks on cooperative adaptive cruise control (CACC)". In: *2017 IEEE Vehicular Networking Conference (VNC)*. 2017.

[17] Alberto Petrillo, Antonio Pescap, and Stefania Santini. "A Collaborative Approach for Improving the Security of Vehicular Scenarios: the case of Platooning". In: *Computer Communications*. 2018.

[18] Mehmet Sinan Inci et al. "Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud". 2015.

[19] Frank McKeen et al. "Innovative Instructions and Software Model for Isolated Execution". In: *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*. ACM, 2013.

[20] Andrew Baumann, Marcus Peinado, and Galen Hunt. "Shielding Applications from an Untrusted Cloud with Haven". In: *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, 2014.

[21] Jae Hyuk Lee et al. "Hacking in Darkness: Return-oriented Programming against Secure Enclaves". In: *USENIX Security Symposium*. 2017.

[22] Ferdinand Brasser et al. "Software Grand Exposure: SGX Cache Attacks Are Practical". In: *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. USENIX Association, 2017.

[23] Johannes Götzfried et al. "Cache Attacks on Intel SGX". In: *Proceedings of the 10th European Workshop on Systems Security*. EuroSec'17. ACM, 2017.

[24] Richard Soja. *Automotive Security: From Standards to Implementation*. Tech. rep. 2014.

[25] Michele Paolino. "ARM TrustZone and KVM Coexistence with RTOS For Automotive". In: 2015.

[26] S. W. Kim et al. "Secure device access for automotive software". In: *2013 International Conference on Connected Vehicles and Expo (ICCVE)*. 2013.

[27] ARM. *ARM Security Technology*. Tech. rep. 2009.

[28] David Kaplan, Jeremy Powell, and Tom Woller. *AMD Memory Encryption*. Tech. rep. 2016.

[29] J. B. Kenney. "Dedicated Short-Range Communications (DSRC) Standards in the United States". In: *Proceedings of the IEEE* (2011).

[30] "IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages". In: *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)* (2016).

[31] Song Gao, Alvin Lim, and David Bevly. "An empirical study of DSRC V2V performance in truck platooning scenarios". In: Digital Communications and Networks, 2016.

[32] Vivek N. et al. "On Field Performance Analysis of IEEE 802.11p and WAVE Protocol Stack for V2V & V2I Communication". In: *Information Communication and Embedded Systems (ICICES)*. 2014.

[33] M. Tamura and M. Ikeda. "1.68 μJ/signature-generation 256-bit ECDSA over GF(p) signature generator for IoT devices". In: *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. 2016.

[34] M. Knezevic, V. Nikov, and P. Rombouts. "Low-Latency ECDSA Signature Verification - A Road Toward Safer Traffic". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2016).