CONClave - Secure and Robust Cooperative Perception for CAVs Using Authenticated Consensus and Trust Scoring

Edward Andert
Arizona State University
Tempe, Arizona, USA
edward.andert@asu.edu

Francis Mendoza
Arizona State University
Tempe, Arizona, USA
fmendoz7@asu.edu

Hans Walter Behrens
Arizona State University
Tempe, Arizona, USA
hwbehren@asu.edu

Aviral Shrivastava
Arizona State University
Tempe, Arizona, USA
aviral.shrivastava@asu.edu

ABSTRACT

Connected Autonomous Vehicles have great potential to improve automobile safety and traffic flow, especially in cooperative applications where perception data is shared between vehicles. However, this cooperation must be secured from malicious intent and unintentional errors that could cause accidents. Previous works typically address singular security or reliability issues for cooperative driving in specific scenarios rather than the set of errors together. In this paper, we propose *CONClave* – a tightly coupled authentication, consensus, and trust scoring mechanism that provides comprehensive security and reliability for cooperative perception in autonomous vehicles. *CONClave* benefits from the pipelined nature of the steps such that faults can be detected significantly faster and with less compute. Overall, *CONClave* shows huge promise in preventing security flaws, detecting even relatively minor sensing faults, and increasing the robustness and accuracy of cooperative perception in CAVs while adding minimal overhead.

ACM Reference Format:
Edward Andert, Francis Mendoza, Hans Walter Behrens, and Aviral Shrivastava. 2024. *CONClave* - Secure and Robust Cooperative Perception for CAVs Using Authenticated Consensus and Trust Scoring. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3658491

1 INTRODUCTION

Cooperative autonomous vehicle operation has the potential to make roadways dramatically more safe and efficient [14]. Even if all the technicalities of executing a cooperative maneuver can be solved, there are still problems securing them from unauthorized participants. Beyond that, there is the issue of preventing both malicious vehicles that intentionally disrupt a cooperative application as well as preventing faulty vehicles that unintentionally disrupt a cooperative application due to an error. For example, a malicious vehicle could try to get ahead in the queue for a cooperative intersection by falsifying data. A fault, on the other hand, could be an autonomous vehicle that has a sensor malfunction and is sharing bad data with another vehicle that is blindly trusting the data to see around a corner that is out of its sensor range. Even if the exact reaction to these types of situations could be vehicle or vendor specific, the overarching identification and prevention of all disrupting vehicles, whether intentional or not, is paramount to running successfully cooperation amongst autonomous vehicles.

Detecting malicious and unintentional faults requires multiple steps, including authentication and verification of incoming data [16]. However, without a trusted third party with its own sensors involved in every vehicle area network, the scope increases to include consensus [12]. Existing state of the art methods typically treat authentication, consensus, and trust scoring either separately or together in a limited subset of cooperative scenarios. Guo *et al.* propose a method to log events using blockchain, but their approach completely ignores the problem of keeping out unauthorized participants and also does not have any mechanism to keep authenticated users from making up events [8]. More recently, trust scoring methods have been used in place of proof of work. For instance, Mankodiya *et al.* [17] use a specialized ML bases trust scoring that could take the place of proof of work but it is not coupled with a consensus method and therefore cannot reap those extra benefits. Bhattacharya *et al.* do tackle the authentication and consensus problems at once, but too many assumptions are made for the specific application, and therefore their approach will not work for general cooperative scenarios [4].

This paper presents *CONClave* – an application-level network protocol designed for sensor networks that require reliable and trustworthy data in the context of Cooperative Autonomous Vehicles (CAVs) and Cooperative Infrastructure Sensors (CISs). The three primary contributions of *CONClave* are:

(1) A three party homomorphic hashing based authentication process which includes the manufacturer, a third party authority/government, and the vehicle itself. This inclusion ensures that all entities (CAVs and CISs) that wish to participate in the system must have the approval of both the manufacturer and governmental stakeholders.
(2) A BOSCO-based single-shot consensus protocol that works in a dynamically changing geo-spatial vehicular networks

This work was partially supported by funding from National Science Foundation grants CPS 1645578 and Semiconductor Research Corporation (SRC) project 3154.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

https://doi.org/10.1145/3649329.3658491

by limiting the latency and resource requirement of the consensus protocol on non-discrete sensed values. Instead of generating consensus on a common world-view, *CONClave* generates consensus on the individual world-view provided by each agent. This eliminates Byzantine attacks on the network, leaving the common world-view generation work for the next sensor fusion step.

(3) A perception trust scoring technique that reports an accuracy score by utilizing sensor and recognition pipeline characterization data as the accuracy predictor, allowing for errors to be detected down to the individual sensor level that are not picked up by other state of the art methods. This trust scoring technique is tightly coupled with the authentication and consensus step so that it can operate in place of a proof of work to improve real-time performance.

CONClave was tested against the state of the art trust scoring method Trupercept [13] using fault and malicious injection on a 1/10 scale model autonomous vehicles using a motion capture system as ground truth. 1100 faults and malicious attacks were injected over the course of 14 different scenarios while varying the severity and number of the fault/injection. *CONClave* detected 96.7% of the 300 sensor extrinsic faults injected, 83.5% of the 300 software faults injected, 67.3% of the 300 malicious injections and removals, and 100% of the 200 communication faults and malicious injections that we subjected it to. On the other hand, the state of the art method TruPercept only detected 29.6% of sensor extrinsic faults, 34% of software faults, 32.6% of malicious injections and removals, and 19.6% of the communication faults and malicious injections. Overall, *CONClave* had a mean time to detection that was 1.83x faster on average and 6.23x faster in the best case when compared to TruPercept on the faults TruPercept could detect.

2 RELATED WORK

Authentication: When distributed agents communicate in the field, authentication is critical or the network is open to Sybil attacks [7, 16]. Further complicating the issue, conditions often prevent real-time communication with a central server, and local resource constraints limit processing and storage [6]. Handy *et al.* assume a more difficult task with no centralized authority or setup phase, but participants establish keys with each new participant through a process that does not consider the Sybil threat [18]. Wang *et al.* rely on specialized hardware such as Physically Uncloneable Functions (PUFs), an impractical choice for real-world deployments [22]. Similarly, approaches that rely on trusted execution environments (TEEs) are susceptible to eventual compromise and key extraction (e.g. via cold boot attacks [9] or side channels [15]). To address these challenges, we use a three-way knowledge partitioning between a government entity, manufacturer, and each individual participant. To allow for reconstruction, we rely on an approach that allows for intermediate hash composition using the homomorphic hash tree described by Behrens *et al.* [2]. Though this produces larger hashes, it allows for asymmetric reassembly of hashes, compartmentalizing information, and preventing the compromise of any one party from undermining the security of the authentication protocol [3].

Consensus: In a distributed environment, cooperative perception algorithms can quickly succumb to byzantine faults [7]. Whether due to communication dropout or malicious intent, faults will manifest themselves as data corruption in the subsequent sensor fusion step. A popular way to solve these issues is byzantine fault tolerant consensus, however, consensus on non binary values is slow. Han *et al.* solve this by eschewing the need for consensus on all sensor values by bounding the problem to just nearby vehicle positions in a platoon in addition to many other specializations [10]. However, this approach will not work for general cooperative perception. To address this challenge, our approach relies on a semi synchronous distributed Byzantine tolerant consensus on the data each party sent, rather than coming to consensus on the correctness of that data. The correctness proof is left for the subsequent trust scoring step in the pipeline. This technique keeps the consensus itself lightweight and eschews the need for any proof of work by using the trust score of the sensed values computed next as substitute.

Trust Scoring: In a cooperative perception environment, a minor disagreement in sensor input caused by a sensor fault or malicious actor could result a catastrophic incident and must be prevented [11, 14]. Cavorsi *et al.* propose a method to apply a trust score against robots sensing local traffic in their region that can detect adversaries and lower the percentage error in the locally fused traffic estimate [5]. However, it is not clear how this method can be generally applied to cooperative perception nor does it take into account the expected accuracy of the sensors involved. Hurl *et al.* propose a cooperative perception specific trust scoring method and test it using simulated data [13]. The trust score is applied to a sensor fusion algorithm as a weight, and the result is a better sensor fusion. Their method is limited to the case that the CAV sensor configurations are uniform, containing both a camera and a LIDAR as they use the camera confidence as the expected accuracy of each sensed object and the LIDAR point count as a proxy for the visibility. To address this, we create a trust scoring system that uses a generalized error estimation technique for heterogeneous sensing platforms borrowed from Andert *et al.* while consuming the results of the previous consensus step to prevent byzantine faults [1].

3 OUR APPROACH

3.1 Overview

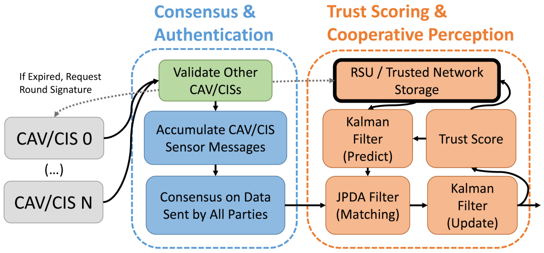


Figure 1: Overview of *CONClave*. Consensus and Authentication steps occur concurrently to reach a sensor data set that all participant CAVs and CISs agree upon. The sensor data set is then taken as input to our cooperative perception and trust scoring steps resulting in trust scores for each participant.

To achieve a secure consensus and trust scoring of CAVs and other sensing infrastructure for reliable cooperative driving, *CONClave* proposes a three-step process, which can be seen depicted at a high level in figure 1. First, all participants are authenticated to address the risk of Sybil attacks. We create a novel authentication scheme that leverages homomorphic hashing, incorporates both the manufacturer and a government entity, and allows participants to authenticate each other in such a way that participants in a consensus round don't always need to have communication with a trusted RSU. Next, we come to consensus on the sensor values that all participants submit to the consensus round using Byzantine fault tolerant consensus protocol such that faults in communication can such as packet delay or dropped messages don't manifest themselves later as error in the output [19]. We come to consensus on the sensor values that each participant sends such that we reduce the computation time by bounding the problem to be consensus on the sensor values each participant sent using the Bosco consensus protocol, which was modified to be semi-synchronous [20]. Finally, a trust scoring technique is applied to the sensing input set that results from the consensus round, to verify the correctness of the data each participant sent. Instead of using camera confidence values as an accuracy indicator like Hurl *et al.* use, we use parameterized sensor pipeline accuracy values from Andert *et al.* [1, 13]. This, along with being closely coupled with a sensor fusion technique, allows our trust scoring to be both fast and more accurate than the previous state of the art. Our trust scoring not only improves the accuracy of cooperative perception, it also serves as a replacement of the proof of work for our consensus step. All of this combines to prevent most known attack vectors and errors that can occur in a cooperative perception environment. Next, we explain the three steps of *CONClave* and its working in more detail.

3.2 Three Party Authentication

Algorithm 1: Three party authentication setup.

```
   Data: ego, mnf, gov, nearbyRSU, expirationTime
1  if ego->keyPair == false then
2      ego.$P_c$, ego.$S_c$, ego.$UUID_c$ = genKeyPair();
3      ego.$Chal_c$ = gov.genChal(ego.$UUID_c$, mnf.$S_m$, gov.$S_g$);
4      ego.$Resp_c$ = gov.genResp(ego.$UUID_c$, mnf.$S_m$, gov.$S_g$);
5      ego.keyPair = true;
6  if ego.roundToken.age() >= .9 * expirationTime then
7      if nearbyRSU.withinRange(ego.position) then
8          if roundNum.age() > expirationTime then
9              roundNum++;
10             $UUID_r$ = genUUID();
11             $sig_r$ = genRoundSig(roundNum, $UUID_r$,
                 mnf.$S_m$, gov.$S_g$);
12             transmitAllRSUsSecure(roundNum, $sig_r$,
                 $UUID_r$);
13             ego.$sig_r$ = transmitAllEgosSecure($sig_r$);
14         ego.$Chal_t$ = nearbyRSU.genChal(ego.$UUID_c$,
             ego.$Chal_c$, $sig_r$);
15         ego.$Resp_t$ = nearbyRSU.genResp(ego.$UUID_c$,
             ego.$Chal_c$, $sig_r$);
```

A high level depiction of our authentication setup process can be seen in algorithm 1. To initialize, both the law enforcement/governmental agency and manufacturer generate a secret key known only to themselves denoted as S_g and S_m respectively. The manufacturer generates an asymmetric keypair P_c, S_c and stores it locally on the vehicle; crucially, we do not require any central database of these keys (line 2). In an interactive process, the two players exchange the CAV/CIS's identity and generate a challenge $Chal_c$ and response hash $Resp_c$ which is also stored locally (line 3-5). This inclusion ensures that all CAVs/CISs which wish to participate in the system have the approval of both stakeholders. Note that neither party exchanges their secret keys S_m or S_g, instead using hashed versions $Chal_c$ and $Resp_c$ to prevent inappropriate use.

Next, both stakeholders must periodically go through an interactive process to refresh what we call as a *round signature* Sig_r – a validity mechanic that allows for either party to exit from participation (lines 8-13). Stakeholders may tune the frequency of this process to increase or decrease the duration in which CAVs/CISs may operate asynchronously. Once generated, these signatures Sig_r are securely distributed to each RSU. As vehicles travel within range of an RSU, they may choose to issue a renewal request to that RSU (line 7). These messages are encrypted with a CAV/CIS's private key, and the corresponding public key is transmitted along with the request to provide authenticity but not secrecy. Nonces prevent replay attacks. The RSU may optionally check the CAV/CIS's identifier against a central database to ensure compliance, such as valid licensing or inspection requirements. Once the RSU validates the request, ephemeral challenge and response *tokens* $Chal_t$ and $Resp_t$ are generated and encrypted with the CAV/CIS's public key before sending them back (lines 14, 15). This ensures that eavesdroppers may not re-use a CAV/CIS's token, as they lack the corresponding private key. Depending on how often rounds change, it may be desirable to store the subsequent tokens to ensure that validation can take place between CAVs/CISs whose round tokens differ in sequence by one.

3.3 Single-shot Consensus

For consensus rounds, we utilize a set area around an intersection with a constant trigger. All CAVs/CISs within range attempt to participate in the round and local IPs are known ahead of time. When establishing a relationship between CAVs/CISs during a given consensus step, each CAV/CIS provides additional metadata with their broadcast to allow for authentication. Each CAV/CIS generates this metadata $Chal_{c1}$, and shares it along with a hashed version of its ID ID_{c1} and its public key P_{c1}. Recipients check the received values using their own local tokens ID_{c2} and $Resp_{t2}$ to ensure compliance, and if valid, they temporarily store the public keys to allow for secure communication during the consensus step.

Next, participants accumulate the sensing messages from all other participants. This stops when a message is received from every known participant or the sensing transmission timeout is reached. Each participant sends out the accumulated set of sensing messages, received with valid authentication, and accumulates the same message from other participants. This stops when a message is received from every known participant or the aggregate transmission timeout is reached. Finally, each participant decides their vote according to the BOSCO algorithm and sends the result to all other participants, as well as nearby trusted RSUs for secure storage [20].

3.4 Accurate Trust Scoring

Algorithm 2: Trust Scoring.

```
Data: trustScores, sensorData, tracks, participants
Result: trustScores, tracks
1 tracks.predictEKF();
2 tracks.JPDAFAssociation(sensorData);
3 prelimResult = tracks.updateUKF(sensorData, trustScores);
4 exists = tallyExistenceVotes(sensorData, participants);
5 sensorDataTrun = remNonByzantine(exists, sensorData);
6 trustScores = calcSSDS(prelimResult, sensorDataTrun);
7 trustScores = enforceMinimums(trustScores);
8 tracks = tracks.updateUKF(sensorDataTrun, trustScores);
```

Our trust scoring method is closely coupled with a UKF based sensor fusion method, depicted in algorithm 2. It can be ran by all participants separately or alternatively ran on a nearby trusted RSU and distributed out. The first step of this is matching observations to tracks using a JPDA Filter while taking into account expected error and bounding box size [21]. We utilize the Unscented Kalman Filter (UKF) approach that Andert *et al.* use for fusion (lines 1-3) [1]. Utilization of observations that are coming from each vehicle is conditional upon the existing trust score, or sensing standard deviation score (SDS). If SDS exceeds a certain value SDS_{max}, the sensor platform is considered not trustworthy and none of its observations nor its own position will be included in this global fusion.

CONClave looks at two factors when calculating the trust score: i) Was an object supposed to be detected or not?, and ii) Was an object detected with the accuracy it was expected to be detected with? In order to evaluate the first, we determine if an object should have been seen or not with respect to other sensor platforms using a Byzantine tolerant voting scheme (line 4,5). We iterate through all the tracks and mark the track as existing if the object is within the FOV and range of a sensor as well as the visibility percentage threshold. Our method maintains byzantine tolerance by requiring that a majority of all participants should see an object according to their FOV and modeled obstructions to vote whether a track exists or not. For the second item, we utilize the estimated accuracy of the local fusion output of each vehicle as defined by Andert *et al.* [1], to determine the expected accuracy of each detection.

All participants within the sensor network need to have a minimum accuracy boundary, otherwise a possible attack vector would be to report its sensors as incredibly inaccurate. For sensing pipelines, we enforce a minimum sensing range requirement, a minimum FOV requirement, and a minimum sensor error magnitude requirement within this range and FOV. For localization pipelines, we enforce a minimum error magnitude requirement. If any participant's sensed values exceed these thresholds, the participant values will not be considered in the trust scoring and should be sent for repairs, even though it passes the authentication and consensus rounds.

Using the matching results from the JPDA filter and bounding box step as well as the fused positions outputted by the UKF for observability consensus, we have the necessary ingredients to perform a trust scoring of the reported accuracy of the sensor platform versus the fused position from the consensus round (line 6). If a track is reported as *exists*, the reported accuracy of the detected value of the CAV / CIS is compared with that contained in the fused output of the UKF. For the sensed values from each CAV/CIS that were matched to the track, the reported position $< x, y >$ (or z_k) is subtracted from the estimated position produced by the UKF (or $\hat{x}_{k|k}$), shown in the numerator of equation 1. $P_{k|k}$ returned by the UKF encompasses the expected error of the measurements from all sensors involved in the local fusion as well as the estimation from the UKF itself [1]. We then normalize by the expected error $\mathbb{E}(\mu_\theta^\alpha)$ which is contained in Σ_θ^α and can be extracted using the eigenvalues, shown in the denominator of equation 1. The result is why this is called the standard deviation score (SDS) as this is simply the standard deviation of the measured error returned by the global fusion *w.r.t* the value of the expected error from the measurement covariance in the local fusion of the sensor platform.

In order to keep a sensor from reporting itself as accurate in a vacuum, we enforce a rule of three – meaning at least three sensors must be matched and detecting the same track for a standard deviation frame to be added to the SDS revolving buffer for that object (line 7). This is in addition to the byzantine tolerant consensus on the existence or lack of existence of the track from all sensors. Furthermore, to keep low confidence tracks from being levied against a sensor, we enforce the second piece of the rule of three which dictates that the hypotenuse of the accuracy reported by the $roundFusion$ track (or $P_{k|k}$) must be three times as accurate as the hypotenuse of the accuracy reported by the sensor itself (or Σ_θ^α). If all of these conditions hold true, then it is prudent that the sensor α is attributed with the SDS by placing it in the last position of the revolving buffer, which is averaged to create the overall trust score. The rule of three applies to missed detection, three sensors must agree the object is there along with the consensus that the track exists. If these constraints are met, any sensor platform that should detect that object will have a missed detection frame added. The value ρ is then added in place of the SDS frame for the sensor platform that did not detect the track. ρ was set to $3 * min\_Sensor\_Accuracy$.

$$SDS_\theta^\alpha = \frac{\sqrt{(\lambda_0^\downarrow(\mu_\theta^\alpha) - \lambda_0^\downarrow(\hat{x}_{k|k_\theta}))^2 + (\lambda_1^\downarrow(\mu_\theta^\alpha) - \lambda_1^\downarrow(\hat{x}_{k|k_\theta}))^2}}{\sqrt{(\lambda_0^\downarrow(\Sigma_\theta^\alpha) - \lambda_0^\downarrow(P_{k|k_\theta}))^2 + (\lambda_1^\downarrow(\Sigma_\theta^\alpha) - \lambda_1^\downarrow(P_{k|k_\theta}))^2}} \quad (1)$$

For each participant in the round, a new SDS score is calculated in equation 1. This score is then added to the $SSDS$ buffer (line 6). The set of $SSDS$ scored for each participant constitutes their trust score. This value is shared globally among RSUs and is updated after each consensus round ends. Finally, the tracks are updated using the curated sensor set and new trust scores (line 8).

4 EXPERIMENTAL SETUP

For testing, we utilize 1/10 scale autonomous vehicle replicas. Our setup consists of four scale CAVs with a front facing 160 degree FOV camera and a 360 degree FOV single channel LIDAR and two mounted CISs with a 160 degree FOV camera. Figure 2 shows our setup with four CAVs.

Using data collected from a set of 10, ten-minute-long tests for each physical configuration, we perform error injection with the 14 scenarios shown in Table 1. The simplest scenarios are sensor errors

Error#: Error Name	Description
E1: Camera Shift	For CAV I, Camera extrinisics skewed N degrees.
E2: LIDAR Shift	For CAV I, LIDAR extrinisics skewed N degrees.
E3: Cam & LIDAR Shift	For CAV I, Camera and LIDAR extrinsics are skewed by the same N degrees.
E4: Random Data Loss	For CAV I, with probability N, each detection the vehicle has may be removed.
E5: Malicious Removal	For CAV I, with probability N, the CAV removes vehicles crossing the intersection.
E6: Malicious Insertion	For CAV I, with probability N, the CAV injects vehicle detections into the intersection.
E7: Localization	For CAV I, the localization error that CAV I is experiencing will be increased by N percent.
E8: Local Sensor Fusion	For CAV I, at the local fusion level the covariance of the LIDAR is decreased by N%.
E9: Global Sensor Fusion	For CAV I, at the global fusion level the covariance of the CAV I is decreased by N%.
E10: Unauthorized User	CAV I has an invalid authentication challenge.
E11: Expired Round Token	CAV I has a round token that has expired.
E12: Byzantine Fault	CAV I drops a packet when sending a message.
E13: Replay Attack	CAV I replays data another participant sent one round before.
E14: Spoofed Localization	CAV I sends the wrong location for itself.

Table 1: Descriptions and examples of the 14 error injection tests we performed against *CONClave* to test resilience.

that can be easily caused in any autonomous vehicle by jarring a sensor. For E1-E3, the same data from the sensors are used, but the extrinsics of the sensors will be skewed by N degrees resulting in a shift in the data from that sensor. The next category of error is malicious error in which we purposely inject or remove detection with probability N. Next, we have software errors that manifest itself as a bad weighting in the sensor fusion where we change weightings by some N percent. Finally, we have communication faults and attacks where we cause N vehicles in the simulation to experience a communication error. The tests are run for a random amount of time from 120 seconds through 540 seconds with normal operation before we begin injecting the specific fault. If the trust score for a vehicle becomes 1.2x of the baseline within 60 seconds after the fault is injected, we consider the fault to be caught and record the MTTD. Each fault injection was run 10 times at each step for a total of 1100 tests.

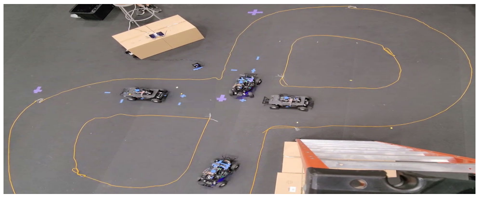


Figure 2: Four one-tenth scale CAVs with IMX160 camera, Slamware M1M1 LIDAR, and Nvidia Jetson Nano for onboard processing along with two one-tenth scale CIS traffic cameras using Jetson Nano and IMX160 camera test setup.

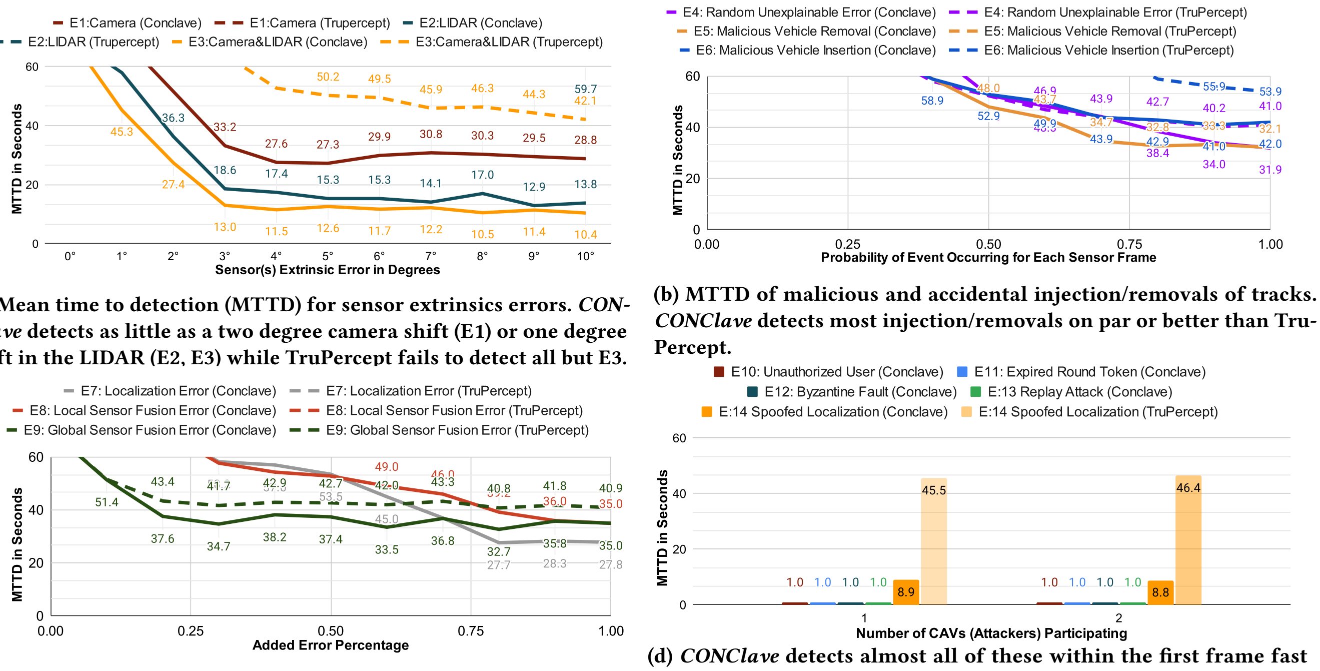
5 RESULTS

5.1 - *CONClave* quickly detects nearly all extrinsics errors E1-E3 where the sensors experience a physical sensor shift of N degrees are shown in figure 3a. These tests showcase *CONClave*'s ability to pick out relatively minor errors within the cooperative perception environment. Errors as minor as a two-degree camera shift, one-degree LIDAR shift, or one-degree shift combination are caught. *CONClave* detects an impressive 96.7% of the 300 tests. TruPercept is only able to catch large magnitude errors, such as a ten-degree LIDAR shift or a four-degree or more camera and LIDAR shift resulting in a detection rate of 29.6% of the 300 tests. TruPercept does not have a suitable predictor that works when the IOU match is still high, but the error variance is higher than it should be, instead relying on the confidence of the camera as a predictor [13]. Therefore, TruPercept only detects these sensor extrinsic errors when they result in the IOU of a track dropping below the 50% threshold, which causes there to no longer be a match to the rest of the CAV/CIS report and finally the offending CAV/CIS is punished for not detecting the object entirely. Conversely, *CONClave* has a concept for how much variance it expects in sensing error and as that variance starts to leave the expected limits, the vehicle trust score is punished for that. This variance threshold results in quick detection of minor extrinsics errors which are missed by TruPercept. RMSE of Conlcave in all tests for E1-E3 is 6.3% higher than TruPercept and 10.3% better than without trust scoring.

5.2 - *CONClave* detects malicious errors faster than TruPercept E4-E6 consist of accidental track removal as well as malicious injection and removal, seen in figure 3b. These tests are similar to what TruPercept was designed for, with the caveat that we only have a single vehicle experiencing the error, whereas TruPercept had all vehicles experiencing the same probability of error [13]. TruPercept performs well in this case, detecting 34% of 300 tests. *CONClave* beats TruPercept, detecting 67.3% of the 300 tests. This is because a malicious actor that is injecting fake vehicles into their data will not purposely report a low camera confidence. Therefore, in the second test, TruPercept relies completely on the IOU mismatch of detections to identify a problem where *CONClave* can detect higher variances and report errors sooner. For the RMSE case, we can see that both *CONClave* and TruPercept respond to E4, so *CONClave* is only 1.9% better than TruPercept and 5. 1% better than without trust scoring. Although these error injections have a high probability, they are filtered out by local sensor fusion, so the RMSE effect is less than E1-E3.

5.3 - *CONClave* detects more software errors, and faster E7-E9 look at common cases of mis-weighting. Overall Trupercept detected 32.6% of the 300 tests while *CONClave* was able to detect 83.5%, which can be seen in figure 3c. For the E7 localization error and E8 local sensor fusion misweight, we can see that *CONClave* detects it but Trupercept misses it due to *CONClave* being tuned to detect variance while Trupercpet is not. E9 on the other hand is detected by both methods with *CONClave* just slightly edging ahead in detection speed. Again, this is due to the nature of *CONClave* being able to detect larger and smaller variance than expected and report the errors quickly. Meanwhile TruPercept has to wait until detections start to mismatch IOU wise before it will start to detect the errors. Meanwhile, TruPercept takes longer to respond and therefore has a worse result. RMSE of *CONClave* was 3.9% better than Trupercept and 5.8% better than no trust scoring.

5.4 - *CONClave* detects many communication faults and attacks E10 - E14 showcase tolerance to a variety of errors and attack vectors that are not typically captured by a trust scoring system alone. This is apparent when looking at figure 3d on the bottom right where only one of the errors, E14, is detected by Trupercept.



(a) Mean time to detection (MTTD) for sensor extrinsics errors. *CONClave* detects as little as a two degree camera shift (E1) or one degree shift in the LIDAR (E2, E3) while TruPercept fails to detect all but E3.

(b) MTTD of malicious and accidental injection/removals of tracks. *CONClave* detects most injection/removals on par or better than TruPercept.

(c) Mean time to detection (MTTD) of localization errors. TruPercept fails to detect most of these while Conlcave succeeds.

(d) *CONClave* detects almost all of these within the first frame fast while TruPercept fails at all except localization spoofing. Note E10-E13 are not plotted for TruPercept as it does not detect them.

Trupercept detected 19.6% of communication faults and attacks while *CONClave* detected a perfect 100% of the 200 tests. Furthermore, *CONClave* detected E10-13 in less than two seconds, or two consensus rounds. We did not compare RMSE because Trupercept as well as the performance of the baseline technique, was rendered inoperable in the case of E10, E11, and E13 beyond recovery.

6 CONCLUSION

In this paper we present a method to secure cooperatively perception-based applications for connected autonomous vehicles that we call *CONClave*. *CONClave* consists of three parts, an authentication method, a consensus round, and a trust scoring method that are pipelined such that it can be run in real time. *CONClave* was able to detect more categories of faults and errors, including both malicious and unintentional errors, while being faster than the state of the art method TruPercept. In future work, we would like to expand *CONClave* to work for all cooperative driving scenarios, including those that need path plan trust scoring.

REFERENCES

[1] Edward Andert et al. 2022. Accurate Cooperative Sensor Fusion by Parameterized Covariance Generation for Sensing and Localization Pipelines in CAVs. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–8.

[2] Hans Walter Behrens et al. 2020. Pando: Efficient Byzantine-Tolerant Distributed Sensor Fusion using Forest Ensembles. In *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.

[3] Mihir Bellare et al. 1997. A new paradigm for collision-free hashing: Incrementality at reduced cost. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 163–192.

[4] Pronaya Bhattacharya et al. 2022. 6G-enabled trust management scheme for decentralized autonomous vehicles. *Computer Communications* 191 (2022), 53–68.

[5] Matthew Cavorsi et al. 2022. Exploiting trust for resilient hypothesis testing with malicious robots. *arXiv preprint arXiv:2209.12285* (2022).

[6] Dharminder Dharminder et al. 2021. Edge based authentication protocol for vehicular communications without trusted party communication. *Journal of Systems Architecture* 119 (2021), 102242.

[7] Mahdi Dibaei et al. 2020. Attacks and defences on intelligent connected vehicles: A survey. *Digital Communications and Networks* 6, 4 (2020), 399–421.

[8] Hao Guo et al. 2020. Proof-of-event recording system for autonomous vehicles: A blockchain-based solution. *IEEE Access* 8 (2020), 182776–182786.

[9] J. Alex Halderman et al. 2009. Lest We Remember: Cold-Boot Attacks on Encryption Keys. *Commun. ACM* 52, 5 (May 2009), 91–98.

[10] Jinheng Han et al. 2022. Distributed finite-time safety consensus control of vehicle platoon with senor and actuator failures. *IEEE Transactions on Vehicular Technology* 72, 1 (2022), 162–175.

[11] Amal Hbaieb et al. 2022. A survey of trust management in the Internet of Vehicles. *Computer Networks* 203 (2022), 108558.

[12] Jianhua He et al. 2019. Cooperative connected autonomous vehicles (CAV): research, applications and challenges. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 1–6.

[13] Braden Hurl et al. 2020. Trupercept: Trust modelling for autonomous vehicle cooperative perception from synthetic data. In *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 341–347.

[14] Mohammad Khayatian et al. 2020. A survey on intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems* 4, 4 (2020), 1–27.

[15] Chen Liu et al. 2022. Frequency Throttling Side-Channel Attack. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1977–1991.

[16] Zhaojun Lu et al. 2018. A survey on recent advances in vehicular network security, trust, and privacy. *IEEE Transactions on Intelligent Transportation Systems* 20, 2 (2018), 760–776.

[17] Harsh Mankodiya et al. 2021. XAI-AV: Explainable artificial intelligence for trust management in autonomous vehicles. In *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. IEEE, 1–5.

[18] Tarak Nandy et al. 2021. A Secure, Privacy-Preserving, and Lightweight Authentication Scheme for VANETs. *IEEE Sensors Journal* (2021), 1–1.

[19] Farzad Sabahi. 2011. The security of vehicular adhoc networks. In *2011 third international conference on computational intelligence, communication systems and networks*. IEEE, 338–342.

[20] Yee Jiun Song et al. 2008. Bosco: One-step byzantine asynchronous consensus. In *Distributed Computing: 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings 22*. Springer, 438–450.

[21] Lennart Svensson et al. 2011. Set JPDA filter for multitarget tracking. *IEEE Transactions on Signal Processing* 59, 10 (2011), 4677–4691.

[22] Weizheng Wang et al. 2021. Blockchain and PUF-based Lightweight Authentication Protocol for Wireless Medical Sensor Networks. *IEEE Internet of Things Journal* (2021), 1–1.