AVGuard: A Forensic Investigation Framework for Autonomous Vehicles

Mohammad Aminul Hoque
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
mahoque@uab.edu

Ragib Hasan
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
ragib@uab.edu

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500652

***Abstract*—Autonomous vehicles (AVs) rely on on-board sensors and computation capabilities to drive on the road with limited or no human intervention. However, autonomous driving decisions can go wrong for numerous reasons, leading to accidents on the road. The AVs lack a proper forensics investigation framework, which is essential for various reasons such as resolving insurance disputes, investigating attacks, compliance with autonomous driving safety guidelines, etc. To design robust and safe AVs, identifying the actual reason behind any incident involving the AV is crucial. Hence, it is essential to collect meaningful logs from different autonomous driving modules and store them in a secure and tamper-proof way. In this paper, we propose *AVGuard*, a forensic investigation framework that collects and stores the autonomous driving logs. The framework can generate and verify proofs to ensure the integrity of collected logs while preventing collusion attacks among multiple dishonest parties. The stored logs can be used later by investigators to identify the exact incident. Our proof-of-concept implementation shows that the framework can be integrated with autonomous driving modules efficiently without any significant overheads.**

***Index Terms*—autonomous vehicle, forensics, security**

I. Introduction

An Autonomous vehicle (AV) is a complex cyber-physical system capable of understanding the road condition and making driving decisions accordingly. The emergence of high-performance computing hardware, development in deep learning algorithms, and improved sensing technology together allow AVs to move on the road gradually. Multiple technology and auto manufacturing companies are working to improve autonomous driving (AD) and advanced driving assistance system (ADAS) [1]. Several ride-sharing companies are providing public ride-sharing services using AVs [2], [3]. The self-driving cars rely on sensors and AD software to make driving decisions on the road. However, sensors and AD software may provide wrong decisions under various circumstances such as hardware failure or cyber-attacks on AV hardware and software. Moreover, the AV safety authority needs to review AVs' performance and compliance with the safety guidelines regularly. Furthermore, if an AV becomes associated with an accident, it is important to figure out the exact reason to resolve the insurance-related disputes. Due to such requirements, a trustworthy forensics investigation framework is essential to collect evidence from AV, ensure secure storage of the evidence, and present the proof to the investigators.

Digital forensics is a process that allows collecting, preserving, and analyzing the incidents that take place in a system. This applied science identifies an incident and collects, examines, and analyzes the collected evidence data [4]. A well-designed forensics investigation framework allows the investigator to properly understand what happened by answering the questions of who, when, how, and why an incident was executed in a system [5]. The digital forensics investigation framework has been used in different domains such as cloud computing [6], Internet of Things (IoT) [7], computer networks [8], etc. However, the traditional forensics approaches cannot meet the requirements of AVs due to various reasons. The large volume of sensor data brings new challenges to AV forensics. Wide range of possible attacks on AVs and high frequency of decisions generated by the AD software also impose challenges to forensics investigations. A different combination of sensing and computation hardware used by the different manufacturing companies makes the problem more difficult. Current research works do not answer the following questions: (i) How can we collect and store evidence from AVs? (ii) How can we ensure confidentiality, integrity, and secure provenance of the evidence? (iii) How can we verify the integrity of the obtained evidence? (iv) How can we detect the source of the fault if an AV is associated with an accident on the road?

Considering the questions posed above, we present *AVGuard* – a forensic investigation framework for AVs. AVGuard extracts important information from different AD modules and stores them in a secure and tamper-proof way. AVGuard proposes to maintain AD decision provenance in two different ways: hash-chain and bloom filter. The framework creates log proofs and publishes the proofs on the web at the end of each day for future investigation. After an incident, the appointed forensic investigator extracts the related logs, verifies the log integrity using the log proof, and figures out the incident exactly. The framework also helps to resolve disputes among different stakeholders of AV. Our proof-of-concept implementation for AD perception module using Waymo open source benchmark dataset [9] demonstrates the feasibility of the framework.

Contribution: The contributions of this paper are as follows:

1) We propose *AVGuard* – a forensic investigation framework for autonomous vehicles.
2) We introduce a tamper evident scheme to prevent different malicious entities from manipulating the collected evidence after-the-fact.
3) We perform security analysis of AVGuard, implement a prototype using autonomous driving benchmark data from Waymo and demonstrate its feasibility.

Organization: The rest of the paper is organized as follows: Section II and III provides the motivation and challenges of designing AV forensics framework. Section IV explains the

building blocks of AVGuard framework. Section V contains the experiment and evaluation of the framework. Section VI presents the related works and we conclude in section VII.

II. Motivation

Irrespective of the after-effect of an accident involving an AV, it causes damage to the associated parties, such as auto manufactures and insurance companies. As more than 40 companies are competing in the AV market [1], such accidents can hugely damage rival companies' reputation. We explain the motivation for the AV forensics framework below:

A. NHTSA guidelines

National Highway Traffic Safety Administration (NHTSA) provides a 12 step guidelines to ensure the safety of the AVs [10]. Though these are only the elementary guidelines, and the manufacturers are not required to follow them completely, the guidelines are highly helpful for enhancing the safety of AVs. A proper forensics investigation framework can determine whether an AD system follows the guideline properly.

B. Insurance dispute

The insurance model is expected to be reformed once AVs start to move on the road in a significant number. We already observe several incidents where the AVs are involved in an accident [11]. After such misfortunes, proper forensic investigation is essential to resolve insurance disputes.

C. Attack strategies in autonomous vehicles

AV forensics framework can be helpful to analyze the impact of different attacks on AV hardware and software. Most important attack strategies that can force the AV to involve in an accident or reveal confidential information are:

Spoofing attacks: Different sensors of AV can be the victim of spoofing attacks. Lidar spoofing attacks can cause freezing or collision attacks [12]. GPS spoofing attacks can force the AV to depart from the lane [13]. Cameras are also vulnerable to spoofing attacks that may cause the AV to move out or move in to the lane [14].

Cache side channel attack: A malicious software can monitor the cache activity of the victim vehicle and relate possible routes with it. The cache activity becomes much higher when the AV makes a turn which is exploited by the attacker to to predict the destination in a known environment [15].

Adversarial attacks: Adversarial attacks fool the machine learning model of AV by generating adversarial examples that contain small noise or perturbation. For example, adversarial LiDAR point clouds try to minimize the model loss and bias the model to adversarial examples. Subtle adversarial manipulation in images and trojan attacks can also cause accidents.

D. Hardware and software failure

Many hardware devices work together to make the AV work correctly. However, a hardware may fail due to different reasons, such as wear out of silicon, broken connections, cosmic radiation, or magnetic fields. Faulty electrical wearing and computing hardware chip can also lead to hardware failure. Non-functional sensors (noisy sensor data or inaccurate perception) can be responsible for software failures. A forensics investigation framework is required to identify such failures.

III. Challenges of Autonomous Vehicle Forensics

As a complex cyber-physical system, the AV imposes new challenges in log collection, maintaining log integrity, and proof generation. In this section, we explore the challenges of AV forensics and analyze the related threat model.

A. Challenges

AV forensics framework imposes several unique challenges for forensics investigation along with the traditional challenges of digital forensics. Significant challenges of AV forensics are:

Huge amount of data: An AV generates a huge amount of data each day. In each second, radar and ultrasonic sensors generate 10-100 KB, GPS generates 50 KB, cameras generate 20-40 MB, and LiDAR generates 10-70 MB data. On average, an AV generates 4000 GB of data each day. Such massive amount of sensor data impose challenges in AV forensics.

Log accessibility: Proper access control of information and logs collected from AV is essential. The developers of AD and the investigators should be able to access the low level information. However, any third party malicious entity should not be able to extract any meaningful data from the logs.

Evidence Examination: An AD system may generate a considerable amount of logs due to dynamic road scenarios. The investigator may face challenges to correlate the logs of different AD modules to extract crucial information as all the AD components work together. The investigator also must be able to construct the proof in front of the court.

Evidence integrity: Current event data recorder systems installed in the AVs do not ensure the integrity of the collected data. A dishonest party can manipulate the data before presenting it as evidence. Hence, the integrity of the collected data is essential and challenging in AV forensics.

B. Threat model

A threat model helps to identify and prioritize potential attacks on a system. Here, we analyze the attackers, attacker model, and potential attacks on AV forensics framework.

1) Assets: Assets are the most important things of a system that are targeted by the attackers. The most important assets of AV forensics are the log proof and the ordering of the log proof. The attackers can be interested in these assets because they may try to frame an honest entity or extract information regarding the user destination, AD software, and sensors.

2) Attacker's capability: In the AV forensics framework, a dishonest auto manufacturing company may alter the collected logs. Moreover, the car owner may have white-box access to the AD system that enables her to modify or insert fake logs. The investigator assigned by the law enforcement agency may collude with an auto manufacturing company or car owner and alter the logs. Finally, an external attacker may install malicious software that may interfere with the log collection process. Despite such capabilities, we assume that the AVGuard can collect the AD logs and publish the proof on the web.

3) Possible attacks: Possible attacks on forensics investigation framework of AV are listed as follows:

Log modification: All the entities related to AV forensics such as car owners, manufacturing companies, and the investigator may try to modify, alter, or insert fake logs.

Confidentiality attacks: An external attacker or dishonest entity may extract information from the collected logs regarding the user and the AV. Attacker can gain knowledge regarding the sensors and AD software from the logs which may allow to launch more sophisticated attacks on the AVs.

Repudiation by the user: In the advanced driving assistance system (ADAS), such as L2 and L3 autonomous driving, a driver can be fully responsible for an accident despite the help from ADAS system. In that case, the user may claim that the logs are altered or belong to another driver.

Repudiation by the auto manufacturing company: An auto manufacturing company may deny a log provenance after-the-fact, which may contain proof of faults that occurred due to software or hardware issues in the AV.

IV. Building Blocks of AVGuard

In this section, we explain the building blocks of AVGuard that contains log collection from different AD modules, proof creation, and proof verification processes. The AVGuard is supposed to be integrated with the AD system and trusted for forensic investigation in AVs. We also assume that the AV is equipped with 4G/LTE/5G communication devices.

A. Log provenance approach

We propose two separate approaches for log provenance: hash chain and bloom filter.

Hash chain: Hashing is one of the most primitive approaches for integrity preservation. Hash chain preserves the ordering of the hash values. The hash of the i^{th} log is appended to the $(i+1)^{th}$ log to create the $(i+1)^{th}$ hash element. For the first element, hash of the first log uses an initialization vector to begin the chain creation process.

Bloom Filter: A Bloom Filter is a probabilistic data structure used to identify whether an element is a member of a set [16]. The Bloom Filter can identify the membership without any false-negative result. It accumulates the hash values of the logs, which is used for membership verification later.

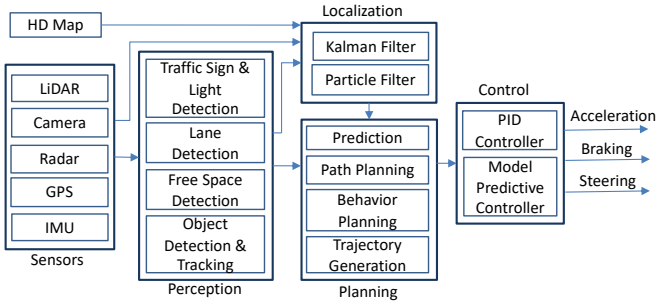

Fig. 1. Autonomous vehicle components

B. Log entry description

AVGuard identifies different events and generates logs from different AD modules. The module-based AD system has five components: sensors, perception, localization, planning, and control. Figure 1 shows the details of all the AD components. Here we explain the formats of the logs that are collected from each of the modules. In each collected log, the first and the second elements denote the module name and timestamp.

Sensors module: All the sensors that are installed in the AV are set to report in a fixed frequency. For example, the cameras are usually set to capture 30 frames per second, and LiDARs report data in 10 Hz frequency, which generates around 10000 points in each scan. AVGuard creates a log each second using the camera, LiDAR, and GPS readings. Format for sensors module log entry is:

LE=<Sensor, T, CameraFPS, LiDARPointCount, GPSFreq>

The log structure uses LiDAR, camera, and GPS considering them as the most important sensors for AD. Readings from other sensors, such as radar, can also be included in the logs.

Perception module: The perception module is responsible for understanding the objects and their movements from sensor data. The main objects required to be recognized by the AV are pedestrians, cars, traffic lights and their condition, road signs, and lanes. The AVGuard framework collects a log for a frame. Log entry structure for perception module is:

LE =< Perception, T, cars, pedestrians, trafficLight, roadsign, laneDetectionConfidence, undefindedObjects>

The perception log structure will help to identify different attacks on AVs later. For example, a LiDAR spoofing attack may create multiple copies of an object, hide an object, or create a fake object - the logs record all of these. Moreover, this strategy will also help identify whether an AV involves an accident due to object recognition failure. The perception module does not require to create logs for each frame as the logs of the adjacent frames will contain similar information. Instead, the module can select a certain number of frames each second and create logs by extracting the required information.

Localization module: The localization module estimates the current location of the ego vehicle within 2-10 centimeter accuracy. This module performs state estimation by fusing sensor data using a multi-sensor fusion (MSF) model through the Extended Kalman Filter. Moreover, this module uses an HD map to estimate the current location using a particle filter, which updates the belief regarding the AV's location using the perception module's output. A new log entry is created when the localization module finds out a new landmark on the road.

LE = <Localization, T, landmark>

Planning module: The planning module predicts the motion of other objects and performs path planning along with trajectory generation. For path planning, the AV uses a finite state machine (FSM) to generate trajectory considering other agents' movement. A state transition in the finite state machine takes place once the module decides to change the maneuver. Format of logs in the planning module is:

LE =< Planning, T, FSM>

Here, FSM denotes the current state of the AV in the finite state machine. As an example, Carla [17] is an open-source autonomous driving simulator that implements the planning module with a finite state machine that has five states: (i) road-following, (ii) left-turn, (iii) right-turn, (iv) intersection-forward, and (v) hazard-stop. For each transition in the finite state machine, a new log entry is created.

Control module: The control module is responsible for executing the vehicle control commands after generating the trajectory. It runs the acceleration, brake, and steering command

to control and move the vehicle forward to reach the destination. Log entry for the control module is defined as follows:

LE =<Control, T, acceleration, brake, steering>

For the control module, new log entries are added when the module decides to accelerate or brake. The steering angle defines the direction of the vehicle. If the steering angle crosses a threshold, then a new log entry is created. In each log entry, one or two out of the three values may be empty.

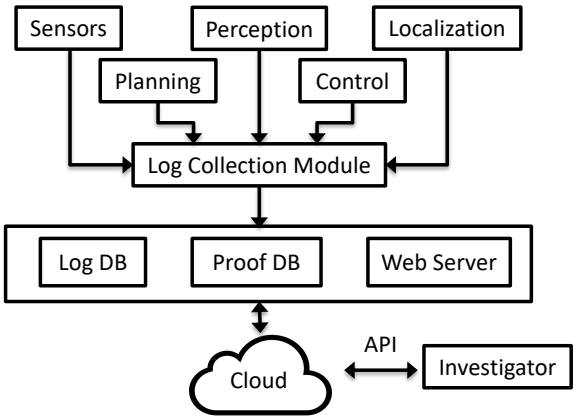

Fig. 2. Overview of the AVGuard Framework

C. Proof creation in AVGuard

We assume that the AV has adequate local storage to store the log provenance. The AV can also communicate with a remote cloud server and publish the newly created log provenance. A robot operating system (ROS) node collects all the logs from different AD modules. The ROS node works in a publisher-subscriber architecture, and each of the modules broadcasts the logs after creating them. Let us consider that one log entry collected from an AD module is as follows:

LE = <Module, T, val1, val2, val3,....>

The log entry, except the timestamp, is encrypted with the public key of law enforcement agency as follows:

ELE = $< PK_{LEA}(Module, val1, val2, val3, ...), T >$

Here, PK_{LEA} denotes the public key of the law enforcement agency. The log is encrypted to ensure the confidentiality of the AD data, which otherwise can be used to infer user information, AD algorithms, hardware properties, etc. To ensure integrity, we hash the entry and chain it with the previous log entry.

HLE = $< H(ELE, HLE_{prev}) >$

The HLE_{prev} is the hashed log chain up to the previous log. The ELE is signed by the private key of the manufacturing company. A log proof LP is created by concatenating the vehicle identification, user identification, signed ELE, and HLE. Storing both the userId and vehicleId is essential because the AV can be used in a ride-sharing service. T is the log timestamp that is used with userId and vehicleId for searching the logs from the storage later during the investigation.

LP = $< HLE, S_M(ELE), vehicleId, userId, T >$

Here, $S_M(ELE)$ is a digital signature computed on the encrypted log entry (ELE) using the private key of the auto manufacturing company. Finally, the log proof is added to a log proof chain (LPC). If there are n log proofs created for a journey, then LPC is created as follows:

LPC = $< LP_1, LP_2, LP_3, ..., LP_n >$

The LPC is saved into the log database in the AV. After reaching the destination, the log collection module retrieves the LPC for that journey and stores in the proof database. Finally, at the end of the day, the log collection module publishes all the LPC in the web. A LPC is considered as the story of a journey from origin to destination. A different LPC is created when another journey starts.

The proof creation can also be performed using a bloom filter. For bloom filter based proof creation, a log proof tuple is created using HLE, ELE, vehicleId, and userId. Later, the tuples are stored in the log database. Once the AV reaches the destination, all the log proofs are restored from the log database and inserted into the accumulator after hashing. Hence, the final log proof (FLP) is created as follows:

FLP = $< H(AE_D), S_M(AE_D), t, userId, vehicleId >$

Here, t is the signature timestamp, and AE_D is the accumulator entry for each journey, signed by the auto manufacturing company's private key. The tuple is stored in the proof database. All the FLPs and the log proofs are published on the web at the end of the day. Figure 3 shows the operational model for log proof creation in the AVGuard framework.

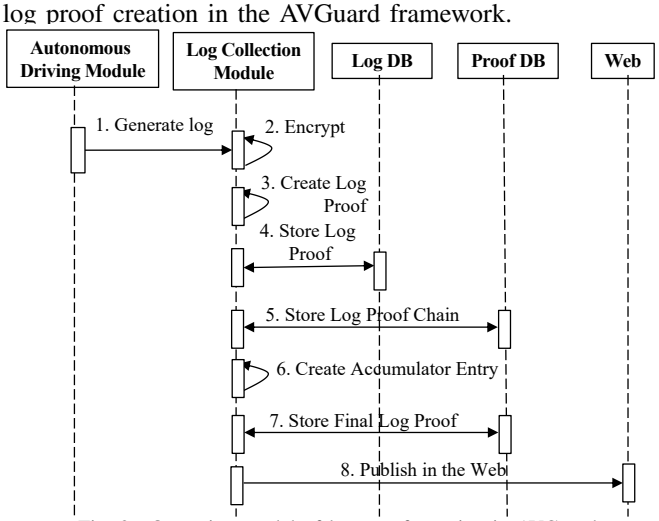

Fig. 3. Operation model of log proof creation in AVGuard

In step 1, different AD modules create logs and send them to the log collection module. The log collection module encrypts the log entry in step 2 and creates the log proof in step 3. The log proof is stored in the log database in step 4. For log chain-based proof creation, the log proof chain is extracted at the end of the journey and stored in the proof database in step 5. For bloom filter based proof creation, the log collection module uses log proofs from log database to create accumulator entry in step 6 and stores the final log proof in the proof database in step 7. Finally, the proof is published on the web in step 8.

D. Proof verification

The investigator extracts the log proof chain or accumulator entry using web API based on timestamp, userId, and vehicleId. The proof verification consists of two steps: integrity verification and sequence verification. We explain the verification process for both hash chain and bloom filter.

Hash chain: For hash chain based proof verification, the log proof chain (LPC) is extracted, and the integrity verification is

performed for each of the log proofs in LPC. Initially, the first log proof LP is taken from the LPC, and ELE is extracted using the auto manufacturing company's public key. Then the initialization vector is concatenated with the ELE of the log proof and the hash of the concatenated value is matched with the stored HLE of LP. If the hashes match, then the integrity is confirmed, and the next log proof is extracted to perform the same operation. Otherwise, the investigator decides that the evidence is tampered with by a malicious entity.

Bloom filter: For integrity verification in bloom filter-based proof, the Accumulator Entry (AE_D) is decrypted using the auto manufacturing company's public key. After signature verification, the hash is calculated and matched with the hash stored in the FLP. The proof is rejected if the signature or hash verification fails. Later, all the log proof tuples are extracted using web API, and their membership of the accumulator entry is verified. The sequence verification is performed by calculating the hash of the concatenation of ELE with the previous HLE. Later, the hash is matched with the HLE stored in the log proof tuple. If any log is altered, the sequence verification fails.

E. Security Analysis

In the AV forensics investigation, the related entities are the auto manufacturing company, car owner/ride-sharing service user, and the investigator. For security analysis purposes, we define the following symbols: M denotes an honest manufacturing company, M' denotes a dishonest manufacturing company, U denotes an honest car owner, U' denotes a dishonest car owner, I denotes an honest investigator, and I' denotes a dishonest investigator. All of them can be malicious individually, or they can collude among themselves. Table I shows the overview of possible attacks and motives. The AVGuard framework can prevent all possible collusion attacks. For hash chain based proof verification, the presence of any altered or reordered logs breaks the log chain, and the verification is rejected. For example, if the investigator tries to insert some fake logs to frame an honest auto manufacturing company, the ELE and HLE are changed. Hence, when the next ELE is hashed to match the HLE, the algorithm rejects it. Again, for bloom filter-based proof verification, the hash of the fake log gets rejected in the membership verification process. Hence, the AVGuard framework can successfully detect any confidentiality and integrity violation listed in table I.

V. Experiment and Evaluation

In this section, we explain our implementation of AVGuard framework prototype for the perception module and analyze the performance based on proof creation time, verification time, and storage overhead.

A. Experimental setup

Currently, there are several sources of high-quality AD benchmark data available. Waymo open dataset [9] is an open-source high-quality and diverse dataset that contains 1150 scenes with a span of 20 seconds each. The data were collected using five LiDAR sensors and five high-resolution pinhole camera. Each frame of the dataset is annotated with five information: count of vehicles, pedestrians, traffic signs, cyclists, and unknown. We extracted frames using the APIs provided by the dataset and figured out the required information to generate the perception logs. We performed this operation for 12000 frames to generate an equal number of logs.

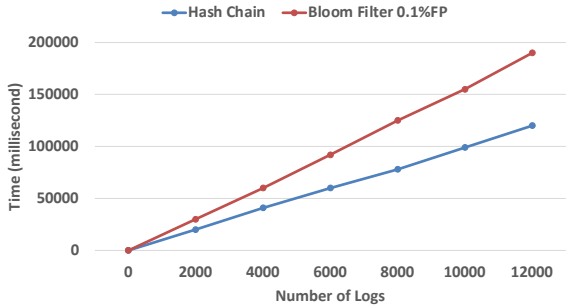

Fig. 4. Proof creation time required by hash chain and bloom filter

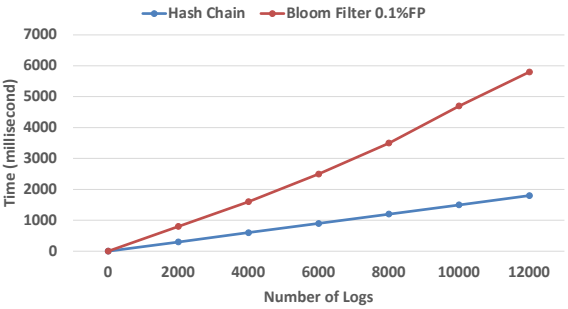

Fig. 5. Proof verification time required by hash chain and bloom filter

For cryptographic operations, we used 2048-bit RSA signatures and a 256-bit SHA-256 hashing algorithm. We extracted all the logs using python and stored them in a file. Later, we used the stored logs and implemented all the cryptographic operations in Java. We performed our experiment in a Macbook pro with a 2.3GHz dual-core Intel Core i5 processor with 8GB memory. The AD hardware contain more computation capability, hence, the computation should be faster in real AV.

B. Result

The proof creation time refers to the time required for creating the proof after receiving the logs from different AD modules. Figure 4 shows the proof generation time for different log provenance methods. The proof verification time refers to the time required for integrity verification and sequence verification. Figure 5 shows the proof verification time required by both the approaches. We observe that for both the cases, time increased linearly with number of logs where the bloom filter required more time than the hash chain.

We also analyzed the storage overhead induced by both the proof creation approaches. For the hash chain approach, each log proof required overhead of 32 bytes. On the other hand, we considered 1% and 0.1% false-positive rates for bloom-filter for 12000 logs, which required 14.04KB and 21.06KB of storage, respectively. AVGuard creates an accumulator entry for each journey. Hence, if an AV completes m number of journeys each day, then the bloom filter's storage requirement would be 21.06*m each day for 0.1% false-positive rate.

VI. Related Works

The forensics analysis framework has been studied widely in the context of connected vehicles. Hossain et al. proposed a

TABLE I
COLLUSION MODEL AMONG DIFFERENT RELATED STAKEHOLDERS OF AV FORENSICS

Is Honest?			Notation	Action	Motive
User	Manufacturing Company	Investigator			
Yes	Yes	Yes	UMI	No Attack	N/A
Yes	Yes	No	UMI'	Tampering logs	False accusation to car owner or auto manufacturing company
Yes	No	Yes	UM'I	Hide or alter logs before publishing in the web	False accusation to car owner or other interacting agents on the road to save company reputation
Yes	No	No	UM'I'	Hide or alter logs or repudiate published log proof	Collusion between manufacturing company and investigator to frame car owner or other interacting agents on the road
No	Yes	Yes	U'MI	Identify other users' destinations or analyze autonomous driving software and hardware	Use extracted information to launch different external attacks on AV
No	Yes	No	U'MI'	Hide or alter logs or repudiate published log proof	False accusation to auto manufacturing company to hamper their reputation
No	No	Yes	U'M'I	Hide or alter logs or repudiate published log proof	False accusation to other related agents or stakeholders such as insurance company
No	No	No	U'M'I'	Hide or alter logs or repudiate published log proof	False accusation to other related agents or stakeholders such as insurance company

forensics investigation framework for the internet of vehicles (IoV) named Trust-IoV [18]. T-Box is an automotive data recording method that collects information from an in-vehicle network [19]. It stores the collected information in a trusted execution environment (TEE). CVShield [20] is also a TEE-based sensor data integrity protection mechanism that relocates all the codes related to sensor data reading and processing from the rich execution environment (REE) to TEE. Oham et al. [21] proposed a blockchain-based liability attribution framework based on the evidence reported by the nearby witness AVs if an AV is involved in an accident. Besides the context of vehicles, researchers have proposed forensics investigation frameworks for different other domains, such as cloud computing [6], computer networks [8], internet of things [7], [22], etc. However, the forensics investigation framework is mostly unexplored for AVs. In this paper, we have addressed the issue by proposing AVGuard.

VII. CONCLUSION AND FUTURE WORKS

With more AVs having started to move on the road, they require a forensics investigation framework for different reasons such as improvements in AD, compliance with AD safety guidelines, resolving insurance disputes, etc. The forensics investigation framework ensures the secure storage of collected information and uses them to analyze an incident later. In this paper, we have proposed AVGuard - a forensic investigation framework for AVs that can ensure the collected logs' confidentiality and integrity and verify the generated proofs' integrity. The AVGuard framework will help to fulfill the requirements of different forensics investigation use cases for AVs. In the future, we plan to implement the log collection process for all other AD modules and analyze the performance.

ACKNOWLEDGEMENT

This research was supported by the National Science Foundation through awards ECCS-1952090, DGE-1723768, ACI-1642078, and CNS-1351038.

REFERENCES

[1] "40+ corporations working on autonomous vehicles," 2020. [Online]. Available: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/

[2] CNBC, "Waymo makes autonomous vehicles available to lyft riders," 2019. [Online]. Available: https://www.cnbc.com/2019/06/27/waymo-makes-autonomous-vehicles-available-to-lyft-riders.html

[3] Baidu, 2019. [Online]. Available: https://www.marketwatch.com/story/baidu-debuts-robotaxi-ride-hailing-service-in-china-using-self-driving-electric-taxis-2019-09-26

[4] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to integrating forensic techniques into incident response," *NIST Special Publication*, vol. 10, no. 14, pp. 800–86, 2006.

[5] R. Marty, "Cloud application logging for forensics," in *proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 178–184.

[6] S. Zawoad, A. K. Dutta, and R. Hasan, "Seclaas: secure logging-as-a-service for cloud forensics," in *ACM ASIACCS*, 2013, pp. 219–230.

[7] M. Hossain, Y. Karim, and R. Hasan, "Fif-iot: A forensic investigation framework for iot using a public digital ledger," in *2018 IEEE ICIOT*. IEEE, 2018, pp. 33–40.

[8] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics: Review, taxonomy, and open challenges," *Journal of Network and Computer Applications*, vol. 66, pp. 214–235, 2016.

[9] P. Sun, H. Kretzschmar *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *IEEE CVPR*, 2020, pp. 2446–2454.

[10] NHTSA, "Automated driving systems: A vision for safety 2.0," 2017.

[11] National Transportation Safety Board, 2018. [Online]. Available: https://j.mp/37WTAfC

[12] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *ACM CCS*, 2019, pp. 2267–2281.

[13] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing," *arXiv preprint arXiv:2006.10318*, 2020.

[14] S. Jha, S. Cui, S. S. Banerjee, Z. Kalbarczyk, and R. Iyer, "Ml-driven malware that targets av safety," *arXiv preprint arXiv:2004.13004*, 2020.

[15] M. Luo and G. E. Suh, "Stealthy tracking of autonomous vehicles with cache side channels," in *USENIX Security*, 2020, pp. 859–876.

[16] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[17] A. Dosovitskiy, G. Ros, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[18] M. M. Hossain, R. Hasan, and S. Zawoad, "Trust-iov: A trustworthy forensic investigation framework for the internet of vehicles (iov)." in *ICIOT*, 2017, pp. 25–32.

[19] S. Lee and D. H. Lee, "T-box: A forensics-enabled trusted automotive data recording method," *IEEE Access*, vol. 7, pp. 49 738–49 755, 2019.

[20] S. Hu, Q. A. Chen, and H. X. Liu, "Cvshield: Guarding sensor data in connected vehicle with trusted execution environment," in *ACM Workshop on Automotive and Aerial Vehicle Security*, 2020, pp. 1–4.

[21] C. Oham, S. S. Kanhere, R. Jurdak, and S. Jha, "A blockchain based liability attribution framework for autonomous vehicles," *arXiv preprint arXiv:1802.05050*, 2018.

[22] V. R. Kebande and I. Ray, "A generic digital forensic investigation framework for internet of things (iot)," in *IEEE FiCloud*. IEEE, 2016, pp. 356–362.