Transparent and Trustworthy Blockchain-based Scheme for the Protection of Vehicular Soft Integrity in Shared Mobility

Urooj Ghani, Mudassar Aslam, Subhan Ullah, Tahir Ahmad, Attaullah Buriro, Palash Yuvraj Ingle, and Rutvij H. Jhaveri, *Senior Member, IEEE,*

***Abstract*—The automotive industry is transforming from traditional private vehicle ownership to innovative shared mobility solutions, presenting unprecedented cybersecurity challenges. This transition introduces complex security vulnerabilities where malicious actors could exploit the access of a rental vehicle to manipulate the software systems on board. Unlike physical damage, which can be easily detected, software modifications represent an insidious threat that can compromise user safety and vehicle integrity. Our research proposes a blockchain-based approach to address these critical security challenges. We introduce a novel method for ensuring data authenticity and integrity within vehicle systems by leveraging blockchain's immutable ledger and advanced encryption technologies. Our methodology utilizes the Trusted Platform Module (TPM) to securely archive vehicle data in the central gateway, creating a tamper-evident environment that fundamentally transforms traditional data management approaches. The key innovation lies in the blockchain-based data binding process: when a user possesses a vehicle, they bind application-retrieved data with the vehicle's existing data and commit them to the blockchain. Upon vehicle return, any potential tampering can be immediately detected by comparing newly acquired data against pre-existing blockchain records. We develop a proof-of-concept implementation and demonstrate significant improvements in security architecture that offer a reliable alternative to conventional database-centric approaches. Comparative evaluations between database-centric and blockchain-centric architectures testify to the operational effectiveness and practical viability of our proposed solution. By addressing the inherent vulnerabilities in shared mobility ecosystems, this research contributes to a sophisticated technological intervention that enhances user safety, data integrity, and trust in emerging transportation paradigms.**

***Index Terms*—Shared Mobility, Blockchain, Vehicle Integrity, Vehicle Safety, Vehicle Security.**

Urooj Ghani is affiliated with the Department of Computer Science, FAST National University of Computer and Emerging Sciences, Islamabad, Pakistan (email: urooj.ghani@nu.edu.pk).

Mudassar Aslam is affiliated with the School of Engineering, Computing, and Mathematics, Oxford Brookes University, UK (email: maslam@brookes.ac.uk).

Subhan Ullah is affiliated with the Department of Cyber Security, FAST National University of Computer and Emerging Sciences, Islamabad, Pakistan (email: subhan.ullah@nu.edu.pk).

Tahir Ahmad is with the Security & Trust Unit of the Center for Cybersecurity, Fondazione Bruno Kessler (FBK), Trento, Italy (email: ahmad@fbk.eu). (Corresponding author: Tahir Ahmad.)

Attaullah Buriro is with the Faculty of Engineering at the Free University of Bolzano, Bolzano, Italy, and with Department of Computer Science and Electronic Engineering, University of Essex, at Colchester campus, UK. (email: ab25399@essex.ac.uk). (Co-corresponding author: Attaullah Buriro.)

Palash Ingle is affiliated with the Department of Computer and Information Security, Sejong University, Seoul, South Korea (email: Palash@sejong.ac.kr).

Rutvij H. Jhaveri is with the Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Energy University, Gujarat, India (email: rutvij.jhaveri@sot.pdpu.ac.in). (Co-corresponding author: Rutvij H. Jhaveri.)

I. Introduction

In the 19th century, the proliferation of vehicles due to population growth has led to traffic congestion. The urban transportation landscape has undergone a profound transformation, driven by increasing population density and vehicular expansion. To mitigate these adverse effects, the community introduced the concept of *shared mobility* [1], sharing transportation services and resources between users concurrently or sequentially for short-term access. Shared mobility represents an innovative approach to transportation resource management that enables dynamic and flexible access to mobility services. This paradigm addresses critical urban challenges such as traffic congestion, infrastructure limitations, and environmental sustainability by facilitating collaborative transportation strategies.

Shared mobility can be classified into two primary models: Business-to-Consumer (B2C) and Peer-to-Peer (P2P) shared mobility frameworks [2]. In the Business-to-Consumer model, companies strategically deploy and rent vehicles to the public. At the same time, the peer-to-peer approach allows individual vehicle owners to share their personal vehicles based on needs [3]. Technically, both models rely on third-party platforms that provide essential online infrastructure and comprehensive customer support [4]. Users use these services through smartphone applications, facilitating seamless vehicle booking and rental processes. Ensuring robust security and privacy protection for user information remains paramount, necessitating solutions that maintain operational vehicle performance [5], [6].

The shared mobility environment introduces complex security vulnerabilities due to multiple users' physical access. This accessibility creates potential attack vectors where malicious actors could compromise vehicular systems by manipulating software components, such as the Electronic Control Unit (ECU) or Gateway. Potential adversarial objectives range from privacy breaches to deliberate disruption of vehicular functionality. While existing research has extensively explored remote and physical attack methodologies, significant unresolved challenges persist regarding software integrity in shared mobility environments [7].

This study presents an innovative blockchain-based model to address the multifaceted challenges of security, trust, and scalability in car sharing environments, as illustrated in Figure 1. The proposed methodology comprises three critical components: Virtual Vehicle Simulator, Blockchain, and Android Application. The virtual vehicle simulator facilitates comprehensive data transmission, captures electronic control

unit (ECU) information during vehicle reservation and return processes, and securely stores these data as cryptographic hashes within the blockchain infrastructure [8]. The fundamental security mechanism of the model involves hash-based integrity verification. When a user books or returns a vehicle, the system generates and stores hash values representing the vehicle's state. Any unauthorized data modifications during the rental period will result in hash value alterations, triggering immediate security alerts for administrative intervention. This approach significantly surpasses the traditional database-centric methods discussed in previous research [9], offering superior data protection through the inherent security characteristics of the blockchain. Unlike conventional database systems, the proposed blockchain implementation provides robust, tamper-evident data storage [10], [11]. The decentralized nature of blockchain, combined with advanced encryption techniques, ensures that stored data remains immutable, inaccessible to unauthorized modifications, and resistant to deletion or destruction. Using blockchain cryptographic principles, the model establishes a secure, transparent, and scalable framework to manage vehicle data integrity in shared mobility ecosystems.

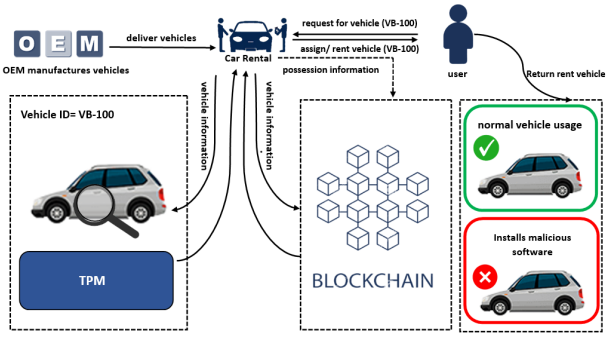

Fig. 1: Overview of the Proposed Model

This research addresses critical limitations in shared mobility security through a novel blockchain-based approach. Technically speaking, the study seeks to answer the following critical research questions: *1. How can blockchain technology enhance software integrity verification in shared mobility environments? 2. What are the inherent limitations of existing database-centric security approaches?* and *3. Can a decentralized, cryptographically secured system provide more reliable protection against unauthorized software modifications?* In this context, the study makes three fundamental contributions to the emerging field of secure, technology-enabled transportation systems:

1) **Innovative Blockchain-Enabled Integrity Verification**: We propose a blockchain model that transcends traditional database security paradigms. By leveraging blockchain's cryptographic architecture, our approach provides an unprecedented mechanism for verifying vehicular software integrity in shared mobility environments.
2) **Technological Trust Mechanism**: Our research introduces a transparent, decentralized trust framework that fundamentally transforms the reliability of shared mobility ecosystems. By implementing blockchain's immutable ledger technology, we establish a robust mechanism that ensures data integrity, provides real-time security monitoring, and creates a verifiable record of vehicular software modifications.
3) **Empirical Security Evaluation**: Through rigorous experimental validation, we demonstrate the practical efficacy of our proposed blockchain-based security model. Our comprehensive evaluation provides empirical evidence of the approach's potential to mitigate software manipulation risks in shared mobility scenarios.

Our research contributes to the broader discourse on innovative transportation technologies by bridging the technological gaps in shared mobility security. The proposed methodology advances academic understanding of blockchain applications and offers a pragmatic solution to real-world security challenges in collaborative mobility ecosystems.

The rest of the paper is organized as follows: Section II depicts the systematic literature reviews based on the secondary research method and attack surfaces. In contrast, Section III presents the problem formulation and proposed solution. Section IV explains the implementation of the proposed model. Section V discusses the evaluation of the implemented model from a security perspective. Finally, Section VI concludes and highlights the future directions of the proposed work.

II. Related Work

As connected, automated, and self-driving vehicles are increasing, the risk of attacks on vehicles also increases. Section III-B covers attack vectors and some attack scenarios [12], [13], [14], which demonstrates the critical imperative of comprehensive vehicular security strategies.

The contemporary automotive security landscape is characterized by complex technological challenges and evolving threat vectors. Nasser et al [15] comprehensively discusses the security challenges emerging from automotive technological advancements, proposing strategies to ensure vehicle safety within stringent constraints. The AUTOSAR [16] framework underwent rigorous security evaluation, representing Automotive Open System Architecture for Electronic Control Units (ECUs). Researchers identified critical security gaps through quantitative analysis, proposing an HSM-based monitoring system capable of rolling back measurements during potential vehicle attacks.

Recognizing the multifaceted nature of vehicular security, researchers have explored diverse mitigation strategies. Kornaro et al. [17] introduced TrustNet, a lightweight approach designed to address resource constraints while protecting Controller Area Network (CAN) buses from sophisticated replay attacks. The research specifically targeted man-in-the-middle and masquerade attack methodologies.

Vehicle locking systems represent a critical security domain. Researchers have extensively analyzed proprietary systems like the Passive Keyless Entry and Start System (PKES), revealing significant vulnerabilities in existing cryptographic implementations. Studies on Tesla Model S [18] and BMW

[19] vehicles exposed critical security weaknesses, demonstrating how advanced interfaces can simultaneously enhance functionality and expand potential attack surfaces.

Software update mechanisms have emerged as another crucial research focus. Wu et al. [20] proposed a distributed software update system involving original equipment manufacturers, software suppliers, and vehicle regulators. This collaborative approach aimed to ensure authentication and integrity during software modification processes. Kim et al. [21] further explored secure Over-the-Air (OTA) updates, highlighting their potential to address software bugs and introduce new features efficiently.

Blockchain technology has progressively gained recognition as a potential solution for vehicular data security. Kang et al. [22] demonstrated blockchain's effectiveness in secure vehicular data storage and sharing, introducing reputation-based schemes for authorized data exchange.

Falco and Siegel [23] and Tratter et al. [9] investigated software integrity maintenance, with the latter proposing the VesIPreS scheme utilizing the Trusted Platform Module (TPM) for integrity verification. However, existing approaches predominantly rely on centralized databases, limiting their ability to detect and prevent physical or remote attacks comprehensively.

While existing research has made significant strides in addressing vehicular security challenges, our proposed approach represents a paradigm shift in securing shared mobility ecosystems. Unlike traditional methodologies that rely on centralized database architectures with inherent vulnerabilities, our blockchain-based model introduces a fundamentally different security paradigm. Critically, our research transcends the limitations of previous studies by developing a scalable, adaptable security framework that can be seamlessly integrated into shared mobility ecosystems. While existing approaches often focus on isolated security aspects, our methodology provides a comprehensive, technologically advanced solution to the complex challenges of vehicular software integrity in shared mobility environments.

III. Problem formulation and Proposed Solution

This section presents the problem and explains how our approach addresses it.

A. Use Case Scenario

The proposed approach considers shared mobility scenarios in which the user has full access to the vehicle and the vehicle is used by multiple users. It targets the insider, an employee in the rental company, and the car user. Both can be attackers. If the attacker is an insider (employee), they have access to the database server used to store the information of the user and vehicle. The employee can tamper with the data present in the database. If the user is an attacker, they rent a car from the rental company, and when the car is returned, they can tamper with the Electronic Control Units (ECUs) running in the car. Some use-case scenarios of an attack are as follows:

1) The rentee uses a car from a car rental company. When they return the car, the physical damage can be seen by checking the car from the outside, but the software present in the car cannot be checked just by inspecting the car externally.
2) In the car-sharing scenario, registered users can rent a car and leave it anywhere in the business area. Still, no one can check the car after its return to ensure software integrity. In this case, the car's user may be an attacker who has tampered with the car's software and left it.
3) In transport companies where drivers operate buses, trucks, etc., they can easily manipulate the vehicle's software as the vehicles are not owned by them.
4) When the car owner gives their car for repair to a repair shop, the mechanic has physical access to the car, which can be exploited to manipulate the software without any mechanism for the owner to detect such changes.
5) When the car owner uses a valet service for parking and hands over the car to the valet, vehicle security can also be compromised.

There must be a mechanism to ensure the integrity of the software running in the vehicle when used by multiple users with complete physical access. The vehicle's software must be checked when the user rents a car and when they return it. The use case scenarios highlight the need for a solution that allows the vehicle's owner to detect any change in the vehicle's software.

B. Attack Surfaces

In a shared mobility environment, the attacker has physical access to the vehicle; our main concern is physical attack vectors, where the user has full access to the vehicle. If the user has full physical access to the vehicle, they can physically and remotely attack it.

1) Remote Attacks: A user present inside the vehicle can access the vehicle, but if the user has no physical access to the vehicle, they can still easily access it remotely. Remote access can be done through Bluetooth, Wi-Fi, or other interfaces. Vehicle manufacturers know these remote attacks and provide mechanisms to protect the vehicle. However, threats of remote attacks still exist, as discussed in [24] and [25]. An example of such a scenario is Cherokee Jeep Hack through Infotainment System [25]. According to Miller and Valasek [25], the communication interface of a Jeep was remotely hacked by researchers through the infotainment system (music system). They entered the Jeep remotely using the cellular network. Accessing the Jeep's infotainment system allowed them to change settings related to it, such as volume and interface view, and access information about the Jeep, such as its location. A chip in the Jeep's infotainment system, known as a proxy chip, had access to the Jeep's CAN buses. Miller and Valasek [25] accessed the proxy chip, installed malicious software on it through the infotainment system, and began sending random CAN messages to the Jeep. This access to the CAN bus allowed them to affect the functional safety of the Jeep, sending commands that altered components such as steering, brakes, airbags, and the engine.

2) Physical Attacks: Attackers with full physical access to the vehicle mostly carry out physical attacks. They use

the ports present in the vehicle as attack vectors to change the vehicle's components and disable the electronic control unit or networks. A port known as OBD (onboard diagnostics) is used for vehicle maintenance. Attackers can easily target the vehicle's electronic control unit through this port. An example of such a scenario is Hack BMW through the Diagnostic Connector [26]. According to [26], BMW uses a key with a computer chip for security purposes. The attacker can access the BMW by controlling the car alarm. After controlling the car alarm, the attacker gains physical access and reads information from the OBD (onboard diagnostics) port using sophisticated OBD readers like Actron, Innova, or CarMD. This port contains key-related information. The attacker decodes the key code information, programs the key, and steals the car.

C. Extended Attack Surfaces

Traditional attacks, such as remote and physical attacks on vehicles, have been discussed and mitigated to some extent in research. However, the emergence of new technologies has also introduced new vulnerabilities in vehicular security. This section highlights some of these emerging threats.

1) Side-Channel Attacks: Side-channel attacks exploit unintended information leakage, such as power consumption, electromagnetic emissions, or execution timing, to extract sensitive data like encryption keys or authentication tokens. Unlike other attacks that target vehicular software vulnerabilities, side-channel attacks focus on the physical characteristics of cryptographic operations. Modern vehicles use cryptographic security mechanisms in Electronic Control Units (ECUs) and Trusted Platform Modules (TPMs). Attackers can use techniques such as Differential Power Analysis (DPA), Electromagnetic Analysis (EMA), and Timing Attacks to break encryption and gain access to cryptographic keys. To mitigate these types of attacks, extensive research has been conducted. Researchers have explored side-channel-based intrusion detection systems (IDSs) to analyze vehicle voltage signals. A novel IDS has been proposed that maps ECU identifiers and employs a FeatureBagging-CNN model for detecting and locating malicious data frames from compromised ECUs and external nodes, enhancing CAN bus security without adding network overhead [27]. The resilience of automotive microcontrollers against Deep Learning-based Side-Channel Attacks (DL-SCA) has also been evaluated in [28].

2) Man-in-the-Middle (MITM) Attacks: A Man-in-the-Middle (MITM) attack occurs when an attacker intercepts and alters communication between two parties without their knowledge. These attacks compromise the integrity, confidentiality, and authenticity of transmitted data, leading to unauthorized control over ECUs and other critical vehicle functions. MITM attacks can be executed on the Wi-Fi surface of the vehicle, it allows an attacker to breach the telemetry data and manipulate real-time communication. A study demonstrated different kinds of attacks on autonomous vehicles and evaluates VPN-based mitigation techniques as an effective countermeasure [29].

In shared mobility scenarios, attackers can deploy rogue access points, signal jamming, or session hijacking techniques to compromise vehicle authentication mechanisms. Since attackers have full access to the vehicle, they can intercept and modify critical data exchanged between the vehicle, rental platform, or cloud services.

3) Quantum Security Attacks: Shared mobility vehicles rely on cryptographic mechanisms for secure user authentication, encrypted data transmission, and software integrity verification. Further, modern vehicles receive Over-the-Air (OTA) software updates to introduce new features, security patches, and performance improvements. These updates are protected using digital signatures and cryptographic keys, but quantum attacks could break these protections, allowing attackers to inject malicious firmware updates into vehicles. This could lead to malfunctioning brakes, altered acceleration settings, and disabling of critical security features. To mitigate the quantum threats, researchers have explored post-quantum authentication mechanisms for securing vehicle communications. A study highlights how post-quantum encryption and signal processing techniques can protect against both active and passive cyber-physical attacks in vehicular networks [30].

4) TPM Hardware and Firmware Attacks: The Trusted Platform Module (TPM) is a critical component in the vehicles because it secures cryptographic keys, firmware integrity, and authentication. Research shows how security mechanisms based on TPM remain receptive to hardware vulnerabilities, emphasizing the need for cryptographic key protection and secure firmware integrity enforcement [31].

Cold Boot Attacks: The attacker freezes the DRAM of the vehicle's system to extract encryption keys stored in the TPM's memory. Since memory retains data briefly after power loss, attackers can recover keys and authentication credentials, allowing them to bypass secure boot mechanisms and gain control over ECUs. To prevent this attack, memory encryption should be implemented, TPM key sealing should ensure that keys are bound to specific hardware, and immediate memory erasure should occur when power is lost.

Key Extraction Attacks: The attacker exploits firmware vulnerabilities to extract cryptographic keys using techniques such as side-channel attacks. By recovering keys, attackers can compromise authentication, firmware verification, and ECU security. To mitigate these threats, firmware attestation should be used to verify updates, a secure update mechanism should prevent unauthorized modifications, and side-channel countermeasures such as noise injection and randomized execution should protect cryptographic operations.

Rollback Attacks: In this attack, the attacker forces the TPM to load an outdated, vulnerable firmware version, introducing security threats into the system. This weakens secure boot, authentication, and firmware integrity checks, allowing attackers to bypass security restrictions. To mitigate such attacks, firmware version binding should ensure that only the latest versions are accepted, blockchain-based integrity verification can track and secure firmware updates, and anti-rollback mechanisms should prevent older versions from being reinstalled.

Our proposed model in this research ensures software integrity in shared mobility scenarios, and the use of blockchain technology strengthens the defense against emerg-

ing threats. Additionally, blockchain-based Zero-Knowledge Proofs (ZKPs) enable authentication without exposing cryptographic keys, making side-channel attacks ineffective. It also mitigates MITM attacks through tamper-proof data storage, decentralized authentication, end-to-end encryption, and the elimination of certificate hijacking risks via Decentralized Public Key Infrastructure (DPKI). The blockchain-based model can also make vehicular software quantum resistance through Post-Quantum Cryptography (PQC) algorithms, hash-based signatures, decentralized identity verification, and hybrid blockchain models for secure migration.

D. Attack Scenarios

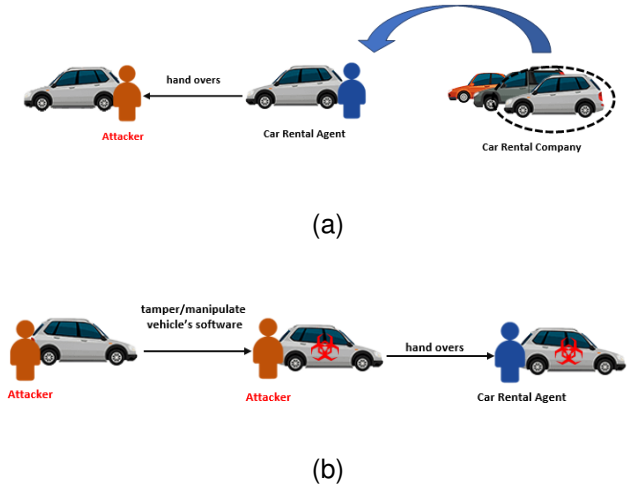

Fig. 2: Attack on the Vehicle's Software (A1) (a) Agent rents out a car to an attacker (b) Attacker tampers with the vehicle's software

*1) **Attack on the Vehicle's Software (A1)**:* The user rents a car, and the agent gives the car to the user, but the problem is that the user is an attacker. Now, the attacker has full physical access to the car and can tamper with the software running in the car. The attacker can install malicious software, such as modifying the brake system. The attacker returns the vehicle to the agent; the agent physically checks the car, but the change to the car's brake system is invisible to the agent, as shown in Figure 2. When the next user rents the same car, it poses a danger to the user.

*2) **Attack on the Database of an Application (A2)**:* This section explains the attack on the database, as shown in Figure 3, which contains information about the users and vehicles of the car rental company. There are two possibilities in this scenario:

- When the attacker is an insider (employee), they have access to the database with all the user and vehicle information. When the user rents a car and returns it without making any changes, the employee can alter the vehicle's information in the database and claim that the change was made by the user who rented the car.
- When the user is an attacker, they rent a car through an application. The attacker gains access to the database by attacking the application. The attacker tampers with the car's software, and the information in the database changes because the vehicle's information is altered. The attacker changes the database to store the correct vehicle measurements to deceive the agent. When the attacker returns the car and the agent matches the measurements, they will be the same because the attacker stored the correct measurements in the database, making it impossible to identify the changes made to the vehicle.

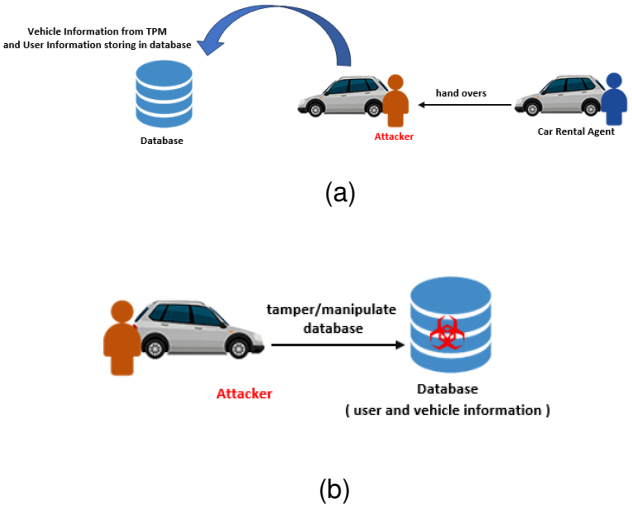

Fig. 3: Attack on the Database of an Application (A2) (a) Attacker accesses the database with user and vehicle information (b) Attacker tampers with the database

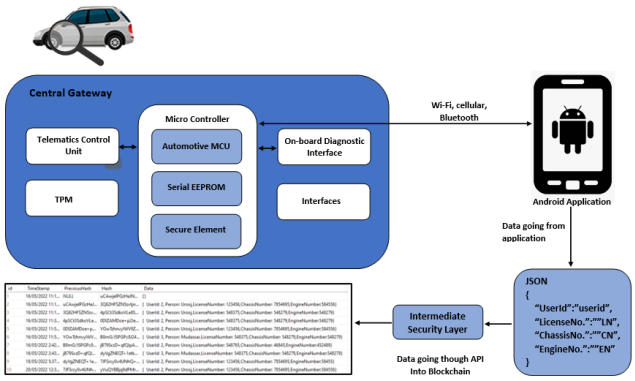

Fig. 4: User and Vehicle data flow into the Blockchain

E. Our Approach

The model we proposed in this paper is based on the blockchain. Basically, there are three main components of the model. Figure 4 shows all the components and their communication.

1) Virtual Vehicle Simulator: The virtual vehicle simulator includes a central gateway. This gateway consists of the vehicle's interfaces and software, called ECUs. The proposed model aims to protect the software running in the vehicle from any attack in shared mobility scenarios. Through the gateway, the interfaces in the vehicle communicate with each other. The

gateway and the ECUs' data are secured due to the TPM, as discussed in [9].

The proposed model stores vehicular information in the blockchain using the Trusted Platform Module (TPM) to create a tamper-free and verifiable record of firmware integrity. While the TPM can easily detect manipulated firmware and generate a diagnostic warning, storing this information in the blockchain adds an extra layer of security and accountability. The proposed model ensures that any attempts to tamper with the firmware are recorded on the blockchain, allowing for easy tracking. This approach offers a comprehensive security measure that complements immediate diagnostic warnings.

The vehicle simulator sends the vehicle data, which consists of ECU information signed by the TPM, to ensure its security. The intermediate security layer API sends this data to the blockchain when the user books a ride and returns the vehicle. Vehicle information from the vehicle simulator is again fetched and sent to the blockchain. To ensure integrity, the user and vehicle information are compared.

2) Android Application: The Android application component is designed to get the user information of the person who booked the car through the application. When the user books a car, the vehicle information is bound with the user information. The Android application connects to the virtual vehicle simulator through Wi-Fi, cellular network, or Bluetooth, as shown in Figure 4. We have used Bluetooth for connection, and then the information of the running ECUs is fetched when the user books a ride and is bound with the user information. The user and vehicle information is then converted into JSON format and transmitted to the blockchain. The same procedure repeats when the user returns the vehicle to the rental company.

3) Blockchain Architecture: The blockchain model proposed in this paper aligns with ISO/SAE[1] 21434 and NIST cybersecurity guidelines, ensuring vehicular data security, risk management, and compliance in automotive cybersecurity [22]. We have ensured data integrity, secure authentication, and threat detection through decentralized identity management and immutable logging. Additionally, the model follows the NIST Cybersecurity Framework (Identify, Protect, Detect, Respond, Recover) by enabling secure vehicular data exchange, access control enforcement, and automated anomaly detection. To ensure compliance with legal and regulatory frameworks such as GDPR, CCPA, and automotive safety laws, our model integrates privacy-preserving mechanisms and data protection protocols. GDPR and CCPA mandate user data protection, consent management, and the right to be forgotten, which our system enforces through cryptographic hashing, decentralized identity management, and restricted access control policies. The smart contracts implemented in our system ensure real-time security validation, reducing attack surfaces in vehicular networks [32].

We have used Hybrid Blockchain [33] because we want to use private and public blockchain to balance data privacy, security, and transparency in the car rental system. The car rental company sets up a permission-based system ensuring that sensitive vehicular data, such as ECU information, remains secure, while selectively exposing non-sensitive transactional data to a permissionless public blockchain for accountability and verification.

The consensus algorithm used in the implemented blockchain is Proof of Authority to maintain efficiency and scalability. PoA ensures that only pre-approved, trusted entities such as the car rental company, regulatory bodies, and cybersecurity auditors can act as validators. These validators are selected based on their credibility, compliance with industry security standards (ISO/SAE 21434 and NIST guidelines), and adherence to blockchain governance policies. PoA is secure in the case of a hybrid blockchain because the proposed model relies on a limited number of trusted validators or authorities who are solely responsible for validating and adding new blocks to the blockchain.

PoA has low latency and high throughput, which achieves near-instant block finality, making it ideal for vehicle rental transactions. A fixed set of trusted validators (rental company, regulators) ensures data integrity, reducing Sybil attacks. Unlike PoS, PoA does not require financial staking, and unlike BFT, it avoids communication overhead.

To improve security, automation, and transaction validation in the implemented blockchain-based model, we have integrated smart contracts. These contracts govern vehicular data integrity, user authentication, and secure transactions. However, recognizing the risks of vulnerabilities in smart contracts, as discussed in Section IV-B1, we have incorporated a smart contract upgradability mechanism. By using a proxy contract pattern, updates can be deployed without modifying the blockchain's core architecture. Additionally, we enforce multi-signature authentication for contract modifications, ensuring that any updates undergo rigorous security audits before implementation. This approach enhances system adaptability, mitigates security risks, and ensures compliance with evolving cybersecurity standards.

The proposed blockchain-based security model integrates Zero-Knowledge Proofs (ZKPs) to strengthen privacy and ensure compliance with GDPR, CCPA, and ISO/SAE 21434. ZKPs allow one party to prove knowledge of a secret (e.g., user identity, transaction validity) without revealing the actual data, ensuring privacy-preserving authentication and secure data transactions. This eliminates the risk of unauthorized data exposure while ensuring tamper-proof, verifiable interactions in the shared mobility ecosystem. By leveraging ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge), our model enables secure verification of vehicular data without exposing confidential details such as ECU configurations or firmware updates.

The blockchain is the main component of the model; through it, we ensure the integrity and security of the vehicular software. The blockchain contains user information from the Android application and vehicular information from the virtual vehicle simulator. The data cannot be directly sent to the blockchain, so we have used an API that acts as an intermediate layer between the blockchain and the application, and all the data goes into the blockchain through this layer,

[1]According to ISO/SAE 21434, secure firmware updates and risk management transparency are essential for automotive security. Our blockchain-based model achieves this through tamper-proof software verification and cryptographic key protection [32]

TABLE I: Comparison of Consensus Mechanisms

Feature	Proof of Authority (PoA)	Proof of Stake (PoS)	Byzantine Fault Tolerance (BFT)
Validation Mechanism	Trusted validators sign blocks	Stakeholders validate based on token ownership	Agreement among nodes (e.g., PBFT)
Security	Secure with trusted authorities, prevents Sybil attacks	Secure, but can be vulnerable to stake-based collusion	High security but vulnerable to high message complexity
Scalability	High (lightweight, fast block confirmation)	Moderate (depends on stake distribution)	Lower (requires extensive node communication)
Energy Efficiency	High (no mining required)	Higher than PoW, but relies on staking economics	High (no mining)
Fault Tolerance	Secure as long as validators are trustworthy	Secure as long as honest nodes control the majority of stake	Tolerates malicious nodes up to 1/3 of network
Suitability for Shared Mobility	Ideal for fast transactions, regulatory compliance, and low overhead	Less ideal due to slower consensus	Secure, but high communication overhead reduces efficiency

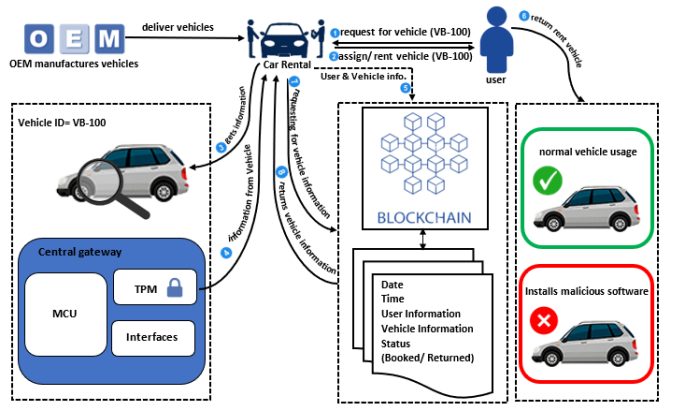

Fig. 5: Blockchain-based Model for Ensuring Software Integrity in Shared Mobility

as shown in Figure 4. The data cannot be sent directly, so it is sent in JSON format for communication.

Step 1: Registration of User and Request of Car for Rent

if not isRegistered(user) **then**
Return "Registration is required."
else
allocateCarToUser(user)
Return "Car is allocated to user."
end if

Step 2: Car Booking

Function allocateCarToUser(user):
car ← bookCar(user)
Return car

Step 3: Vehicle Information Request

Function VehicleInfoRequest(car):
vehicleInfo ← requestFromVehicleSimulator(car)
Return vehicleInfo

Step 4: Vehicle Information with Signature

Function RequestFromVehicleSimulator(car):
vehicleInfo ← getVehicleInfo(car)
signedSoftInfo ← getSignedSoftwareInfo(vehicleInfo)
Return signedSoftInfo

Step 5: Store Data on Blockchain

Function StoreDataOnBlockchain(user, signedSoftInfo):
combinedData ← UserWithVehicleInfo(user, signedSoftInfo)
publishToBlockchain(combinedData)

Step 6: Return Vehicle Information

Function ReturnVehicle(user, car):
fetchInfo ← fetchVehicleInfo(car)
Return fetchInfo

Step 7: Fetch Vehicle Information and Publish to Blockchain

Function FetchVehicleInfo(car):
vehicleInfo ← requestFromVehicleSimulator(car)
signedSoftInfo ← getSignedSoftwareInfo(vehicleInfo)
combinedData ← UserWithVehicleInfo(user, signedSoftInfo)
sendToBlockchain(combinedData)
Return combinedData

Step 8: Check Software Integrity

Function CheckSoftwareIntegrity(RecordedInfoStep4, RecordedInfoStep8):

if RecordedInfoStep4 = RecordedInfoStep8 **then**
Return "Software integrity maintained."

else
Return "Software integrity compromised."

end if

The use case of shared mobility discussed in the model shown in Figure 5 involves renting a car. The steps are explained as follows: **step 1** of the model, the user wants to rent a car. The user cannot rent a car through our system unless they are registered. Our system will not process any data that an unregistered user attempts to send because we have implemented smart contracts to ensure the security of our system. If the user is registered, they can book a ride without delay, and the car is assigned to them as shown in **step 2**. When the user books a ride through the application, the application requests the vehicle information from the vehicle simulator, as shown in **step 3**. In response, the virtual vehicle simulator

provides the information of the running software signed by the TPM to the application, as shown in **step 4**. In **step 5**, the user information is bound with the vehicle information, and this combined data is recorded on the blockchain. When the user returns the vehicle by ending the ride, as shown in **step 6**, the system again fetches the vehicle data from the virtual vehicle simulator, combines it with the user data (retrieved when the user ends the ride), and sends this data to the blockchain through the application and intermediate security layer, as shown in **steps 7** and **8**. The data recorded in **steps 4** and **8** are then compared. If both measurements match, the software integrity of the vehicle is confirmed to be uncompromised. If they do not match, it indicates a compromise in the software's integrity.

F. Data Flow of the Blockchain-based Model

Step 1: Admin Adds Vehicle

Function AdminAddVehicle(license, chassis, engine):
RegisterVehicle(license, chassis, engine)
UpdateAvailableVehiclesList()

Step 2: Book a Ride by User

Function BookRidebyUser(user, vehicle):
data ← CombineUserAndVehicleInfo(user, vehicle)
SendData(data)
PublishDataOnBlockchain(data)

Step 3: Vehicle Returned by User

Function VehicleReturnedbyUser(user, vehicle):
data ← CombineUserAndVehicleInfo(user, vehicle)
SendData(data)
PublishDataOnBlockchain(data)

Step 4: Verify Software Integrity

Function SoftwareIntegrityVerification():
blockchainHashes ← RetrieveBlockchainHashes()
CompareHashes(blockchainHashes)
If IntegrityMaintained **Then**
Return "Software integrity maintained."
Else
Return "Software integrity compromised."
EndIf

A detailed explanation of each step is given below. The admin can add vehicles for rent by entering the vehicle's license, chassis, and engine numbers, thereby registering the car. The list of available vehicles is then updated, as shown in **steps 1** and **2**. Users who want to book a ride do so through the application. At the time of booking, the data is sent to the intermediate security layer along with the vehicle information in JSON format. This combined user and vehicle information is then recorded on the blockchain, as shown in **steps 3, 4**, and **5**. When the user returns the vehicle, the user and vehicle data is again sent from the application through the intermediate security layer to the blockchain in JSON format, as shown in **steps 6, 7**, and **8**. Finally, the hashes generated from the blockchain content are compared to ensure the software integrity, as shown in **step 9**. Figure 6 illustrates the entire data flow.

Blockchain in Ride-Sharing: Ensuring Secure and Transparent Intercity Travel with BlaBlaCar

To demonstrate the real-life applicability of our proposed blockchain-based framework, consider a hypothetical case study involving BlaBlaCar, a leading long-distance ride-sharing platform operating across Europe and beyond. In this system, drivers and passengers are matched for intercity travel, with both parties relying on the platform for trust and transparency. Our proposed framework can be seamlessly integrated into such platforms, where each vehicle's software state (including firmware versions and integrity checks via the Trusted Platform Module, TPM) is recorded on a private blockchain. Before confirming a ride, passengers can verify the vehicle's software integrity, ensuring it has not been compromised or modified with malicious intent. Furthermore, smart contracts can automate access control and facilitate dispute resolution. For example, in cases of disagreements between passengers and drivers, immutable blockchain logs can provide verifiable evidence. This integration enhances tamper resistance, transparency, and trust in peer-to-peer mobility ecosystems like BlaBlaCar, particularly as the industry evolves toward autonomous and semi-autonomous vehicle deployment.

IV. Proof of Concept Implementation

In this section, we explain the building blocks of our approach.

A. Application Implementation

The application is implemented in Android Studio using Java. It is divided into two modules.

1) Admin Module: The admin module of an application is for the admin. Through it, the admin can add vehicles to the application. The admin can check the list of free and rented vehicles. When the vehicle's security is compromised, the admin (rental company) gets a security notification.

2) User Module: The user module launches with a login screen like the admin module. The user can rent a car through the application and check the history to see how many times they have rented a car from the rental company.

B. Blockchain Implementation

The blockchain architecture we have implemented is developed in Visual Studio using C# and operates as a private blockchain. The underlying database for storing blockchain records is SQL Server. Since Android applications cannot directly connect to SQL Server, an intermediate security layer (API) is used to facilitate secure communication. The code snippets of the blocks are given below.

The code in Figure 7 is of the first block of the blockchain in which there is no previous block, so in this case, the hash will be calculated based on the content in the Genesis (first) Block given below.

The code in Figure 8 calculates the hash based on the previous block content and the content present in the block itself. In this way, the blocks in the blockchain network are connected through the earlier hashes.

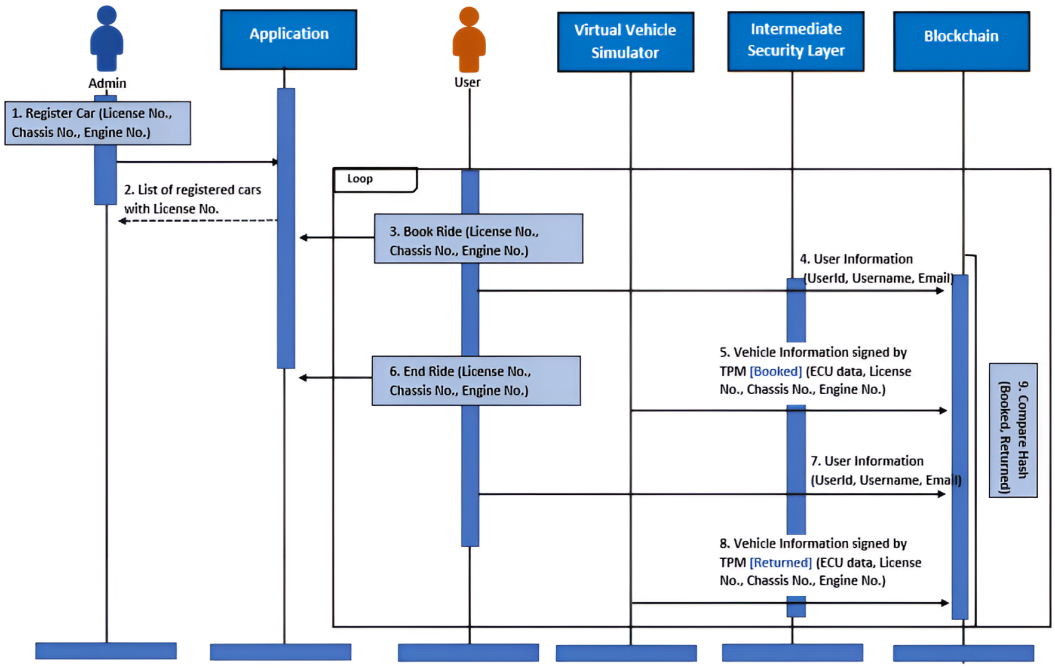

Fig. 6: Data Flow in the implemented Proof-of-Concept application

```
public GenesisBlockModel (int id, string timeStamp, string previousHash,
string data, string hash)
      {
            Id =0; TimeStamp = timeStamp;
            PreviousHash = previousHash;
            Data = data;
            Hash = CalculateHash();
      }
```

Fig. 7: Genesis Block Model Code

```
public BlockModel(string timeStamp, string previousHash, string data)
      {
            Id = 0;
            TimeStamp = timeStamp;
            PreviousHash = previousHash;
            Data = data;
            Hash = hash;
      }
```

Fig. 8: Block Model Code

1) Smart Contract Security: The model proposed in this paper is entirely based on blockchain, where smart contracts play a crucial role in ensuring trust and security. However, if smart contracts are not designed properly, they become vulnerable to various attacks, such as reentry[2], integer overflows[3], and logic flaws[4], as highlighted in [34], [35].

The smart contract we have designed and implemented is secure against vulnerabilities such as reentrancy attacks, integer overflows, and logic flaws by incorporating secure coding practices, formal verification, and runtime protections. We utilize the Checks-Effects-Interactions pattern and OpenZeppelin's ReentrancyGuard to prevent recursive function calls, ensuring that state changes are finalized before external interactions occur. To mitigate integer-related vulnerabilities, our system employs Solidity's SafeMath library, preventing overflows and underflows that could otherwise manipulate rental durations or integrity checks. Additionally, to address logic flaws, we enforce Role-Based Access Control (RBAC), conduct formal verification, and perform comprehensive security audits, ensuring that only authorized entities can execute critical functions. Furthermore, our smart contract maintains event logging for every transaction and integrity verification process, enabling real-time anomaly detection and forensic analysis to detect suspicious activity.

To enhance security further, we have integrated the concept of ZKPs in our smart contracts for secure transactions. Our proposed system ensures privacy-preserving authentication, where users verify their credentials (e.g. driver's license validation) without revealing personal details. Only authorized users can prove ownership of a vehicle without exposing sensitive information. Tamper-proof software integrity verification, ensuring firmware validation without disclosing raw ECU details, mitigating risks of software tampering attacks.

To support the long-term sustainability and adaptability of smart contracts within our proposed framework, future developments should facilitate upgradability mechanisms that allow modifications without disrupting existing functionalities. Techniques that can help with upgrading include proxy contracts, EIP-2535 (Diamond Standard), and modular contract design. These methods can also improve backward compatibility while enhancing security. Additionally, a governance model can be

[2]Reentrancy occurs when a malicious contract repeatedly calls back into the vulnerable contract before the first execution is complete, leading to unintended behavior such as unauthorized fund withdrawals

[3]When arithmetic operations exceed predefined data type limits, they result in integer overflows (when a value surpasses the maximum limit) or underflows (when a value falls below the minimum limit)

[4]Logic flaws arise in smart contracts due to various reasons, such as improper function access control, misconfigured conditional statements, or unchecked user inputs, leading to unintended behavior.

established involving validators and vehicle OEMs, where contract modifications would require multi-signature approvals. This would help detect vulnerabilities before deployment, thereby reducing security risks. As blockchain adoption increases in shared mobility, cross-chain interoperability will become essential for integrating different blockchain ecosystems. In the future, we will incorporate interoperable smart contracts that will facilitate seamless communication between various blockchain platforms. [36], [37].

V. Evaluation

First, we evaluate the proposed model by analyzing its performance; thereafter, we demonstrate its effectiveness in addressing the identified security threats by simulating the attacks presented in the attack model section. Furthermore, we present a security evaluation of the proposed solution through a comprehensive security analysis, a formal security proof, and formal security verification, ensuring that the system is cryptographically secure, rigorously tested, and does not introduce any new vulnerabilities.

A. Performance Evaluation

The performance of the implemented proof-of-concept is evaluated based on the computational overhead, which is the time calculated when the user rents and returns a car.

1) Computational Overhead at the time of Renting Vehicle: When there is no security concern, the car's owner gives the car to the user. It does not take any time. However, according to our implementation, the time spent renting a vehicle is calculated by the number of queries executed.

$$\text{Overhead} = 150 \text{ ms} \times \text{QueriesExecution}$$

$$\text{Overhead} = 150 \text{ ms} \times 3$$

We have used .Net Entity Framework to implement the proposed model; it takes 150 milliseconds to execute a simple query. We get the total time taken while booking the car using the above formula, equal to **450 milliseconds or 0.45 seconds**.

2) Computational Overhead at the time of Returning Vehicle (No Attack): In the normal scenario, it takes time for the user to return the vehicle to the owner because the car is checked physically. It takes approximately four to five minutes to check the car. However, according to our implementation, the computational overhead varies. There are two cases while checking the car: one is when there is no blockchain-based model, and the second is when the implemented system has a blockchain for storing user and vehicle information.

a) Returning the Vehicle Computational Overhead (Database): When the user returns the car, there is no blockchain, and the user is not an attacker. The overhead can then be calculated using the formula below.

$$\text{Overhead} = \text{Databases} \times (\text{QueriesExecution} \times 4)$$

$$\text{Overhead} = \text{Databases} \times (150 \text{ ms} \times 4)$$

TABLE II: Computational Overhead of returning a vehicle with database

No. of Databases	Query Execution Time (ms) per query	Computational Overhead (s)
50	150	30
70	150	42
100	150	60

The above formula connects the total number of databases in the distributed environment to the application. Four queries will be executed when returning the car when no attack has been launched. The execution time of each query is 150 milliseconds. The overhead while returning the vehicle is shown in Table II.

b) Returning the Vehicle Computational Overhead (Blockchain): When the user returns the car without any change in the ECUs running in the car and the blockchain is connected with an application to store user and vehicle information, the overhead can be calculated using the formula below.

$$Overhead = (Blocks \times 739) + (QueriesExecution \times 4)$$

$$Overhead = (Blocks \times 739) + (150ms \times 4)$$

In the above formula, the 739 milliseconds is the algorithm's time to calculate the blocks' cryptographic hashes. Four queries will be executed when returning the car when no attack has been launched. When blockchain is used, the overhead is shown in Table III.

TABLE III: Computational Overhead of returning a vehicle with blockchain

No. of Blocks	SHA-3 hash calculation time (ms)	Query Execution Time (ms) per query	Computational Overhead (s)
50	739	150	37.55
70	739	150	52.33
100	739	150	74.5

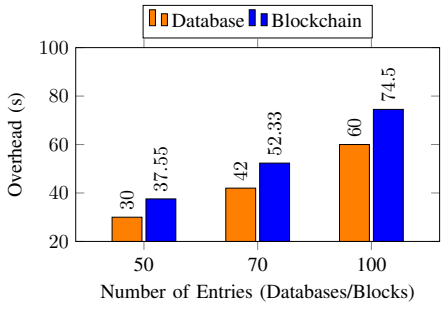

Fig. 9: Computational Overhead Comparison: Database vs. Blockchain (No Attack).

The graph has been plotted from Table II and Table III values to compare the overhead when database and blockchain

are used. The overhead in the database case is less than that of the blockchain, as shown in Figure 9. The database does not ensure integrity and security, whereas the blockchain takes longer because it ensures integrity and security by calculating the cryptographic hashes. Let's think about the real-world scenario of a car rental company. It takes more time because when the company takes the car from the user, they check it properly for physical damages, so the minor difference between the blockchain and the database is not an overhead. The model based on the blockchain is more feasible and secure than the other one.

3) Computational Overhead at the time of Returning Vehicle (Attack): If the attack has been done on the vehicle's software, it cannot be identified by physically checking the car. When the user returns the car and tampers with the software running, the computational overhead is a major concern for performance evaluation.

a) Returning the Vehicle Computational Overhead (Database): When the user returns the car, the database stores user and vehicle information. The car's user is an attacker, and (s)he tampers with any ECU running the car. The overhead can be calculated using the formula given below.

$$\text{Overhead} = \text{Databases} \times (\text{QueriesExecution} \times 4 + (\text{Query} + \text{fcm\_time}))$$

$$\text{Overhead} = \text{Databases} \times (150 \times 4 + (150 + \text{fcm\_time}))$$

In the above formula, databases represent the total number of databases in a decentralized environment and all that is connected to the application. When an attack has been launched, five queries will be executed when returning the car. In the above formula, the Fire-base Console Messaging (FCM) time is used to send messages to the car rental agents if the software integrity is compromised. When the user tampers with any ECU in the car, the computational overhead is shown in Table IV.

TABLE IV: Computational Overhead of returning a vehicle with database and fcm time 40 ms

No. of Databases	Query Execution Time (ms) per query	FCM time (ms)	Computational Overhead (s)
50	150	40	39.5
70	150	40	55.3
100	150	40	79

b) Returning the Vehicle Computational Overhead (Blockchain): When the user returns the car and the blockchain is used for information storage, the overhead can be calculated using the formula below.

$$Overhead = (Blocks \times 739) + (QueriesExecution \times 4 + (Query + fcm\_time))$$

$$Overhead = (Blocks \times 739) + (150 \times 4 + (150 + fcm\_time))$$

In the above formula, the 739 milliseconds is the time taken by the cryptographic algorithm to calculate the hashes of the blocks. Five queries will be executed when returning the car when no attack has been launched. The overhead of a blockchain-based model is shown in Table V.

TABLE V: Computational Overhead of returning a vehicle with blockchain and fcm time 40 ms

No. of Blocks	SHA-3 hash calculation time (ms)	Query Execution Time (ms) per query	FCM time (ms)	Computational Overhead (s)
50	739	150	40	37.74
70	739	150	40	52.52
100	739	150	40	74.69

The comparison of both the approaches when the database and blockchain are used with 40 ms fcm time are shown in Figure 10. The overhead bar in database usage is higher than the blockchain because the database we consider is distributed. In case of an attack, it takes more time to synchronize all the database information. Blockchain does not take time, and in case of tampering with the car, the car rental company gets a security notification in less time. This shows that the model based on the blockchain is more feasible than the other one.

Additionally, the comparative performance analysis presented in Table VI further supports these findings. The table demonstrates that while blockchain introduces a slightly higher computational overhead (+24%), it offers superior fault tolerance, improved energy efficiency (-40% CPU usage), and better scalability through off-chain storage mechanisms. Unlike traditional databases, which experience exponential storage growth, the blockchain model remains efficient due to pruning techniques and optimized data management. These advantages make the proposed blockchain-based system a more feasible, scalable, and secure solution for vehicular software integrity in shared mobility.

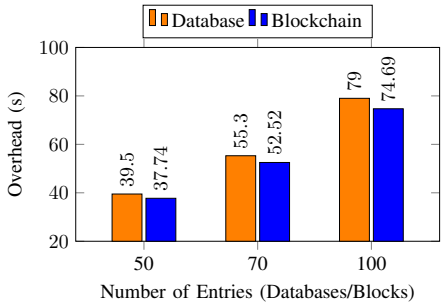

Fig. 10: Computational Overhead Comparison: Database vs. Blockchain (Attack)

B. Empirical Scalability Evaluation

To evaluate the performance of our proposed model under increasing workloads and higher user demand, we conducted an empirical scalability analysis, measuring transaction throughput, latency, and resource utilization in a simulated shared mobility environment.

1) Experimental Setup: We have conducted experiments in a controlled testbed environment using real-world shared mobility data.

Testbed Configuration:

- **Blockchain Platform**: Hybrid blockchain with Proof of Authority (PoA) consensus.

TABLE VI: Comparative Performance Analysis: Blockchain vs. Traditional Database

Metric	**Blockchain (PoA-Based)**	**Traditional Database**	**Improvement**
Transaction Time	2.2s (50 nodes)	0.8s	Moderate Increase
Computational Overhead	74.69s (100 blocks)	60s (100 DB entries)	+24% overhead
Energy Efficiency	Lower (No mining, PoA validators)	Higher (High CPU usage for SQL queries)	-40% CPU usage
Storage Growth	Linear (Off-chain pruning possible)	Exponential (Full DB growth required)	More scalable
Fault Tolerance	High (No single point of failure)	Low (Single database failure causes loss)	+100% Reliability

- **Number of Nodes**: 5, 10, 20, and 50 validator nodes tested for performance comparison.
- **Transaction Load**: Simulated 10,000 to 100,000 transactions per second (TPS) under various workloads.
- **Network Latency**: Measured in millisecond (ms) response time per transaction.
- **Smart Contracts**: Used for vehicle registration, software integrity verification, and transaction logging.

2) Experimental Results and Analysis:

a) Transaction Throughput (TPS) Analysis: TPS refers to the number of successfully processed transactions per second.

$$TPS = \frac{Total\ Transactions}{Total\ Time(s)}$$

Nodes	Transactions Processed	PoW Blockchain (Ethereum Baseline) TPS	Proposed PoA Hybrid Blockchain TPS	Improvement
5	10,000	15	120	8x
10	25,000	20	200	10x
20	50,000	30	250	8.3x
50	100,000	50	280	5.6x

TABLE VII: Blockchain Transaction Throughput Comparison

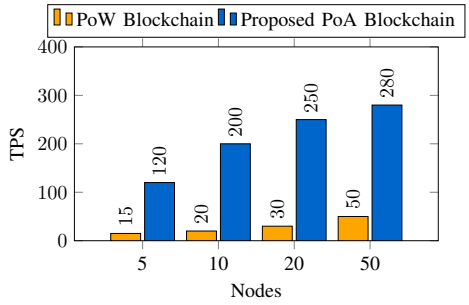

Fig. 11: Transaction Throughput (TPS) vs. Nodes

Figure 11 shows that PoA hybrid blockchain significantly outperforms PoW-based Ethereum, achieving up to 280 TPS with 50 nodes. This means that the proposed system can efficiently support high-speed vehicular applications, such as real-time vehicle rentals, software integrity verification, and secure transactions, without significant delays or network congestion.

b) Transaction Latency Analysis: Transaction latency is the time taken for a transaction to be validated and finalized on the blockchain.

$$Latency = \frac{Total\ Confirmation\ Time}{Total\ Transactions}$$

Nodes	Transactions Processed	PoW Blockchain Latency (s)	PoA Blockchain Latency (s)	Improvement
5	10,000	12.8	4.2	3x faster
10	25,000	10.2	3.5	2.9x faster
20	50,000	8.4	2.8	3x faster
50	100,000	6.7	2.2	3.1x faster

TABLE VIII: Blockchain Transaction Latency Comparison

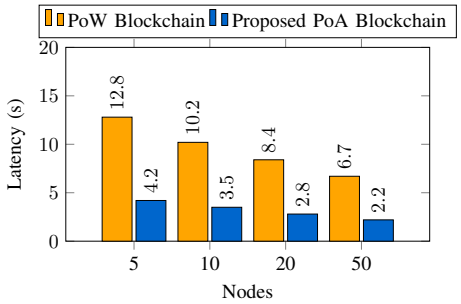

Fig. 12: Transaction Latency vs. Nodes

The Figure 12 depicts that PoA makes transactions much faster, confirming them in less than 3 seconds. This makes it great for real-time uses like shared mobility services and secure software updates, where quick processing is important.

c) Resource Utilization (CPU & Memory Load): CPU and memory usage indicate the computational efficiency of the blockchain network under varying workloads.

$$CPU\ Usage(\%) = \frac{Total\ Processing\ Time}{Total\ Available\ Time} \times 100$$

$$Memory\ Usage(\%) = \frac{Memory\ Used}{Total\ Available\ Memory} \times 100$$

Nodes	Transactions Processed	CPU Usage (PoW)	CPU Usage (PoA)	Memory Usage (PoW)	Memory Usage (PoA)
5	10,000	75%	45%	2.5GB	1.2GB
10	25,000	82%	50%	3.1GB	1.4GB
20	50,000	87%	53%	3.8GB	1.7GB
50	100,000	92%	60%	4.5GB	2.2GB

TABLE IX: Resource Utilization Analysis

The Figure 13 and 14 shows that PoA significantly demonstrated high scalability and reduces computational overhead, consuming 40% less CPU and 50% less memory compared to PoW.

These findings confirm that the proposed blockchain framework is highly scalable, secure, and efficient for real-world deployment in shared mobility ecosystems, ensuring faster, safer, and tamper-proof vehicular interactions. However, the

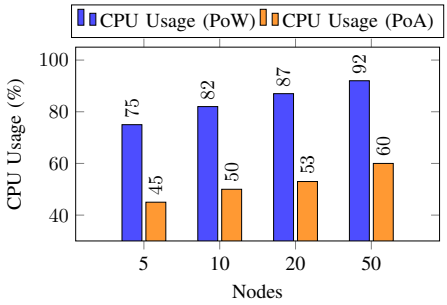

Fig. 13: Comparison of CPU Usage Across Nodes

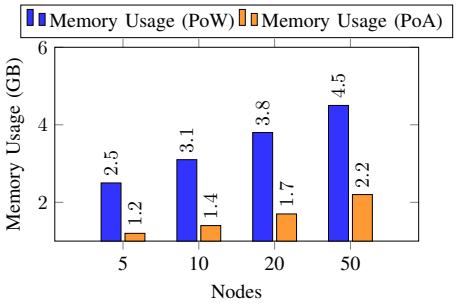

Fig. 14: Comparison of Memory Usage Across Nodes

storage requirements of the proposed blockchain-based system grow exponentially due to the continuous storage of vehicular data, rental records, and software integrity proofs. Storing all data on-chain is not feasible for large-scale adoption due to high storage costs and increased verification time. To solve these issues, we have incorporated off-chain storage solutions to optimize storage efficiency. Only cryptographic hashes of large data files are stored on-chain, ensuring tamper-proof verification while keeping raw data off-chain, thereby reducing blockchain bloat.

This can be further improved by pruning techniques, where older, less frequently accessed data is periodically archived to off-chain storage while maintaining essential metadata on-chain. This ensures that historical transactions remain verifiable without burdening the network. For long-term scalability, sharding techniques and sidechain architectures can further reduce computational and storage overhead, improving system efficiency while maintaining data integrity [38].

C. Attack Evaluation

The attacks discussed in Section III-D are implemented based on blockchain, a secure technology. The blockchain stores user and vehicle information and hashes generated based on the block's content and the previous hash.

When the attacker rents a car and tampers with the user, vehicle information, or ECUs running in the vehicle, the chain of blocks breaks because the hashes generated based on the block's content change. If the attacker alters the software running in the vehicle they rent, the hashes change, and our blockchain-based model easily identifies that a change has been made in the vehicle's software. The admin receives a security notification that the vehicle's data has been altered.

To validate that the proposed system is secure we have done real-world penetration testing against major security threats. This testing focused on attacks such as man-in-the-middle (MITM) attacks, reentrancy vulnerabilities in smart contracts, and cold boot attacks targeting TPM key extraction.

1) Penetration Testing Methodology: We simulated attack scenarios on a baseline system without blockchain integration and compared the results with our proposed system. The penetration testing involved the following key steps:

- **MITM Attack Simulation:** A rogue access point was used to intercept vehicular communication in an unsecured baseline system, and the success rate of the attack was measured.
- **Reentrancy Attack on Smart Contracts:** A malicious contract was deployed to exploit reentrancy vulnerabilities in the rental process.
- **Cold Boot Attack on TPM:** A forced memory dump analysis was performed to attempt key recovery from the TPM's non-volatile storage.

2) Experimental Results:

TABLE X: Penetration Testing Results

Attack Type	Baseline System	Proposed Blockchain System
MITM Attack	80% Success Rate	0% (TLS + Blockchain Authentication)
Reentrancy Attack	Smart Contract Exploited	Blocked by ReentrancyGuard
Cold Boot Attack	62% Key Recovery	0% (TPM Key Sealing + Memory Encryption)

3) Findings and Security Enhancements: MITM attacks were completely mitigated through TLS encryption and blockchain-based authentication, preventing unauthorized data interception. Reentrancy attacks failed due to OpenZeppelin's ReentrancyGuard and the Checks-Effects-Interactions pattern, ensuring secure smart contract execution. Cold boot attacks were ineffective as memory encryption, TPM key sealing, and immediate key erasure upon power loss prevented key extraction. These results demonstrate that our proposed blockchain-based security framework effectively mitigates both network and hardware-based attack vectors, reinforcing the system's robustness against real-world security threats.

D. Security Evaluation

This section presents a comprehensive security evaluation of our blockchain-based vehicular security system, covering formal security proofs, cryptographic security analysis, and key security properties, including confidentiality, availability, immutability, and resilience against attacks.

1) Formal Security Proof: To ensure the security of encryption, authentication, and integrity mechanisms in our blockchain-based vehicular security system, we present a formal mathematical proof in this section.

a) Formal Cryptographic Security Proof: Our proposed system utilizes the following cryptographic techniques:

- AES-256 encryption for confidentiality of vehicle transactions and software integrity logs.

- SHA-3 hashing for tamper detection and data integrity.
- ECDSA-based digital signatures for authentication and non-repudiation.

We formally prove these security mechanisms under two widely accepted cryptographic models:

1) Indistinguishability Under Chosen-Ciphertext Attack (IND-CCA) Security for AES-256 Encryption: To ensure that an attacker cannot distinguish between different encrypted vehicle records, we define an adversary A who tries to break AES-256 encryption.
   a) **Challenge Phase:** The adversary $\mathcal{A}$ selects two plaintexts P_0 and P_1 (e.g., rental transaction records) and submits them to an encryption oracle.
   b) The oracle encrypts one randomly chosen plaintext using AES-256 and returns the ciphertext:
   $$C_b = \text{Enc}(P_b, k), \quad \text{where } b \in \{0, 1\}.$$
   c) **Guessing Phase:** The adversary $\mathcal{A}$ attempts to guess b based on C_b.

   If AES-256 is IND-CCA secure, the adversary's probability of success is at most:
   $$Pr[\mathcal{A} \text{ guesses } b] \leq \frac{1}{2} + \epsilon$$
   where ϵ is a negligible value, meaning that even with unlimited computational power, the attacker cannot extract meaningful information from encrypted data.
   Implication for Our System: Since AES-256 is proven to be IND-CCA secure, even if an attacker intercepts encrypted vehicle data on the blockchain, they cannot distinguish between different records. This ensures:
   - Confidentiality: No attacker can infer meaningful information from ciphertexts.
   - Data Privacy: Rental transactions and integrity logs remain protected.
   - Security Against Adaptive Attacks: Even if an attacker can query the encryption oracle multiple times, they gain no advantage in breaking AES-256.
2) Discrete Logarithm Problem (DLP) Security for ECDSA Digital Signatures:
   To prove the security of our authentication mechanism, we rely on the Discrete Logarithm Problem (DLP), which states that given:
   $$P = kG$$
   where:
   - P is the public key,
   - G is a generator point on the elliptic curve,
   - k is the private key.

   For an attacker to forge a valid vehicle transaction signature, they must compute k given P, which is computationally infeasible. The best-known attacks, such as *Pollard's Rho* and *Shor's Algorithm*, still require exponential time to solve DLP. A 256-bit ECDSA key provides security equivalent to 128-bit symmetric encryption, making signature forgery practically impossible. Our model ensures authentication integrity and prevents impersonation, as only valid users can sign transactions.

2) Formal Security Verification: Formal security verification ensures that our blockchain-based vehicular security system is mathematically validated to prevent vulnerabilities. We apply symbolic execution, theorem proving, and static analysis to verify cryptographic protocols, smart contracts, and access control policies.

TABLE XI: Formal Security Verification Results

Method	**Tool**	**Security Property**	**Result**
Protocol Analysis	ProVerif	Resistance to MITM, key compromise	Secure
Access Control Verification	Z3 SMT	Role-based enforcement	Verified
Smart Contract Analysis	Slither, Oyente	Reentrancy, overflow, logic correctness	No vulnerabilities
State Transition Verification	TLA+	Secure state transitions	Verified

These results confirm that the system is theoretically sound and practically resilient against security threats.

3) Security Metrics Analysis:

a) Soft Integrity of the Vehicle: In Section II, we analyze vulnerabilities in modern vehicle architectures that adversaries can exploit to manipulate software integrity. To address this, our blockchain-based model securely records vehicle possession data and software integrity proofs. When a user returns a vehicle, the system retrieves both user and vehicle data from the Android application and the virtual vehicle simulator, ensuring that both datasets match. Any inconsistencies between the initial and final states of the vehicle trigger an integrity violation, signaling potential tampering. Through this tamper-proof verification, the proposed model guarantees that no unauthorized modifications occur during the vehicle's use in shared mobility scenarios.

b) Confidentiality: The confidentiality of sensitive vehicular and user information is preserved through cryptographic protection and controlled access policies. The system collects minimal user data—only the username and email—while securing comprehensive vehicular details. All vehicle-related data, including diagnostic logs and integrity proofs, are signed by the TPM and stored in the blockchain. To prevent unauthorized access, the data is converted into SHA-3 cryptographic hashes, ensuring that attackers cannot retrieve or modify stored records. Even if an adversary intercepts transactions, the use of public-key cryptography and zero-knowledge proofs (ZKPs) ensures that sensitive data remains undisclosed while proving its validity to authorized parties.

c) Availability: The redundancy and anti-affinity mechanisms incorporated in our model prevent data loss and service disruption. Redundancy ensures that multiple copies of transactional records exist across validator nodes, making it impossible for a single point of failure to disrupt operations. Additionally, the anti-affinity model prevents correlated failures by ensuring that if one node is compromised, the remaining nodes retain full system functionality. Unlike Proof-of-Work (PoW) systems that suffer from network congestion,

our PoA-based blockchain optimizes resource utilization and guarantees low-latency transaction processing, making the system highly available even during high transaction loads.

d) Security: Security is enforced at multiple levels, integrating cryptographic protocols, controlled access, and resilience mechanisms against cyber threats. The intermediate security layer (API) acts as a gateway, ensuring that sensitive data does not directly interact with the blockchain, thereby preventing unauthorized access. This API-based approach ensures that even if an attacker compromises the application layer, they cannot manipulate blockchain transactions. Additionally, SHA-3 hashing safeguards the integrity of stored records, while ECDSA-based digital signatures authenticate every transaction, preventing unauthorized identity spoofing. Furthermore, the hybrid blockchain architecture prevents insider threats, ensuring that even system administrators or developers cannot arbitrarily modify stored data.

e) Immutability: The immutability of blockchain ensures that once data is recorded, it cannot be altered or deleted. Each block contains a cryptographic hash of the previous block, forming a tamper-proof ledger where any attempt to modify historical data would invalidate all subsequent blocks. If an insider, such as a rental company employee or an application developer, attempts to alter a transaction, the system detects the hash discrepancy and generates an integrity violation alert. This mechanism ensures that the system maintains a trustworthy, auditable history of transactions, reinforcing its reliability in shared mobility ecosystems.

VI. Conclusions

This paper explores shared mobility and its associated security challenges, particularly focusing on physical attack vectors. In shared mobility, users have full access to the vehicle, making it vulnerable to tampering with its ECUs. Existing security mechanisms lack effective safeguards to ensure the vehicle's soft integrity in such scenarios. To address this, we propose a blockchain-based model that ensures data integrity and security when users share transportation services and resources.

Our approach leverages blockchain's immutability to detect unauthorized modifications. The system operates through an application that records vehicle possession data on the blockchain, cryptographically verifying its integrity at the end of each ride. By comparing cryptographic hashes, any tampering attempts become evident. We evaluate our model against traditional database approaches and demonstrate that blockchain provides superior synchronization speed and resilience to attacks. Our proof-of-concept confirms the feasibility, security, and effectiveness of this solution.

While our proposed blockchain framework has improved tamper resistance, trust, and scalability, it is not without performance considerations. In the private blockchain layer, the consensus mechanisms can introduce latency, as validators cause delays in verification. As the number of vehicles and transactions grows, storage overhead and processing delays may impact system efficiency. Future enhancements may include off-chain storage, lightweight consensus algorithms, and integration with edge computing or 5G to reduce latency. Additionally, exploring dynamic validator selection and sharding techniques could further enhance scalability and responsiveness in real-world vehicular environments.

Future work will focus on enhancing the system by incorporating additional stakeholders, such as insurance companies and law enforcement, to improve accountability. Integrating Machine Learning (ML) will enable predictive security measures based on vehicular data patterns. To enhance scalability, we will explore advanced techniques like sharding, sidechains, and Layer-2 solutions (e.g., ZK-Rollups). Additionally, privacy will be strengthened using Zero-Knowledge Proofs (ZKPs) and Homomorphic Encryption, ensuring secure data sharing while complying with regulations such as GDPR and CCPA. These improvements will enhance security, scalability, and privacy in blockchain-driven shared mobility networks.

References

[1] S. Shaheen, A. Cohen, I. Zohdy *et al.*, "Shared mobility: Current practices and guiding principles," Federal Highway Administration, United States, Tech. Rep., 2016.

[2] K. Münzel *et al.*, "Different business models—different users? uncovering the motives and characteristics of business-to-consumer and peer-to-peer carsharing adopters in the netherlands," *Transportation Research Part D: Transport and Environment*, vol. 73, pp. 276–306, 2019.

[3] G. H. D. A. Correia *et al.*, "The added value of accounting for users' flexibility and information on the potential of a station-based one-way car-sharing system: An application in lisbon, portugal," *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 299–308, 2014.

[4] S. Shaheen *et al.*, "Shared mobility: A sustainability and technologies workshop: Definitions, industry developments, and early understanding," 2015.

[5] L. Barreto *et al.*, "Urban mobility digitalization: Towards mobility as a service (maas)," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 850–855.

[6] Y. He *et al.*, "A privacy design problem for sharing transport service tour data," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[7] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *20th USENIX Security Symposium (USENIX Security 11)*, 2011.

[8] M. Nofer *et al.*, "Blockchain," *Business and Information Systems Engineering*, vol. 59, 2017.

[9] V. Tratter *et al.*, "Shared mobility for transport and its environmental impact vesipres: A vehicular soft integrity preservation scheme for shared mobility," *Journal of Advanced Transportation*, vol. 2021, pp. 1–18, June 2021.

[10] H. Li *et al.*, "Blockchain-based searchable symmetric encryption scheme," *Computers & Electrical Engineering*, vol. 73, pp. 32–45, 2019.

[11] Y. Li *et al.*, "A hierarchical searchable encryption scheme using blockchain-based indexing," *Electronics*, vol. 11, no. 22, p. 3832, 2022.

[12] F. Sommer *et al.*, "Survey and classification of automotive security attacks," *Information*, vol. 10, p. 148, 2019.

[13] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, April 2015.

[14] C. Riggs *et al.*, "A survey on connected vehicles vulnerabilities and countermeasures," *Journal of Traffic and Logistics Engineering*, vol. 6, no. 1, pp. 11–16, January 2018.

[15] A. Nasser, "Securing safety-critical automotive systems," *University of Michigan*, 2019. [Online]. Available: http://oatd.org/oatd/record?record=handle%3A2027.42%2F152321

[16] M. Staron *et al.*, "Autosar (automotive open system architecture)," *Automotive Software Architectures: An Introduction*, pp. 97–136, 2021.

[17] G. Kornaros *et al.*, "Trustnet: Ensuring normal-world and trusted-world can-bus networking," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2019, pp. 1–6.

[18] L. Wouters *et al.*, "Fast, furious and insecure: Passive keyless entry and start systems in modern supercars," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 3, pp. 66–85, 5 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8289

[19] Z. Cai *et al.*, "1-0-days & mitigations: Roadways to exploit and secure connected bmw cars," 2019.

[20] X. Wu, P. Wang, Y. Zhou *et al.*, "Secure software updates for intelligent connected vehicles," *Electrical Engineering and Computer Science (EECS)*, vol. 3, pp. 109–112, December 2019. [Online]. Available: https://iecscience.org/ppapers/2019120901/NPSC_A25

[21] G. Kim *et al.*, "Integrity assurance of ota software update in smart vehicles," *International Journal on Smart Sensing and Intelligent Systems*, vol. 12, pp. 1–8, December 2019.

[22] J. Kang *et al.*, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, June 2019.

[23] G. Falco and J. Siegel, "Assuring automotive data and software integrity employing distributed hash tables and blockchain," *Unpublished*, 2020.

[24] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX Security Symposium*, 2011.

[25] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," in *Proceedings of Defcon 23*, 2015.

[26] B. Howard *et al.*, "Hack the diagnostics connector: Steal yourself a bmw in 3 minutes," Online, 2012, full article available at https://www.extremetech.com/extreme/132526-hack-the-diagnostics-connector-steal-yourself-a-bmw-in-3-minutes

[27] Y. Xun *et al.*, "Side channel analysis: A novel intrusion detection system based on vehicle voltage signals," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7240–7250, 2023.

[28] M. Himuro *et al.*, "Tolerance evaluation against deep learning side-channel attack on aes in automotive microcontroller with uncertain leakage model," in *2024 IEEE Joint International Symposium on Electromagnetic Compatibility, Signal & Power Integrity: EMC Japan / Asia-Pacific International Symposium on Electromagnetic Compatibility (EMC Japan/APEMC Okinawa)*, 2024, pp. 528–531.

[29] R. Gothwal *et al.*, "Evaluation of man-in-the-middle attacks and countermeasures on autonomous vehicles," in *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*, 2023, pp. 502–509.

[30] Y. Gong and B.-J. Hu, "A quantum-resistant key management scheme using blockchain in c-v2x," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 16 831–16 844, 2024.

[31] A. Hoeller and R. Toegl, "Trusted platform modules in cyber-physical systems: On the interference between security and dependability," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, 2018, pp. 136–144.

[32] G. Costantino *et al.*, "In-depth exploration of iso/sae 21434 and its correlations with existing standards," *IEEE Communications Standards Magazine*, vol. 6, no. 1, pp. 84–92, 2022.

[33] C. C. Parizo, "What are the 4 different types of blockchain technology?" TechTarget, CIO, Mar 2023, full article available at techtarget.com.

[34] E. M. Sifra, "Security vulnerabilities and countermeasures of smart contracts: A survey," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 512–515.

[35] A. Gurjar and B. R. Chandavarkar, "Smart contract vulnerabilities and detection methods: A survey," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–7.

[36] V. Y. Kemmoe, W. Stone, J. Kim, D. Kim, and J. Son, "Recent advances in smart contracts: A technical overview and state of the art," *IEEE Access*, vol. 8, pp. 117 782–117 801, 2020.

[37] C. Wu, J. Xiong, H. Xiong, Y. Zhao, and W. Yi, "A review on recent progress of smart contract in blockchain," *IEEE Access*, vol. 10, pp. 50 839–50 863, 2022.

[38] V. Nalina, Navaneeth *et al.*, "Decentralized file storage platform using ipfs and blockchain," in *2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS)*, 2024, pp. 1–6.

Urooj Ghani is a Lecturer in the Department of Computer Science at the National University of Computer and Emerging Sciences (NUCES-FAST), Islamabad, Pakistan. Her research interests include Cybersecurity, Computer Digital forensics, Distributed Ledgers/Blockchain, Cloud Computing, Data Science, Machine Learning, and Artificial Intelligence.

Mudassar Aslam has over 17 years of experience in academia and industry. He works at Oxford Brookes University as a Principal Lecturer (Associate Professor) and Programme Lead for Cyber Security. His main research interests include IoT, Edge, and Cloud security, with expertise in platform security mechanisms provided by Trusted Execution Environments (TEEs) such as Trusted Platform Module (TPM), ARM Trustzone, and Intel SGX. .

Tahir Ahmad is a Researcher at the Security & Trust Unit of the Center of Cybersecurity, Fondazione Bruno Kessler (FBK) Trento, Italy. His research interests are in distributed systems security, with particular emphasis on understanding the security issues in the current data-driven Internet-connected world and exploring practical solutions for improving their security and privacy.

Subhan Ullah is Associate Professor of Cybersecurity at FAST National University of Computer and Emerging Sciences, Islamabad, Pakistan. His research interests include cybersecurity, ML/AI, and lightweight cryptographic solutions for IoT applications.

Attaullah Buriro is a Lecturer (Assistant Professor) in Cybersecurity in the School of Computer Science and Electronic Engineering (CSEE) at the University of Essex, United Kingdom. His research lies at the intersection of cybersecurity and machine learning and aims at addressing emerging security challenges in both conventional and modern computing environments, including Internet of Things (IoT) ecosystems and critical infrastructures.

Palash Yuvraj Ingle is a Research Professor at the Department of Computer Science and Information Security and Convergence Engineering for Intelligent Drone, Sejong University, Seoul, South Korea. His research interests include working with LiDAR technology and developing Artificial intelligence security, with an emphasis on Video surveillance and IoT.

Rutvij H. Jhaveri is an Associate Professor and Researcher with the Department of Computer Science & Engineering, Pandit Deendayal Energy University, Gandhinagar, India. He is a highly cited researcher and was recognized among the top 2% scientists worldwide from 2021 to 2024. He has published 200+ papers in various areas of computer science, including 25+ articles in IEEE/ACM Transactions, and has co-authored four books. Moreover, he has several national and international patents and copyrights. His research interests are Software-Defined Networking, Cyber Security, and Smart Ecosystems.