

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Alshaeri, Abdulaziz, and Mohamed Younis. "Efficient Distributed Authentication for Intelligent Transportation Systems Using Mobile Devices." IEEE Transactions on Intelligent Transportation Systems, 2024, 1–16. <https://doi.org/10.1109/TITS.2024.3376517>.

<https://doi.org/10.1109/TITS.2024.3376517>.

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

Efficient Distributed Authentication for Intelligent Transportation Systems Using Mobile Devices

Abdulaziz Alshaeri¹, Graduate Student Member, IEEE, and Mohamed Younis¹, Fellow, IEEE

Abstract—Intelligent Transportation Systems (ITS) opt to improve safety and efficiency by internetworking vehicles, road infrastructure, pedestrians, etc. Given the ad-hoc connectivity and dynamic topology of such a network, robust authentication of member nodes is essential. The authentication process should also suit the resource constrained ITS nodes. This paper proposes an efficient approach for Distributed Authentication for ITS (DAITS). DAITS employs drivers' mobile devices to act as verifiers, and hence message authentication is provided in an as-a-service basis for the ITS nodes. Moreover, DAITS is a certificateless system, which deploys private smart contracts in a permissioned blockchain, for certifying nodes. Furthermore, the smart contracts store authentication tokens for the ITS nodes which ensure authentication between the ITS nodes and road infrastructure. DAITS relies on lightweight security primitives such as hash function, bitwise XOR, and Hash-based Message Authentication Code (HMAC). Extensive security analysis shows that DAITS can resist various security attacks. The simulation results demonstrate that DAITS is both resource-efficient and scalable, and outperforms competing schemes in terms of computation and communication overhead, and verification delay.

Index Terms—Intelligent transportation systems, authentication, vehicular communications, blockchain.

I. INTRODUCTION

ITS have emerged as an effective means for urban mobility and recently gained increasing attention from both academia and industry due to the drive for establishing smart cities. The US Department of Transportation defines ITS as the incorporation of wireless communication technologies and computational resources into vehicles and traffic infrastructure to increase efficiency and safety [1]. For instance, vehicles can share their speed, position, and planned turns with other road users to avoid collisions. Other useful information can be shared with road infrastructure, i.e., traffic lights, road signs, bridges, etc., to improve traffic flow which in turns reduce trip times, accident rates, traffic congestion, and air pollution [2], [3]. Interconnectivity of ITS components is attained by utilizing various wireless technologies such as Dedicated Short-Range Communication (DSRC), Cellular Vehicle-to-Everything (C-V2X), Long-Term-Evolution-Advance (LTE-A), and 5G [4].

Manuscript received 25 June 2022; revised 20 December 2022, 24 May 2023, 19 September 2023, and 23 November 2023; accepted 8 March 2024. The Associate Editor for this article was X. Cheng. (Corresponding author: Abdulaziz Alshaeri.)

The authors are with the Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA (e-mail: ZO40379@umbc.edu; younis@umbc.edu).

Digital Object Identifier 10.1109/TITS.2024.3376517

A. Challenge

ITS raises concerns related to node authentication, user privacy, and secure message dissemination to avoid making the entire system prone to cyberattacks. At the level of inter-vehicle communication, safety messages are subject to very low latency requirement and hence the adoption of the computationally heavy asymmetric cryptography techniques, e.g., public key infrastructure (PKI), is not favorable; yet lightweight symmetric cryptography-based techniques present their own security limitations. This makes devising an efficient authentication and secure message dissemination approach to be quite challenging. Non-repudiation is also an important property from a legal perspective since it enables establishing liability and supports forensics. Furthermore, preserving privacy while ensuring non-repudiation is another issue. Although safety message broadcasts ought to sustain the user anonymity, uniquely identifying the message source is crucial to ensure accountability. Realization of the Connected Autonomous Vehicles (CAV) concept [5] would further magnify the concern due to the potentially dramatic consequence of successful cyberattacks.

B. Technical Gap

While numerous studies have been published on addressing the aforementioned security issues, most approaches in the literature fall short in addressing the problem in a complete ITS system and rather have a narrower focus, i.e., vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), or vehicle-to-pedestrians (V2P). Moreover, most of the existing approaches offer centralized solutions by relying significantly on a central trusted authority (TA) and roadside units (RSUs) to achieve authentication and secure message dissemination which increases delay. Furthermore, they assume widespread deployment of RSUs; yet, such pervasive deployment is not guaranteed in the near future, and researchers should rather consider the practical scenario where scarcity of RSUs is often the norm [6]. In addition, these centralized approaches would suffer scalability issues when adopted in a complete ITS due to the heterogeneity of the involved players and the complexity and diversity of their interactions. While autonomous message verification mitigates a vehicle's dependency on others, it lacks sufficient onboard computational capability to verify many messages within a short time frame [7]. Finally, the last category of approaches pursues cooperative message authentication to distribute the heavy burden among vehicles. However, such a strategy is vulnerable to Sybil attacks [6].

C. Contribution

This paper fills the technical gap and promotes DAITS, a novel approach for a distributed, lightweight, and privacy-aware authentication and message verification in ITS. Specifically, our approach achieves the design goals as follows:

- To alleviate the heavy dependence on infrastructural support, i.e., TA and RSUs, and tackle the efficiency and scalability issues, DAITS leverages the availability of mobile devices, e.g., smartphones, and employs them as message authenticators.
- The authentication and verification of data origin and integrity are offered to the resource-constrained ITS components, e.g., vehicles' On-Board Units (OBUs), by mobile devices in an as-a-service fashion.
- Instead of the conventional batch or cooperative message verification, we introduce a novel delegation mechanism that eradicates the cost of repetitive message verification in vehicular communications.
- Since offloading the message authentication to RSU or TA could double the communication cost on the ITS nodes, DAITS reduces the overhead by only broadcasting negative verification of messages.
- DAITS is computationally efficient and relies on lightweight primitives such as bitwise XOR, hash function, and hash-based message authentication code (HMAC).
- To mitigate the shortcomings of symmetric cryptography in vehicular communications that raise security concerns such as the key exposure and the lack of non-repudiation, we assign a secret key to each node.
- To certify ITS nodes at the time of enrollment, we utilize Blockchain and smart contracts, which constitute a distributed, cost-efficient, and scalable way compared to PKI-based mechanisms such as digital certificates. Digital certificates need computationally heavy verification and impose overhead for handling and maintaining the certificate revocation lists (CRLs). DAITS employs private smart contracts that store the certification status of nodes in the system and hence eliminates the communication and computation overhead of CRLs.

D. Organization

The rest of the paper is organized as follows. We summarize the related work in Section II. Section III provides a brief background and describes the system's architecture, threat and communication models. The operational phases of DAITS are explained in Section IV. Section V presents the security analysis. We discuss the simulation results in Section VI and compare the DAITS' performance to various competing schemes. Finally, the paper is concluded in Section VII.

II. RELATED WORK

Published authentication approaches for vehicular communications can be classified based on the underlying cryptographic method into asymmetric-based and symmetric-based. For instance, Elliptic Curve Digital Signature Algorithm (ECDSA) [8] is a widely adopted scheme in the context of VANETs for providing authentication, non-repudiation, and

data integrity. However, due to the reliance on asymmetric cryptography, ECDSA requires more computation power and processing time compared to the symmetric cryptography-based approaches. Bae et al. [9] have analyzed the use of ECDSA in latency-critical applications and questioned its efficiency in handling a high volume of messages. Some other approaches have relied on symmetric cryptography to alleviate the shortcomings of asymmetric cryptography. For example, TESLA [10] is a well-known broadcast authentication protocol that is commonly used in the context of VANETs. TESLA relies on hash chains with delayed key disclosure for the authentication of the message sender. However, TESLA suffers from the lack of non-repudiation and the susceptibility to memory-based DoS attacks since the packets get buffered until the associated keys are disclosed [11].

Some approaches in the literature, such as [12], [13], and [14], have employed either tamper-proof devices (TPDs) or smart cards to store the authentication parameters. However, incorporating TPDs or smart card readers in the OBUs design would increase the cost and may hinder economic viability [7]; in addition, it would restrict the ability of nodes to participate in the ITS ecosystem [6]. Palaniswamy et al. [14] assume a smart card at the driver side and a card reader at the vehicle side, and employ a lightweight authentication and key exchange protocol for V2I and V2V. The protocol supports continuous authentication by employing a handoff capability in order to mitigate the roaming of the vehicles between RSUs. In addition to the cost concern about tamper-proof hardware, mentioned above, such a scheme raises other security issues since cards are susceptible to physical attacks, being stolen, or compromised. We also note that this scheme is not completely lightweight as it still relies on PKI for V2V key dissemination. To mitigate the cost concern, some approaches employ lightweight hardware primitives, e.g., Physical Unclonable Functions (PUFs), and pursue multi-factor authentication. For example, Alfadhly et al. [12] use a one-time secret key and a PUF. Due to limitations of using a single system key, regional keys are factored in by RSUs. Before broadcasting a beacon, the vehicle's PUF generates a dynamic broadcasting pseudo-identity, and a signature is then created using a one-way hash function. Upon receiving the beacon, the surrounding nodes verify the signature with the help of the regional key. However, this approach relies heavily on the TA as a vehicle moves from the range of RSU to another, which makes the TA a bottleneck and risk scalability.

When the underlying verification mechanism is considered, current literature can be classified into autonomous, cooperative, and delegated message verification. An autonomous mechanism implies that each vehicle independently verifies the received message, which could potentially exhaust the limited resources available for the ITS nodes given the message volume [7]. In a cooperative mechanism [15], [16], [17], [18], vehicles collectively apply the verification process where each vehicle contributes based on its available resources. Consequently, the message authentication delay diminishes since vehicles make their authentication decision based on the verification results received from other neighboring nodes. However, cooperative verification requires the engagement of many vehicles that are willing to participate;

moreover, the reliability of this mechanism would not suffice for safety-critical applications where beacons are sent at a high rate [7]. In contrast, approaches that delegate the message verification to the system infrastructure, e.g., RSUs, can considerably reduce the heavy burden of the continuous message verification on the resource-constrained OBU [19], [20]; yet, their effectiveness remains questionable, specifically in safety-critical applications, due to the performance impact by the heavy load of messages and the need for widespread deployment of RSUs.

Our approach avoids the aforementioned shortcomings; it pursues a novel delegation mechanism for message (source) authentication and data integrity verification by associating each ITS node with a verifier. Hence, the heavy burden from the continuous message verification is evenly distributed between verifiers of the ITS nodes since each verifier will check messages from a single source. Additionally, unlike autonomous and cooperative approaches, our delegation mechanism efficiently utilizes the available resources since the cost for authenticating a single message is paid once at a particular verifier. Furthermore, message authentication requires performing a single lightweight symmetric cryptography to ensure fast verification while tackling the vulnerabilities of single key cryptography and satisfying non-repudiation.

We further avoid the overhead of digital certificates by leveraging blockchain and smart contracts for the verification and revocation of node certificates. While some published schemes have utilized blockchain for certificate management in the context of vehicular networks, e.g., [21], [22], and [23], these schemes still rely on the computationally-heavy PKI techniques. Our message authentication approach avoids PKI in order to suit latency-sensitive and resource-constrained safety-critical applications. DAITS is essentially certificateless where traditional PKI-based digital certificates are not adopted. In other words, neither digital certificates nor CRLs are shared or stored in the blockchain. Instead, smart contracts are employed to replace digital certificates and offer efficient status verification. A smart contract is deployed at a permissioned blockchain network for every node in the system to maintain and report the status for the node's certificate. Access to the blockchain is restricted to system facilities, e.g., RSUs. As a result, our mechanism is lightweight, efficient, and secure. We emphasize that our approach will not further increase the infrastructural support; rather, it aims to reduce it by minimizing the heavy reliance on the TA and RSUs.

III. SYSTEM DESCRIPTION

In this section, we first highlight the security requirements and provide a brief background. Then, we discuss the system architecture, the threat, and communication models for DAITS.

A. Security Requirements

Due to the high frequency and volume of message broadcast, the public insecure nature of vehicular communications, and the required low message delivery latency, an authentication scheme for ITS applications ought to fulfill the following goals:

1. *Sender authentication*: The legitimacy of a message source (data origin) has to be verified by receivers in order to thwart attempts by malicious nodes to send bogus messages.
2. *Message Integrity*: It is important to verify that the message content is genuine and has not been manipulated, e.g., through a Man-in-the-Middle attack.
3. *Non-repudiation*: To establish accountability, a sender should not be able to deny the transmission of the message.
4. *Low overhead*: Some of the ITS applications require instant message verification despite the limited resources that a device has. Examples include collision avoidance and platoon coordination messages. Therefore, the cost and number of the cryptographic operations for the authentication process should be minimized to reduce delay.
5. *Scalability*: The authentication scheme has to be scalable to cope with the size and heterogeneity in the ITS applications.
6. *Resiliency to known attacks*: The authentication scheme has to thwart cyberattacks such as impersonation, message replay, and message flooding, launched by a single actor.

B. Hash-Based Message Authentication Code

The short processing time along with low communication overhead make hash-based authentication an efficient approach [7]. A Message Authentication Code, also called MAC tag, is created by leveraging a keyed hash function algorithm, known as HMAC, which takes as an input a secret key and message content and generates a unique MAC tag for the message. The MAC tag can be used to ensure the integrity of the message content and the authenticity of message source (sender) [24], i.e., $MAC_{tag} = HMAC(secretkey, message)$. It is practically infeasible to reproduce the same MAC tag without knowing the key [7]. When receiving a message, the receiver regenerates the MAC tag using the secret key that is shared with all nodes and compares the generated tag with the one attached to the message by the sender. If both tags are identical, this implies the message has not been manipulated and is being generated by a trusted sender. Hence, this technique is based on symmetric cryptography. Using a strong hash function and a proper cryptographic key significantly boosts the robustness of the underlying HMAC algorithm [24].

However, symmetric cryptography has some drawbacks when being utilized in vehicular communications. First, using a single key in VANET is problematic since if this key is leaked, the entire network will be compromised. That is, the adversary can join the network and hence becomes able to send/receive, intercept, and manipulate messages. Second, symmetric cryptography does not support non-repudiation, which is a critical security feature in VANET. Since all messages that are being sent in the VANET use the same single key, any node (message source) can easily deny the transmission of the message. However, this feature can be guaranteed when using public key cryptography (asymmetric), yet at the price of much higher overhead which could be

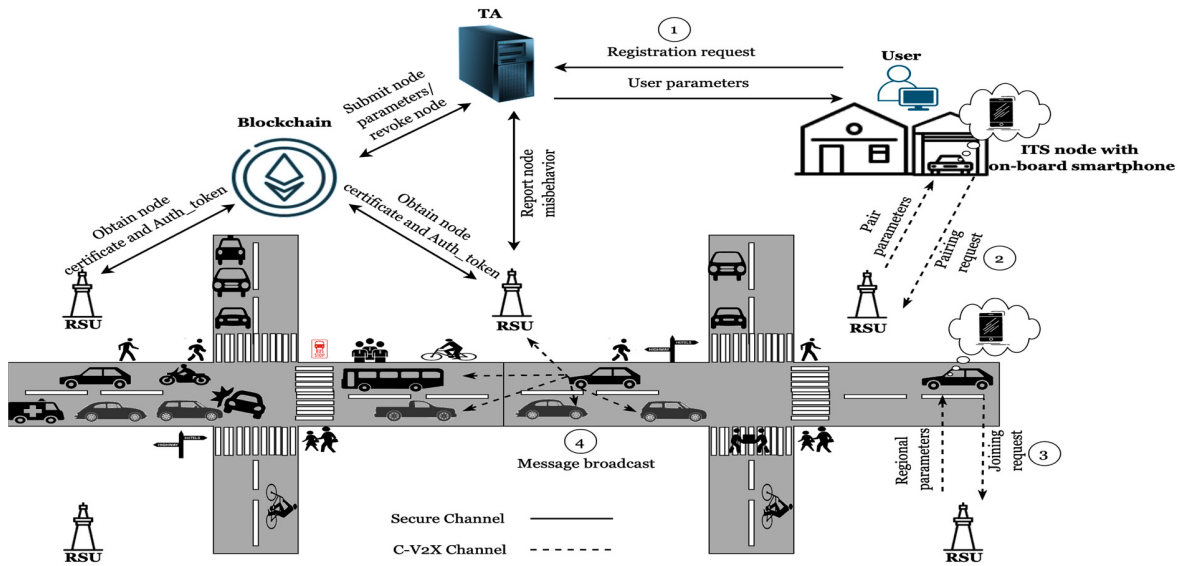


Fig. 1. An overview of the proposed security solution, highlighting the main players. The numbered arrows reflect the phases of the DAITS protocol.

unbearable for resource constrained ITS designs. To still take advantage of symmetric cryptography, our DAITS approach makes some adjustments. Instead of using a single shared key, we assign a unique key for each ITS node. Hence, DAITS guarantees non-repudiation since the MAC tags are created using distinct keys.

C. System Architecture

Figure 1 depicts the architecture of our proposed solution, which consists of the following entities: a TA, RSUs, permissioned blockchain, edge nodes, and heterogeneous ITS nodes that vary in their capabilities and trustworthiness level. Each entity is described as follows:

- **TA:** This constitutes the fully trusted entity in the system and is assumed to have reliable and abundant computation and storage capabilities. It is responsible for generating system parameters, registering RSUs and ITS nodes, and generating entity-specific credentials such as secret keys and seeds for authentication tokens. In addition, the TA is responsible for revoking system entities. Some of the existing work pursues a hierarchical architecture where the TA is the root; we argue that such a hierarchical management structure increases the overhead. Hence, the TA in our solution can be a central entity given its limited role. Specifically, the TA does not participate in authenticating the ITS nodes.
- **RSU:** The RSUs are computing and communication devices deployed at roadsides. They are assumed to be semi-trusted and equipped with sufficient resources. The RSUs are responsible for authenticating ITS nodes and smartphones, handling joining requests to their geographical regions, and reporting any observed misbehavior of ITS nodes to the TA. We assume a more realistic and feasible ITS ecosystem in which the pervasive deployment of RSUs is not guaranteed. Hence, it is important to mitigate the heavy burden imposed by continual flow of authentication messages on RSUs.

- **Blockchain:** It is a permissioned secure distributed ledger that is managed and controlled by the TA and can be accessed only by RSUs. Our solution employs such blockchain and its smart contracts to handle the certification of nodes. Specifically, a node's certificate will be verified through a function call to its smart contract and the certificate will be revoked by submitting a transaction to the smart contract. Each smart contract stores a certificate-status, which reflects the legitimacy of the node, and a current value of an authentication token, $Auth_token_i$, for the node. Furthermore, the blockchain will be utilized for sharing system and regional parameters which can prevent being accidentally exposed to attackers during transmission, but rather, they will be obtained through function calls to smart contracts. The smart contract will be deployed at the blockchain after the registration of the associated node and is controlled by the TA.
- **Edge nodes:** They constitute mobile edge-computing nodes and are referred to as verifiers. Those nodes are considered untrusted; yet they're equipped with moderate resources that can be leveraged. These nodes will be employed to offer an authentication-as-a-service for the resource-constrained ITS nodes. Unrelated to our security solution, edge nodes can also be used to offload moderate computations and provide traffic-related services such as analytics (route optimization). The role of the edge nodes will be played by smartphones and hence no additional infrastructural support is required for this service. This is unlike contemporary approaches in the literature that assume many stationary RSUs to support frequent authentication of ITS nodes. While the driver's smartphone can naturally play the role of a verifier, the availability of service must be guaranteed in case the ITS node's default verifier (driver's smartphone) is out of service, e.g., has a dead battery. The details of how our solution handles such an issue will be provided in Section IV.

- ITS nodes: They constitute entities that either generate or request traffic-related data such as vehicles, buses, trains, etc. They are untrusted entities in the system and are equipped with limited processing and storage resources such as on-board units (OBUs). Thus, the overhead due to the continuous authentication of messages ought to be mitigated. Our security solution aims to provide authentication and message verification for the ITS nodes in an as-a-service fashion by the edge nodes, as noted above.

D. Threat Model

DAITS considers two types of adversaries: external and internal. The former is a global passive adversary that is equipped with sufficient resources to intercept and analyze broadcasted messages in the entire network. This could be achieved by deploying antennas to intercept messages transmitted by ITS nodes and RSUs. The captured messages are then correlated with a target node to infer the travel trajectory. This type of adversary lacks the system security parameters needed to maliciously participate in the ITS as it is not a legitimate member in the system. Meanwhile, an internal adversary is a part of the ITS system, e.g., a compromised node. Hence, such an adversary is aware of the system parameters and can actively participate in the network operation. By countering both these external and internal adversary types, DAITS withstands potential attacks such as broadcasting bogus messages, eavesdropping, traffic analysis, message replay, message manipulation/injection, node impersonation, and message flooding. While multiple nodes could be compromised, collusion among these nodes is not possible. In other words, coordinated multi-actor attacks are out of the scope of this paper.

E. Communication Model

Interconnectivity between various ITS components will be offered by utilizing Cellular Vehicle-to-Everything (C-V2X) technology. C-V2X is a promising cellular technology based on LTE/5G and is designed to offer vehicle-to-vehicle, vehicle-to-infrastructure, vehicle-to-pedestrians, vehicle-to-cloud-based services. Such technology provides various advantages over alternative vehicular communication options. First, C-V2X is a part of the 5G rollout and leverages the comprehensive coverage of the secure LTE network. It enables highly reliable communication at a high speed even under heavy load. Furthermore, C-V2X supports both short and long-range transmission between vehicles and roadside infrastructure. There are two transmission modes in C-V2X: (i) direct communications between ITS nodes, that works independently of the cellular networks, and (ii) network communications which leverages the conventional mobile network for the transmission of the road-related information. DAITS assumes the former mode, i.e., direct interaction between ITS nodes.

IV. SYSTEM OPERATIONAL PHASES

DAITS involves the following six phases: initialization and registration, pairing, joining, secure message dissemination, delegated message verification, and user revocation. Each phase is discussed in detail in the balance of this section and Table I lists the parameters will be encountered in this section.

TABLE I
DESCRIPTION OF DAITS' PARAMETERS

Parameter	Creator	Purpose
$admin_{sk}$	TA	Generation of node-related secret parameters
$system_{key}$	TA	Generation of regional keys
RSU_{key}	RSU	Generation of regional keys
$node_{sk}$	TA	Generation of authentication tokens and MAC tag ₂
$Token_{seed}$	TA	Generation of authentication tokens
$Auth_{token}$	ITS node & its smart contract	Authentication and masking secret parameters
$Smart_{contract}$	TA	Querying node certificate and authentication token
$Mutual_Pseudo_{id}$	Smart contract	Shared pseudo identity for a pair of nodes
$Mutual_{token}$	RSU	Shared authentication token for a pair of nodes
k_{random}	ITS node	Generation of secret key for the pair
$pair_{sk}$	ITS node	Generation of MAC tag ₁
PD	RSU	Pseudo identity for privacy protection
$Regional_Pseudo_{agen}$	RSU	Verification of pseudo identities
$Regional_{key}$	RSU, Verifier, and ITS node	Generation of MAC tag
Θ	RSU	Generation of pseudo identities

A. Initialization and Registration (ITS Node ↔ TA)

As an initialization of our approach, the system TA, e.g., the department of motor vehicles, determines the system-wide parameters, namely, a lightweight one-way hash function h , an admin secret key $admin_{sk}$, and a system key $system_{key}$. A user communicates with the TA through a secure channel to register its vehicle and smartphone. The user could send the driver's license number, vehicle VIN, and smartphone's International Mobile Subscriber Identity (IMSI). The TA checks its database to ensure the validity of the provided information and makes sure that they are not associated with another registered user in the system. A user is expected to keep the information current, e.g., notifies the TA when selling a vehicle, changes smartphone service, etc. It is worth noting that updating a vehicle owner's record is usually mandatory by the authorities. If the provided information by the user is valid, the TA determines node-related parameters, specifically a random seed value for generating authentication tokens and a secret key. The secret keys for the vehicle and the smartphone are calculated as:

$$vehicle_{sk} = h(vehicle_{id} \parallel admin_{sk}) \quad (1)$$

$$smartphone_{sk} = h(smartphone_{id} \parallel admin_{sk}) \quad (2)$$

Each authentication token, $Auth_{token}$, is computed through the hash function, h , such that $Auth_{token}_{i+1} = h(Auth_{token}_i \parallel node_{sk})$. Additionally, the TA creates a smart contract for each registered ITS node at the private (permissioned) blockchain network. The smart contract stores the node's certificate-status and current $Auth_{token}_i$. Then, the TA provides to the user the secret keys, the token seed values, and the addresses of relevant smart contracts at the blockchain; such information will be stored in the vehicle's OBU and

the user's smartphone. A node would have to present a valid address of its smart contract at the blockchain network to any system facility to prove its registration status. Moreover, the node's authentication tokens can only be calculated by the node and its associated smart contract. It should be emphasized that a particular *Auth_token* is only valid for a complete system operational phase; hence, it becomes invalid when the phase is completed. Moreover, no other node will be able to derive *Auth_token_{i+1}* given the current *Auth_token_i*. We note that the access to the blockchain is restricted to the TA and its associated infrastructure such as RSUs. Hence, ITS nodes and verifiers have no access to the blockchain.

B. Pairing (ITS Node ↔ Verifier)

This phase is meant to associate an ITS node, e.g., vehicle, to a verifier, i.e., smartphone, after mutually authenticating each other with the help of a nearby RSU. Specifically, this phase aims to establish a node-smartphone pair by creating a mutual pseudo-ID, agreeing on the pair's secret key, and deriving the system key where the RSU will facilitate the accomplishment of this phase. For each ITS node, the favorable verifier will be the driver's smartphone, but the service also handles the worst-case scenario when the favorable verifier is unavailable, e.g., dead battery. This phase is established once the driver rides the vehicle; when accomplished successfully, the pairing mode continues for the entire trip unless the smartphone becomes unavailable. The messages between RSU and smart contracts are basically function calls that will be executed inside a secure blockchain environment. The transactions generated by RSU (RSU-to-Blockchain) are secured using the blockchain public key cryptography. During this phase, the messages between a vehicle and smartphone are transmitted using short-range communication means, e.g., WiFi or Bluetooth. DAITS ensures the confidentiality and integrity of these messages as explained below. To start the pairing process, the following steps must be applied:

- 1) The driver's smartphone (potential verifier) sends the address of its smart contract along with a nonce, i.e., $\{smart - contract_s, nonce_s\}$, to the ITS node, i.e., the OBU on the vehicle, through a short-range wireless communication link, e.g., WiFi or Bluetooth.
- 2) In turn, the node prepares a pairing request, $\rho = \{smart - contract_v, smart - contract_s, nonce_v, nonce_s\}$ and sends it out to a nearby RSU.
- 3) The RSU calls *getCertStatus()* and *getCurrtAuthToken()* functions on each smart contract to return its certificate-status: active or revoked, and current authentication token respectively. If both are active, the RSU calls *formPair(smarts-contract_s)* function on the vehicle's smart contract while attaching the address of the smartphone's smart contract to form a pair.
- 4) The node's smart contract communicates with the smartphone's smart contract to obtain its ID. Specifically, it calls *getID()* function on the smartphone's smart contract in order to generate a mutual pseudo ID, using the Eq. (3). Then, it calls *setPair(verifier_{id}, smart-contract_s)* to store the credential of its pair, and finally, sends

$\{Mutual_Pseudo_{id}\}$ to the RSU.

$$Mutual_Pseudo_{id} = h(vehicle_{id} \parallel smartphone_{id}) \quad (3)$$

- 5) The RSU calculates a mutual token using Eq. (4) and generates m_1 and m_2 using (5) and (6), respectively. Then, it sends $\{Mutual_Pseudo_{id}, Mutual_token, m_1, m_2, \sigma\}$ to the nodes, where the signature, σ , is generated using Eq. (7).

$$Mutual_token = Auth_token_{v,i} \oplus Auth_token_{s,i} \quad (4)$$

$$m_1 = \{system_{key}^*, nonce_v\}, \text{ where } system_{key}^* = system_{key} \oplus Auth_token_{v,i} \quad (5)$$

$$m_2 = \{system_{key}^*, nonce_s\}, \text{ where } system_{key}^* = system_{key} \oplus Auth_token_{s,i} \quad (6)$$

$$\sigma = h\{Mutual_Pseudo_{id}, Mutual_token, Auth_token_{v,i}, Auth_token_{s,i}, m_1, m_2\} \quad (7)$$

- 6) Each node uses Eq. (8) to obtain the other's authentication token and Eq. (7) to verify the signature. If the signature is not valid, the session is to be aborted. Otherwise, each node derives the system key and stores it in its local storage.

$$Auth_token_{other,i} = Auth_token_{self,i} \oplus Mutual_token \quad (8)$$

- 7) The ITS node generates a random key, k_{random} , that is to be used to calculate the pair's secret key, i.e., $pair_{sk} = h(k_{random} \parallel node_{sk})$. The node uses Eq. (9) to mask the pair secret key and Eq. (10) to generate a signature. Finally, the node sends a message to the smartphone including $\{pair_{sk}^*, nonce_v, \sigma\}$.

$$pair_{sk}^* = pair_{sk} \oplus Auth_token_{s,i} \quad (9)$$

$$\sigma = h(pair_{sk}^* \parallel pair_{sk} \parallel nonce_v) \quad (10)$$

- 8) The smartphone derives the pair's secret key using Eq. (11) and applies Eq. (10) to verify the signature. If the signature is valid, the pair's secret key will be stored on the smartphone. This key will be used by the node to create MAC tags for its broadcasted messages, and by the smartphone to verify the authenticity and integrity of the node's messages.

$$pair_{sk} = pair_{sk}^* \oplus Auth_token_{s,i} \quad (11)$$

Upon successful completion of the pairing process, the smartphone becomes the verifier for the node's broadcasted messages. Additionally, the smartphone and ITS node can now communicate privately and securely with each other, by encrypting their communication using the pair's key ($pair_{sk}$), as being deemed a secret key. As we articulated previously, our approach could rarely encounter a situation in which the verifier becomes unavailable due to dead battery. During such an exceptional scenario, the ITS node would temporarily appoint the RSU to act as a verifier by revealing the pair's secret key to the RSU. To do so, the ITS node would have to mask the key using Eq. (9) but with its current

$Auth_token_{v,i}$ and send a message to the RSU including: $\{smart - contract_v, pair_{sk}^* \parallel tag\}$, where the MAC tag is generated using the RSU regional key. The RSU verifies the tag and contacts the smart contract to obtain the ITS node's current token to unmask the key.

C. Region Joining (RSU \leftrightarrow Pair)

Algorithm 1 is invoked periodically for a paired ITS node and verifier to introduce themselves to a nearby RSU. Specifically, the region joining phase aims to derive the regional key and pseudo agent, which will be used by the RSU and all regionally authenticated nodes, and to generate a set of shared pseudo-identities for the pair. Such parameters, namely, the regional key, pseudo agent, and the pseudo-identities are only valid within the geographical coverage of that RSU. The regional key will be utilized for the generation (by verifiers) and authentication (by ITS nodes) of the negative verification messages. The regional pseudo agent is to verify the validity of pseudo-identities. It is important to distinguish fake pseudo-identities generated by malicious nodes since no verifiers are assigned to those nodes. That is, the regional pseudo agent is motivated by invalid rather than valid confirmation from verifiers. We simply want fake messages not to be accepted and filtered out since no verifier will check them. For a regional join, the following steps must be taken:

- 1) Either the node or its verifier broadcasts a join request γ to authenticate themselves to the RSU. γ includes the pair's mutual pseudo ID and the addresses of their associated smart contracts, i.e., $\gamma = \{Mutual_Pseudo_{id}, smart - contract_v, smart - contract_s\}$.
- 2) The RSU interacts with the pair's smart contracts to check their certificate-status, active or revoked, and obtain their current authentication tokens, $Auth_token_{v,i}$ and $Auth_token_{s,i}$. It should be emphasized that the blockchain is private; hence, the access to the blockchain is restricted to selected trusted parties to the system.
- 3) The RSU performs the operations in Eq. (12), (13), and (14) to generate $Regional_key$, and mask its RSU_key and $Regional_Pseudo_agent$, respectively. Moreover, the RSU generates a set $\{PD_1, PD_2, \dots, PD_n\}$ of n shared pseudo-identities, for the ITS node and its verifier, which enable them to switch between pseudo identities and prevent a potential traceability attack by an adversary. Each pseudo-identity (PD) is generated, using Eq. (15), by enciphering $Mutual_Pseudo_{id}$ using a member of the Θ set. Θ is managed by individual RSUs and is employed to withstand message linkability attacks. In other words, Eq. (15) allows varying appearance of the $Mutual_Pseudo_{id}$ in a particular region. Then, each PD is masked using Eq. (16). A packet $\{m_1, m_2, \sigma\}$ is then sent to the ITS node and its verifier, as defined in Eq. (17). Finally, the RSU will store the record: $[PD : smart - contract_v, smart - contract_s]$ in its local storage. The time and storage requirement of maintaining such a type of records locally at the RSU will be influenced by the road traffic. For instance, in a highway scenario, the records of the authenticated ITS nodes will be short-lived since these nodes are expected

Algorithm 1 Joining a RSU Region

Data: RSU_j receives a join request γ from Vehicle V_i and Smartphone S_i :

1. call $getCertStatus()$ function;
2. call $getCurrtAuthToken()$ function;
3. **if** ($certificate-status_{v_v} == active$) & & ($certificate-status_s == active$) **then**
4. store $Auth_token_{n_{v,i}}$ and $Auth_token_{s,i}$;
5. mask RSU_key ;
6. mask $Regional_Pseudo_agent$;
7. generate a set of n pseudo-identities PD ;
8. mask PD ;
9. generate m_1 and m_2 ;
10. generate signature σ ;
11. store $[PD : smart - contract_v, smart - contract_s]$;
12. broadcast the packet $\{m_1, m_2, \sigma\}$;
13. **else**
14. abort the session;
15. **end**

Data: Vehicle V_i and Smartphone S_i receive the packet $\{m_1, m_2, \sigma\}$ from RSU RSU_j :

1. calculate current $Auth_token_i$;
2. unmask RSU_key ;
3. calculate $Regional_key$;
4. unmask $Regional_Pseudo_agent$;
5. unmask PD ;
6. generate signature σ' ;
7. **if** (σ' is identical to σ) **then**
8. store RSU_key , $Regional_key$, $Regional_Pseudo_agent$, and PD ;
9. **else**
10. abort the session;
11. **end**

to stay in a region for a short period of time due to the high mobility (travel speed).

$$Regional_key = RSU_key \oplus system_key \quad (12)$$

$$RSU_{key,v}^* = RSU_key \oplus Auth_token_{v,i} \quad (13)$$

$$RSU_{key,s}^* = RSU_key \oplus Auth_token_{s,i}$$

$$Regional_Pseudo_{agent,v}^* = Regional_Pseudo_agent \oplus Auth_token_{v,i}$$

$$Regional_Pseudo_{agent,s}^* = Regional_Pseudo_agent \oplus Auth_token_{s,i} \quad (14)$$

$$PD_i = Mutual_Pseudo_{id} \oplus \theta_i,$$

where $\theta_i \in \Theta = \{\theta_1, \theta_2, \dots, \theta_n\}$ (15)

$$PD_v^* = PD \oplus Auth_token_{v,i}$$

$$PD_s^* = PD \oplus Auth_token_{s,i} \quad (16)$$

$$m_1 = \{RSU_{key,v}^*, Regional_Pseudo_{agent,v}^*, PD_v^*\}$$

$$m_2 = \{RSU_{key,s}^*, Regional_Pseudo_{agent,s}^*, PD_s^*\}$$

$$\sigma = h(Regional_key \parallel m_1 \parallel m_2) \quad (17)$$

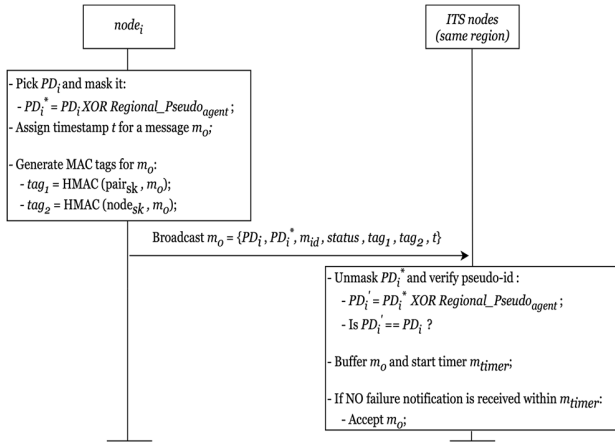


Fig. 2. Illustration of the proposed secure message dissemination.

- 4) Upon receiving the packet, both the ITS node and its verifier will first derive the current $Auth_token_i$ and use Eq. (18), (12), (19), and (20) to unmask the RSU key and derive the regional key, the regional pseudo agent, and the shared pseudo-identities. It should be noted that the $system_key$ is stored locally at verifiers and ITS nodes and is obtained from the RSU during the pairing phase. Finally, the signature σ will be verified ($\sigma' == \sigma$) using Eq. (17). If σ is not valid, the session will be aborted. Otherwise, the mutual authentication and the derivation of the regional key, pseudo agent, and shared pseudo-identities, are successful.

$$RSU_{key} = RSU_{key}^* \oplus Auth_token_i \quad (18)$$

$$Regional_Pseudo_agent = Regional_Pseudo_agent^* \oplus Auth_token_i \quad (19)$$

$$PD = PD^* \oplus Auth_token_i \quad (20)$$

It should be emphasized that the RSU's unique parameters, namely, the RSU key, the pseudo agent, and Θ are updated by the RSU at a time interval as preventative measures to ensure robustness. DAITS ensures efficiency and scalability with the minimum possible overhead on RSU and does not require pervasive deployment of RSUs. An RSU is only responsible for facilitating the node-verifier pairing, and tracking membership in its region, while the load for continuous message and sender authentication is split among verifiers.

D. Secure Message Dissemination

When an ITS node wants to broadcast a message in a particular region (administered by an RSU), it first goes through the joining phase and then applies the steps depicted in Figure 2. The specifications of DAITS' secure message dissemination are as follows:

- 1) The node picks a pseudo-identity and masks it with the RSU's regional pseudo agent using:

$$PD_i^* = PD_i \oplus Regional_Pseudo_agent \quad (21)$$

- 2) The node generates two MAC tags for the message:
 - (1) using the pair's secret MAC key for the first

tag, (2) using the node's secret key for the second tag. Then, the two tags are appended to the message being broadcasted. The packet m_0 , will consist of: $\{PD_i, PD_i^*, m_{id}, status, tag_1, tag_2, \tau\}$, where m_{id} is the message sequence number, $status$ is the node's spatio-temporal status, e.g., current time, position, speed, and direction, and τ is a timestamp. We note that the node's spatio-temporal information reflects the message payload and is not related to DAITS. The first tag is employed for authenticating the data origin and ensuring its integrity; hence tag_1 can only be validated by the node's associated verifier. In addition, this tag enables non-repudiation assurance at the pair level since only the node and its verifier know the MAC key. However, the verifier's knowledge of the MAC key necessitates the use of tag_2 to enable an additional layer of non-repudiation assurance. tag_2 is generated using the node's secret key and is employed only to achieve non-repudiation at the node level. It is only verified during a dispute, e.g., when malicious activity is reported to the TA; note that the TA is the only entity that knows the secret keys of all nodes. In essence, tag_2 could be used to identify the compromised member in the pair, i.e., either the ITS node or its verifier, in a situation when the ITS node and its verifier belong to different users, e.g., when driving a rental vehicle. Meanwhile, the user accountability is determined based on the uncovered $Mutual_Pseudo_{id}$ that relies on the sender's PD included in the malicious message. To elaborate, PD is used to uncover $Mutual_Pseudo_{id}$ which is used to reveal the identity of the user. The user accountability will be discussed later in the node revocation phase.

- 3) Every message receiver within the region will verify the sender's pseudo-identity by unmasking PD_i^* to get PD'_i and validating that $PD'_i = PD_i$. If they do not match, this implies that the pseudo-identity is fake and the m_0 packet will be dropped (the message ignored); otherwise, the receiver will buffer the packet, start a timer m_{timer} , and wait for any possible NEGATIVE verification from the sender's verifier.
- 4) If no negative verification message arrives within m_{timer} time units, the packet will be accepted and processed.

E. Delegated Message Verification

Figure 3 shows the proposed scheme for our delegated message verification mechanism. A major question is how to deliver the outcome of a message authentication check while reducing overhead. To reduce the communication overhead, DAITS adopts negative rather than positive notifications. The verifier will only send an announcement, m_1 , if the authenticity of the m_0 message could not be verified. Receivers can process the message if they do not get m_1 within a time duration m_{timer} . Meanwhile, it is crucial to detect fake m_1 message. A misbehaving ITS node or verifier should not be allowed to broadcast m_1 to wrongfully invalidate m_0 of other nodes. Moreover, an ITS node ought to be able to verify the authenticity and integrity of the received m_1 packet. DAITS tackles these issues as follows:

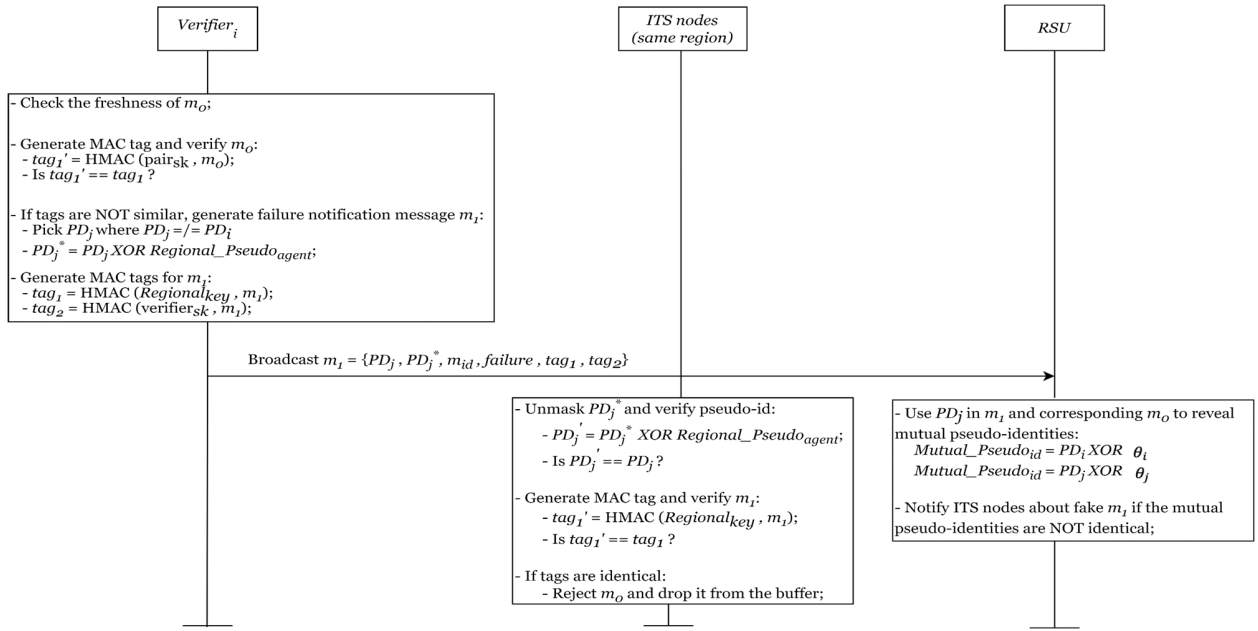


Fig. 3. Illustration of the delegated message verification procedure.

- 1) When verifiers receive the message, m_0 , they will check the pseudo-identity, PD_i , trying to recognize the message sender. Only the sender's appointed verifier will be able to do so; such a verifier will then check the timestamp to ensure message freshness (to counter message replay attacks) and use the pair's secret key to regenerate tag_1 .
- 2) The regenerated tag_1 is then compared to the one appended to the message. If the two tags are identical, the authenticity of the message is verified, and no further action is needed.
- 3) If the tags do not match, the designated verifier sends a negative verification message m_1 within the allowed time frame. The verifier will pick a different pseudo-identity from PD_i , and mask it with the RSU's regional pseudo agent; this is done to prevent message linkability, as we explain below. Then, it generates two MAC tags for the m_1 message: (1) using the regional key for the first tag, (2) using the verifier's secret key for the second tag. The regional key was obtained during the joining phase. tag_1 is employed for the authentication of the data origin and integrity of the failure notification message and hence it can be verified by the waiting receivers. Meanwhile, tag_2 is employed to achieve non-repudiation at the verifier level during a dispute. Thus, the failure verification message is $m_1 = \{PD_j, PD_j^*, m_{id}, failure, tag_1, tag_2\}$, where m_{id} is the identifier of the invalidated message and $PD_j \neq PD_i$.
- 4) When m_1 is received by the ITS nodes, they will first verify the sender's pseudo-identity. Then, they will regenerate the tag_1 using the regional key to ensure the integrity of m_1 . If the tags match and RSU has not flagged out m_1 within m_{timer} , the ITS nodes will reject m_0 .
- 5) Meanwhile, the RSU would use the PD_j and PD_i included in m_1 and m_0 to uncover the mutual

pseudo-identity of the ITS node (sender of m_0) and its verifier (sender of m_1) as: $Mutual_Pseudo_{id} = PD_j \oplus \theta_j$. This is done to ensure the legitimacy of the verifier. It's worth mentioning that the pseudo-identities for m_1 and m_0 are expected to be different (to satisfy message unlinkability requirement). If the revealed mutual pseudo-identities are not identical, it means that the m_1 message has been generated by a misbehaving node pretending to be the appointed verifier, and consequently, the RSU will: (i) first flag out m_1 by notifying neighboring ITS nodes to ignore it and accept m_0 instead, and (ii) apply the user revocation procedure.

Lastly, we should emphasize that unlike the safety messages broadcasted by ITS nodes, the frequency of negative verification messages is expected to be quite small. Therefore, such a verification step by the RSU should incur an insignificant overhead compared to the verification overhead of the safety messages. Furthermore, given the availability of on-board smartphone charging, we believe that the probability of verifier unavailability due to dead battery is expected to be very low. Hence, the overhead on the RSU for occasionally acting as a verifier will be negligible.

F. Node Revocation

When misbehavior, i.e., broadcast of malicious message, is reported to an RSU, the RSU applies a consensus algorithm to ensure the validity of such a claim, e.g., by requiring that the misbehavior be reported by a certain number of nodes before informing the TA. If the report is deemed valid, the PD, included in the message is factored in to confirm the identity of accused user; specifically, the PD will be used to uncover the pair's $Mutual_Pseudo_{id}$ that is then used by the system admin to reveal the identity of the associated user. Consequently, the user is to be revoked from the system which technically involves his/her ITS node and mobile device. We note that both the ITS node and the smartphone are

being viewed as assets of the same user; hence, the user is held accountable for the misbehavior by any of these assets. Similarly, when the ITS node and its verifier do not belong to the same user, e.g., rental vehicle, the corresponding users of the ITS node and verifier are both held accountable, unless the compromised member in the pair is identified using its tag_2 signature. Additionally, the revocation is also applied when the RSU detects misbehavior from a verifier, e.g., broadcasting fake negative verification for unpaired ITS node. For that, the RSU applies the steps in Algorithm 2, which is described as follows:

Algorithm 2 User Revocation

Data: RSU_i detects malicious behavior from an enrolled node:

1. unmask PD_j^*
2. **if** $(PD_j' == PD_j)$ **then**
3. find[$PD : smart - contract_v, smart - contract_s$];
4. delete [$PD : smart - contract_v, smart - contract_s$];
5. create new RSU_{key} and $Regional_Pseudo_{agent}$;
6. **for** each authenticated pair remained in the memory **do**
7. mask RSU_{key} ;
8. mask $Regional_Pseudo_{agent}$;
9. broadcast $\{RSU_{key,v}^*, RSU_{key,s}^*,$
 $Regional_Pseudo_{agent,v}^*, Regional_Pseudo_{agent,s}^*\}$;
10. **end**
11. derive $Mutual_Pseudo_{id}$;
12. report to TA;
13. **else**
14. drop the packet;
15. **end**

Data: TA receives a report from RSU_i about malicious behavior:

1. call $getID()$;
 2. verify tag_2 ;
 3. call $setCertStatus(revoked)$;
-

1. The RSU uses the pseudo-identity, PD_i , to locate the credential of the ITS node/verifier pair, i.e., [$PD : smart - contract_v, smart - contract_s$], and will delete the record from its list of regionally authenticated nodes.
2. Then, the RSU will generate a new key, RSU_{key} , and regional pseudo agent and then mask them with the current authentication tokens of the regional authenticated nodes on the list, (like Eq. (13) and (14) in the joining phase).
3. The RSU will send a key update message to its regional authenticated nodes.
4. The RSU will then reveal the mutual pseudo-identity of the misbehaving ITS node/verifier by using Eq. (22) and send a report to the TA.

$$Mutual_Pseudo_{id} = PD_i \oplus \theta_i \quad (22)$$

5. Upon receiving the report, the TA retrieves IDs of the ITS node and its verifier by calling function $getID()$ on $smart - contract_v$ and $smart - contract_s$,

respectively. These IDs will be used to identify the corresponding user(s) in the database. If the ITS node and its verifier belong to different users, the TA generates two copies of tag_2 by using $node_{sk}$ and $verifier_{sk}$. Then, the two copies are matched with tag_2 included in the malicious message to identify the compromised member in the pair. Last, the TA sends a blockchain transaction to the smart contract of the compromised node to mark its certificate-status as revoked, i.e., call $setCertStatus(revoked)$. Meanwhile, when the ITS node and its verifier belong to the same user, the certificate-status of both nodes will be revoked as their corresponding user is deemed malicious.

V. SECURITY ANALYSIS

This section provides both formal and informal analysis of the security and privacy properties in DAITS and demonstrates how it prevents various contemporary attacks. It also shows the results of formal validation using the AVISPA tool.

A. Analyzing Security Properties and Scalability

The following lemmas focuses on how DAITS meets the security requirements and mitigates various attacks.

Lemma 1: DAITS enables a verifier to authenticate messages broadcasted by an ITS node.

Proof: Since we rely on the hash-based message authentication code as the security primitive to achieve authentication, it is sufficient to show that by having access to the MAC key used by the sender, a verifier would be able to authenticate the messages. By successfully executing the pairing phase between an ITS node and verifier, e.g., vehicle and driver's mobile device, the mobile device would gain access to the pair's secret key ($pair_{sk}$). Hence, the mobile device will be able to authenticate future messages broadcasted by the vehicle. Based on Eq. (9) in the pairing phase, the pair's secret MAC key is shared securely with the mobile device by enciphering the key using the mobile device's secret $Auth_token$. Therefore, only the mobile device can act as a designated verifier for the node since it is the only one that has access to the node's MAC key.

Lemma 2: DAITS detects bogus messages broadcast.

Proof: To fool legitimate nodes to accept fake messages, an adversary should have access to either regional keys or pseudo agents, that are controlled by individual RSUs. Per Eq. (12), (13) and (14) of the region joining phase, the regional parameters are shared securely with a node by enciphering them using the node's secret authentication token, $Auth_token$. Furthermore, the regional parameters are updated at a regular time frame while the node's $Auth_token$ is only used once. Consequently, ITS nodes can autonomously detect fake messages broadcasted by the adversary. This is achieved by unmasking the pseudo-identity attached to the message with the help of the regional pseudo agent. Since the adversary is not an authenticated regional member under an individual RSU, it will not be able to generate either a legitimate pseudo-identity or failure notification message since the regional parameters are not known. Furthermore, since

every legitimate ITS node is paired with a verifier, when the adversary reattaches a legitimate pseudo-identity to a fake message, this will result in a negative verification by the verifier since the adversary is unaware of the MAC key.

Lemma 3: DAITS prevents message modification, injection, and replay attacks.

Proof: To modify a particular message or inject malicious content into the message, an adversary should have access to the MAC key or regional key. As shown in the previous lemmas, the adversary cannot have access to these parameters. As a result, any message manipulation will be detected by the assigned verifier since the manipulation will result in a different MAC tag from the one generated by the message source. Meanwhile, timestamps are employed to counter replay attacks. A verifier will first check the freshness of a message before authenticating the message. Therefore, the adversary cannot disturb the system by replaying messages.

Lemma 4: DAITS withstands Denial-of-Service attacks through message flooding.

Proof: A receiver can verify a pseudo-identity through a single bitwise XOR operation that incurs a negligible and infinitesimally small computational overhead. Thus, when an external adversary floods the system with fake messages, they can be filtered out without exhausting the resources of the target nodes. Meanwhile, exploiting a compromised node to flood the system with genuine messages is ruled out since the impact of such an attack will be limited to the verifier of the compromised node, which basically belongs to the same user. Moreover, DAITS ensures non-repudiation, and the TA will eventually be informed to exclude the malicious node.

Lemma 5: DAITS achieves conditional privacy for ITS nodes and prevents message linking attacks from tracking the travel trajectory.

Proof: To obtain the real identity of a pair, their $Mutual_Pseudo_{id}$ must be uncovered. Even if $Mutual_Pseudo_{id}$ becomes known, the real identity of the pair cannot be revealed since the employed one-way hash function is irreversible. Furthermore, since blockchain is permissioned, only the TA has administrative control and can query the node ID through its smart contract. Therefore, DAITS guarantees a conditional privacy where a pair stays anonymous.

In addition, DAITS assigns unique pseudo-identity for every pair, i.e., an ITS node and its verifier, when joining an RSU region. Even when rejoining the same region again, the pair are issued different pseudo-identity due to the periodic regional parameters update; specifically, the update of the Θ by the RSU. The pseudo-identities are only valid within the issuing region, and they are meaningless, except to the issuing RSU agent. Furthermore, DAITS uses a different pseudo-identity for a message from its associated negative verification. To crack pseudo-identities, an adversary must have access to the regional Θ that was used to encipher the pair's $Mutual_Pseudo_{id}$. Since the adversary has no access to the regional Θ and they are updated regularly, the collected pseudo-identities cannot be exploited by a tracker as they belong to different issuing agents and they cannot be linked to each other. Since pseudo-identities are only valid within the geographical region under the control of the RSU, an adversary needs to compromise all the RSUs within the trajectory of the

target pair to crack all pseudo-identities used by the pair in order to infer the driving trajectory. Therefore, it is practically impossible for an adversary to track the location of the ITS nodes by eavesdropping on transmitted messages in the entire network.

Lemma 6: DAITS sustains non-repudiation at the pair level as well as the node level.

Proof: DAITS assigns a unique MAC key for every pair that is known only to them. Thus, the pair cannot deny the transmission of a message if the enclosed tag was generated using their MAC key, and non-repudiation is enforced at the pair level. To achieve non-repudiation at the node level, another MAC tag is attached to the message using the node's secret key that is only known to the node and the TA. Hence, an individual node cannot deny being the source of the transmission if the second MAC tag of the message was generated using its secret key. Therefore, DAITS ensures non-repudiation at the pair level, through the pair secret key, and the node level, through the node's secret key.

Lemma 7: DAITS mitigates verifier's untrustworthiness.

Proof: A legitimate verification-failure message should satisfy: $Mutual_Pseudo_{id} = Mutual_Pseudo'_{id}, \forall PD_i \neq PD_j$ where $Mutual_Pseudo_{id} = PD_i \oplus \theta_i$, and $Mutual_Pseudo'_{id} = PD_j \oplus \theta_j$. Based on Eq. (15) and (16), a malicious verifier can neither replicate nor intercept the list of PDs of another pair due to the unawareness of Θ , $Auth_token_{v,i}$, and $Auth_token_{s,i}$. Thus, a malicious verifier cannot successfully disturb the operation of the system by either using fake PD or impersonating another verifier to deliberately invalidate another node's messages. As shown in Figure 3, this will be detected and flagged out by the monitoring RSU due to unmatched $Mutual_Pseudo_{id}$ since the verifier is unable to generate PD_j that satisfies the formula. Moreover, when a compromised verifier chooses not to validate malicious/fake messages broadcasted by its associated ITS node, those messages will be reported by neighboring ITS nodes to the RSU which may then result in revocation procedure as explained in subsection F in Section IV.

If a compromised verifier attempts to launch a DoS attack by flooding a particular region with fake negative verification reports while hiding its identity using fake: (i) PD and (ii) key to generate tag_2 , an RSU could be employed for thwarting such an attack. The monitoring RSU could utilize the Received Signal Strength Indicator (RSSI) to measure proximity to the message transmitters, i.e., the vehicle's OBU or the verifier. Given that the OBU and the verifier are aboard the same ITS node, the variation in the distance measurements will be consistent for messages broadcasted by the vehicle's OBU, and the "fake" negative verification messages. Therefore, the RSU could correlate multiple fake reports with messages associated with the same source vehicle, i.e., m_0 of the same ITS node, based on the distance measurements and reception time to determine the location of the malicious verifier. The RSU should first ensure that the PDs in the located set of m_0 messages are associated with the same ITS node, i.e., PDs point to the same $Mutual_Pseudo_{id}$. Then, the RSU reports the $Mutual_Pseudo_{id}$ of the pair to the TA, as explained in the node revocation procedure. Thus, DAITS tackles all potential misbehavior by verifiers due to the untrustworthiness.

Lemma 8: DAITS prevents a pair from using past knowledge to deceive a member of the system.

Proof: In DAITS, past knowledge cannot be exploited to deceive members of the system since the system relies on regional parameters, i.e., RSU key, regional pseudo agent, and Θ , that are controlled by individual RSUs and updated at a regular interval to ensure the security and robustness of the approach. Thus, a pair must successfully execute the RSU joining phase to obtain the current regional parameters and hence be able to actively engage in the network.

We note that the update frequency and the size of the Θ , which determines the number of legit pseudo-identities, depends on: (i) the frequency of the message broadcast, (ii) the time the pair is expected to be in the region which is affected by the traffic status and model. For instance, a pair would be expected to stay longer in a school zone than on a segment of a highway. Thus, the longer stay within a region requires more pseudo-identities, and less frequent parameters update. Therefore, a pair would not be able to exploit past knowledge to either deceive an RSU or other nodes. If the same pair enters the RSU region again, they would not be able to use their prior pseudo-identities to participate in the network without successfully executing the joining phase.

Lemma 9: DAITS prevents a misbehaving node from participating in the network.

Proof: Since a smart contract validates the legitimacy of a node to join the system, it is sufficient to show that a revoked node cannot modify its certificate status in the smart contract. First, DAITS relies on a permissioned blockchain that is fully controlled by the TA. Individual nodes have no access to the blockchain. Furthermore, the access of RSUs to the blockchain is limited and cannot modify, add or delete a smart contract. Thus, neither an individual node nor a compromised RSU can change the certificate status. Consequently, when the TA revokes the certificate of a particular node, the status cannot be changed by any entity other than the TA, and the revoked node cannot convince an RSU to join its regional network. Even if the revoked node deceives an RSU by presenting another smart contract address with valid certificate status, the joining phase will fail due to the node's unawareness of the current authentication token and the secret key. Therefore, DAITS' novel smart contract-based node certification is secure and can effectively evict a node from the ITS system.

For safety critical ITS applications, messages are sent periodically. It is important for a message to be timely processed. Naturally such processing should be complete before the next message is sent. Therefore, the authentication process should be rapid, which implicitly requires lightweight verification and scalability for large settings. The following two lemmas address such a requirement.

Lemma 10: DAITS achieves timely message authentication even for high traffic density and messaging volume.

Proof: DAITS employs symmetric cryptography rather than PKI to provision message tags. Assume that it takes η time units to authenticate a message. Since DAITS pairs a stand-alone verifier to a single ITS node, the authentication time will be η units no matter what the traffic density is and the number of messages received within a particular time frame.

Lemma 11: DAITS is a scalable and resource-efficient security solution.

Proof: DAITS assigns an individual verifier to a single message broadcaster. As a result, the cost of authenticating a single message is incurred only once, i.e., at the assigned verifier, rather than by every receiver. Therefore, DAITS can scale very well to suit the heterogeneity in the ITS.

B. Validation Results

We have also utilized AVISPA [25] for validating the security properties of critical phases in our approach. AVISPA is a push-button tool for the automated validation of internet security protocols and applications. The validation is demonstrated by listing the specified security goals and the validation result for each phase as follows:

1) *Pairing Phase:* To validate this phase, we have specified two security goals. First, the authentication tokens suffice for the RSU to authenticate an ITS node and its associated mobile device (verifier). That is, the RSU authenticates the node and the mobile device on their current authentication tokens, i.e., $Auth_token_v$ and $Auth_token_s$, since no one can calculate these values except the nodes and their smart contracts. Second, the pair's MAC key stays secret and is only known to the associated pair. As shown in Figure 4 (a), AVISPA deems this phase safe, which means the security goals are satisfied and no vulnerability is found.

2) *Joining Phase:* We have validated this phase based on the following security goals: (i) maintaining the secrecy of the regional key, regional pseudo agent, and the set of the pseudo-identities for the ITS node and its verifier; they should not be known to any entity other than the corresponding nodes, and (ii) authenticating an ITS node and mobile device by an RSU on the value of $Auth_token_v$ and $Auth_token_s$ respectively. Figure 4 (b) depicts the validation result through the AVISPA tool.

3) *Secure Message Dissemination and Verification Phases:* We have validated these phases based on the fulfillment of the security goal of authenticating a verifier (mobile device) by ITS nodes (receivers); both the ITS nodes and the verifier should generate the same MAC tag. In particular, the message receivers are to make sure that the appended tag is indeed created by a legitimate verifier and is not replayed from a previous session. As shown in Figure 4 (c), AVISPA deems these phases safe, which means that no vulnerability is found.

VI. SIMULATION AND PERFORMANCE EVALUATION

We have utilized Veins [26] to study the performance of our proposed delegated message authentication scheme. Veins, which stands for Vehicles in Network Simulation, is an open-source vehicular network simulation framework that combines OMNeT++, an event-based network simulator, and SUMO, a road traffic simulator, in one framework. It has been widely adopted for the simulation of vehicular networks in academia. In our settings, the Instant Veins virtual machine 5.2-i1 was installed on Oracle VM VirtualBox that was running on a machine equipped with 8 GB RAM and 2.8 GHz Intel Core i7 processor. We have utilized Crypto++ 8.6.0 library [27], a popular free C++ class library of cryptographic schemes,

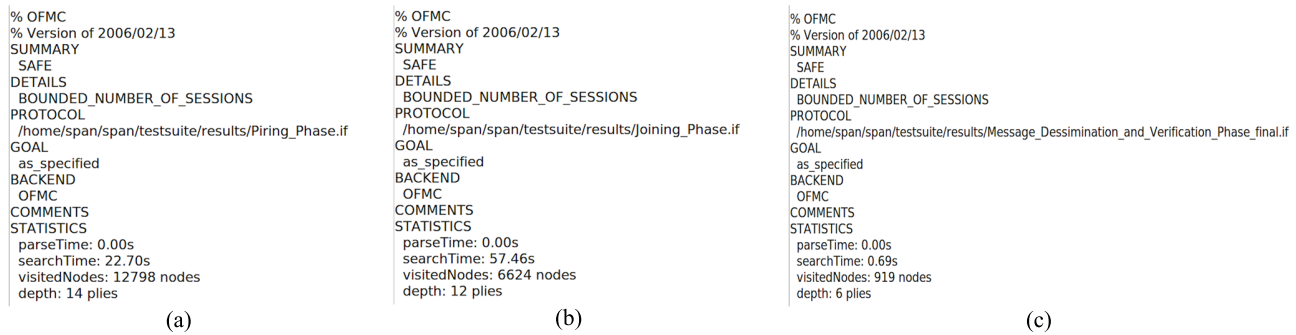


Fig. 4. Screenshots of the AVISPA validation results for the: (a) pairing phase, (b) joining phase, and (c) secure message dissemination.

to implement the message authentication code algorithm; specifically, HMAC-BLAKE2b with 512-bit key size.

We have used the example of the traffic model scenario that comes with Veins by default. In that scenario, there is a single stationary RSU and a set of vehicles that are driving sequentially around the city of Erlangen, Germany where the size of the simulated area is 2500m (X) x 2500m (Y) x 50m (Z). While SUMO governs the insertion of the vehicles into the simulated traffic model, a single vehicle joins the traffic every 3 seconds. Hence, the time between successive vehicles is 3 seconds. Additionally, the maximum speed for the vehicles is 14 m/h, the speed acceleration is 2.6 m/h, and the average trip time for vehicles is 147.52 seconds. During the simulated traffic scenario, a complete stop is reported by a vehicle due to an accident by sending a message to the RSU and all approaching vehicles. As a result, some of the approaching vehicles turn around and change their route to avoid the traffic congestion. We have modified this scenario to serve our needs. First, we have prevented the occurrence of the accident to increase the traffic density by allowing all the vehicles to keep following on the same trajectory. Furthermore, we have enabled beaconing by allowing vehicles to send Basic Safety Messages (BSMs) through IEEE 802.11p MAC layer protocol. Then, we have modified the packet payload of the BSM to accommodate the contents of the packet previously discussed in the Secure Message Dissemination subsection. After that, we have implemented DAITS to authenticate the vehicle's periodic BSMs since they are highly frequent messages. We have set the beacon interval to 100 milliseconds to satisfy the safety-critical applications requirements.

In order to gauge the efficacy of DAITS, we compare it with well-known baseline schemes, namely, TESLA, ECDSA, VAST [11], and PBA [28], as well as recently published solutions such as [13] and [20]. Moreover, we compare the overhead of DAITS with various pseudonym-based authentication schemes in VANETs [29], [30], [31], [32], [33]. We note that these schemes do not rely on Blockchain for the authentication of message broadcast; similarly, the use of Blockchain in DAITS plays no role in the authentication of BSMs.

A. Communication Overhead

Figure 5 compares DAITS with the baseline and competing schemes in terms of the incurred communicational overhead. We note that the communication overhead is calculated based

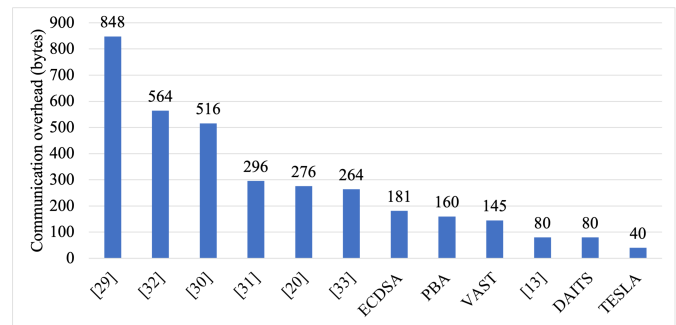


Fig. 5. Comparing the communication overhead of DAITS to that of competing schemes.

on the size of the cryptographic bit-strings, i.e., MAC tag, certificate, signature, etc., that are attached to the BSM and employed as security parameters to achieve the authentication. TESLA incurs the least overhead as it requires the beacon message to include a MAC tag of size 20 bytes along with 20 bytes for the disclosed symmetric key; hence, the total overhead is 40 bytes. Meanwhile, the scheme of [13] involves 80 bytes that consist of two MAC tags of size 20 bytes each, 20 bytes disclosed key, and a prediction value of size 20 bytes. On the other hand, DAITS incurs 80 bytes of communication overhead since the sender must attach a pseudo-identity PD , masked PD^* , and include two MAC tags, each is of size 20 bytes.

While TESLA incurs the lowest communication cost, such a scheme lacks non-repudiation assurance since the enclosed security parameters cannot identify the message sender. Furthermore, TESLA incurs a verification delay since a message cannot be authenticated until the associated key is disclosed in the subsequent message, and it would suffer from packet losses during high traffic. DAITS matches the low overhead of [13] and involves significantly less communication overhead than the other baseline schemes. Furthermore, when the relationship between communication overhead and number of nodes in communication range is considered, DAITS and [13] would similarly maintain the lowest overhead compared to all other schemes, except TESLA. However, since the scheme of [13] employs TESLA as the underlying authentication protocol, it has relied on mobility estimation to achieve instant authentication and mitigate the shortcoming of TESLA. Yet, such mobility prediction can be challenging, especially in

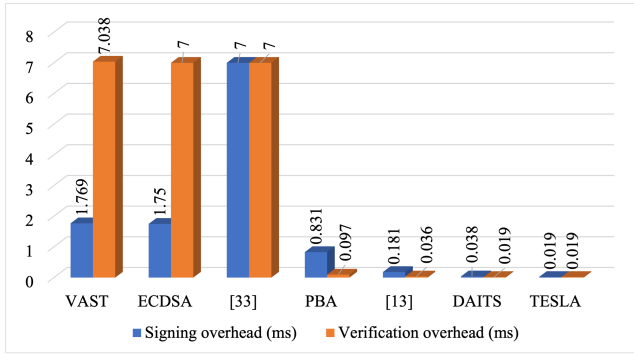


Fig. 6. Comparison of the computation overhead of DAITS to that of the baseline and competing schemes.

urban scenarios where frequent turns and changes in route and speed might be necessary. In addition, to achieve instant authentication, this scheme relies on broadcasting HMAC of the mobility prediction before sending the beacon. We believe that such a scheme will suffer during packet losses and hence its efficiency will be degraded.

B. Computation Overhead

Let PM , h , BP , BP_PA , $M2P$, $H2P$, BM , M , EXP , and XOR denote Elliptic curve point multiplication operation, hash operation, Bilinear pairing operation, Bilinear pairing-based point addition operation, map-to-point hash operation, hash-to-point function, bilinear map, modulo, exponentiation, and bitwise XOR operation, respectively. Table II enlists the underlying cryptographic operations that are needed for signing and verifying a beacon message for DAITS and the competing schemes. DAITS involves one XOR operation along with two HMAC for signing a beacon and only a single HMAC operation for verifying the beacon. Hence, DAITS outperforms all the competing schemes, except TESLA, since it incurs fewer and less expensive operations for the message signature and verification. On the other hand, TESLA requires the least computational overhead due to performing a single HMAC operation for either signing or verifying a beacon message. Moreover, as articulated previously, TESLA is an undesirable solution in the context of vehicular communications since it lacks both the non-repudiation and instant message verification and is sensitive to packet losses. Specifically, it incurs verification latency equals to the beacon interval since the message verification is delayed until the subsequent message is received. Although the approach of [13] uses TESLA, signing a beacon requires eight hash and modulo operations in addition to two HMAC operations where the signature can be verified using a single hash, HMAC, and modulo operation.

The computation overhead of DAITS and the competing schemes reported in Figure 6 are based on the execution time of the cryptographic operations calculated by Hathal et al. [13]. We note that the computation overhead of the remaining schemes in Table II are not shown in Figure 6 since they involve expensive cryptographic operations. In addition, the time of performing an XOR operation is deemed negligible. For DAITS, we have considered the potential fluctuation in the computation time due to how busy the node's processor is

TABLE II
COMPARISON WITH THE COMPETING SCHEMES IN TERMS OF THE INVOLVED CRYPTOGRAPHIC OPERATIONS

Scheme/Metric	Beacon's Signature Cost	Beacon's Verification
TESLA	$HMAC$	$HMAC$
ECDSA	PM	$4PM$
VAST	$PM + HMAC$	$4PM + 2HMAC$
PBA	$61h + 2HMAC$	$6h + HMAC$
[13]	$8h + 8M + 2HMAC$	$h + HMAC + M$
[20]	$PM + BP_PA$	$3BP + PM + M2P$
[29]	$6EXP$	$2BM + 5EXP$
[30]	$5PM + 2H2P$	$3BM + PM + H2P$
[31]	$4PM + H2P$	$2BM + PM + H2P$
[32]	$4PM + 2H2P$	$4BM + 2PM$
[33]	$4PM$	$4PM$
DAITS	$XOR + 2HMAC$	$HMAC$

during heavy messaging load. Specifically, we have measured the elapsed computation time for every message signing by a particular node and then calculated the mean value, which is found to be 0.042 ms. Similarly, the elapsed execution time for each message verification by the verifier has been measured; the average over all messages is found to be 0.034 ms. It can thus be observed that in terms of the individual authentication and verification rounds, DAITS outperforms all the competing schemes, except TESLA. However, DAITS is superior on a collective basis where the total overhead imposed on all nodes is much less than TESLA. Specifically, in DAITS, the computation cost of the message authentication is only paid once, i.e., by the designated sender's verifier, while in the other schemes, it is incurred repeatedly at every node that receives the message. Therefore, DAITS is indeed a cost-efficient solution for message authentication in ITS.

To further compare the efficiency of DAITS with the other approaches, Figures 7 reports the verification delay by every scheme under varying message counts in a regional road network, i.e., with one RSU. The delay in TESLA is greatly influenced by the message interval which is 100 ms in our settings. Meanwhile, PBA requires excessive computation as it involves sixty-one hash operations plus two HMAC operations for the signature generation while six hash operations and an HMAC are required for the verification. The scheme of [13] yields a slightly higher delay than DAITS due to use of HMAC, modulo, and hash operations needed for verifying a message while DAITS only uses a single HMAC. It can be observed that DAITS outperforms all other schemes by yielding minimum and stable delay. For instance, compared to other schemes, DAITS reduces the delay for 100 messages by at least 99%. Such a superiority is attributed to the fact that a verifier is not impacted by the increase in the number of messages. In other words, DAITS promotes instant message verification since the verifier is assigned to only a single source.

C. Verification Latency

To study the efficiency and scalability of our approach, we have tracked the end-to-end delay metric, which measures the elapsed time from the creation of an BSM by the sender, the verification by the assigned verifier, until the failure

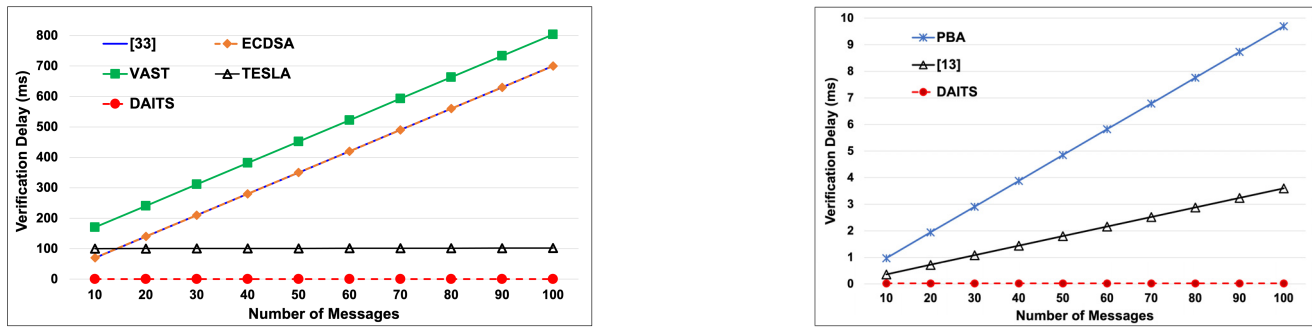


Fig. 7. Comparing DAITS to the baseline and competing schemes in terms of verification delay with varying number of received messages.

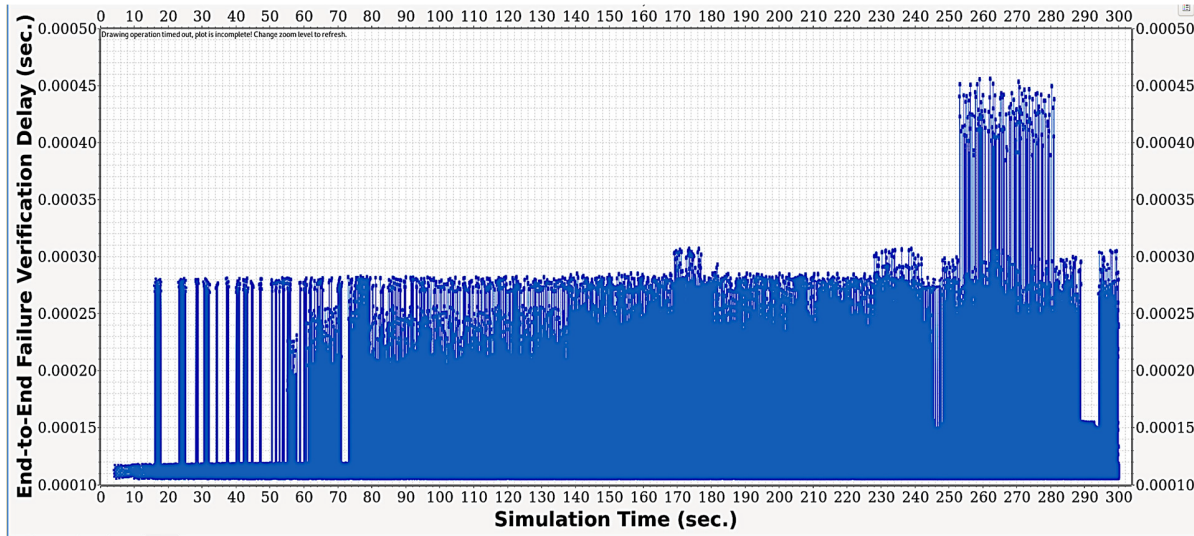


Fig. 8. The end-to-end delay performance for DAITS.

notification message has been verified by the receiver. In other words, this metric constitutes the following computation and communication delays: i) creation of security parameters by the sender, ii) node-to-verifier communication delay, iii) verification of the security parameters by the verifier, iv) generation of security parameters for failure notification message by the verifier, v) verifier-to-receivers' communication delay, and vi) verification of the security parameters by the receivers. Figure 8 shows the results of the comprehensive end-to-end metric for every failure notification received by a node during the simulation time. It can be observed that DAITS yields a stable end-to-end delay that is not affected by the increase in traffic density (number of the ITS nodes). We note that the number of nodes keeps growing at a rate of one vehicle every 3 seconds during the simulation time. Each of the initially inserted nodes has a total trip time between 251 – 275 sec.; thus, the heaviest traffic load is experienced between 250 to 280 of the simulation time where a delay of 0.45 ms is experienced during this period. By carefully tuning the timer for m_0 , i.e., m_{timer} , the ITS nodes would be able to ensure that no flag has been issued by the regional RSU regarding m_1 before dropping m_0 from the buffer. Based on the authentication latency of DAITS (Figure 8), we believe that a receiver can easily allocate a sufficient time before deciding to discard m_0 without causing message reception delay. Although the authentication of BSMs have been delegated to verifiers,

DAITS yields a negligible authentication latency, and hence, is deemed to be suitable for latency-critical ITS applications. Thus, DAITS can scale very well without a notable degradation in the performance.

VII. CONCLUSION

In ITS, there are three critical security challenges, namely, authentication, privacy, and secure message dissemination. Existing security solutions in the literature either address only a subset of these challenges or fail to handle the dynamic and heterogeneous nature of ITS. This paper has presented DAITS, an effective and cost-efficient solution that scales well for large deployments and suits the limited resources of ITS nodes. DAITS mitigates the limitations of PKI-based digital certificates by employing blockchain and smart contracts for handling node certificates and revocation in a more secure, distributed, and scalable way. We also rely on symmetric cryptography and orchestrate the assignment of a distinct key per node, without any elaborate system-wide key management process. Unlike node-based authentication approaches, the simulation results show that DAITS is cost-efficient and consistently yields good performance without being impacted by the increase in traffic density. Moreover, it significantly reduces the required infrastructure support to make ITS economically viable. Thus, DAITS would contribute to the

advancement of security and privacy in intelligent transportation systems. For future work, we plan to increase the level of autonomy and cooperation among the ITS nodes to further diminish the role of infrastructure.

REFERENCES

- [1] U.S. Dept. Transp. (2021). *Intelligent Transportation Systems—Benefits of Intelligent Transportation Systems Fact Sheet*. Accessed: Nov. 9, 2021. [Online]. Available: https://www.its.dot.gov/factsheets/benefits_factsheet.htm
- [2] A. Gharaibeh et al., “Smart cities: A survey on data management, security, and enabling technologies,” *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2456–2501, 4th Quart., 2017.
- [3] G. Karagiannis et al., “Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions,” *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 584–616, 4th Quart., 2011.
- [4] P. K. Singh, S. K. Nandi, and S. Nandi, “A tutorial survey on vehicular communication state of the art, and future research directions,” *Veh. Commun.*, vol. 18, Aug. 2019, Art. no. 100164.
- [5] J. He et al., “Cooperative connected autonomous vehicles (CAV): Research, applications and challenges,” in *Proc. IEEE 27th Int. Conf. Neww. Protocols (ICNP)*, Oct. 2019, pp. 1–6.
- [6] D. Manivannan, S. S. Moni, and S. Zeadally, “Secure authentication and privacy-preserving techniques in Vehicular Ad Hoc NETWORKS (VANETs),” *Veh. Commun.*, vol. 25, 2020, pp. 1–18.
- [7] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, “Authentication strategies in vehicular communications: A taxonomy and framework,” *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, p. 129, Dec. 2021.
- [8] D. Johnson, A. Menezes, and S. Vanstone, “The elliptic curve digital signature algorithm (ECDSA),” *Int. J. Inf. Secur.*, vol. 1, no. 1, pp. 36–63, Aug. 2001.
- [9] M. A. R. Baee, L. Simpson, E. Foo, and J. Pieprzyk, “Broadcast authentication in latency-critical applications: On the efficiency of IEEE 1609.2,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11577–11587, Dec. 2019.
- [10] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, “The TESLA broadcast authentication protocol,” *RSA Cryptobytes*, vol. 5, no. 2, pp. 2–13, 2002.
- [11] A. Studer, F. Bai, B. Bellur, and A. Perrig, “Flexible, extensible, and efficient VANET authentication,” *J. Commun. Netw.*, vol. 11, no. 6, pp. 574–588, Dec. 2009.
- [12] S. A. Alfadhli, S. Lu, K. Chen, and M. Sebai, “MFSPV: A multi-factor secured and lightweight privacy-preserving authentication scheme for VANETs,” *IEEE Access*, vol. 8, pp. 142858–142874, 2020.
- [13] W. Hathal, H. Cruickshank, Z. Sun, and C. Maple, “Certificateless and lightweight authentication scheme for vehicular communication networks,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16110–16125, Dec. 2020.
- [14] B. Palaniswamy, S. Camtepe, E. Foo, L. Simpson, M. A. R. Baee, and J. Pieprzyk, “Continuous authentication for VANET,” *Veh. Commun.*, vol. 25, Oct. 2020, Art. no. 100255.
- [15] Y. Xie, S. Zhang, X. Li, and Y. Li, “An efficient cooperative message authentication based on reputation mechanism in vehicular ad hoc networks,” *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 6, Jun. 2019, Art. no. 155014771985491.
- [16] M. Yang, S. Wei, R. Jiang, F. Ali, and B. Yang, “Single-message-based cooperative authentication scheme for intelligent transportation systems,” *Comput. Electr. Eng.*, vol. 96, Dec. 2021, Art. no. 107390.
- [17] A. Sarencheh, M. R. Asaar, M. Salmasizadeh, and M. R. Aref, “An efficient cooperative message authentication scheme in vehicular ad-hoc networks,” in *Proc. 14th Int. ISC (Iranian Soc. Cryptol.) Conf. Inf. Secur. Cryptol. (ISCISC)*, Sep. 2017, pp. 111–118.
- [18] H. J. Jo, I. S. Kim, and D. H. Lee, “Reliable cooperative authentication for vehicular networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1065–1079, Apr. 2018.
- [19] I. Ali and F. Li, “An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in VANETs,” *Veh. Commun.*, vol. 22, Apr. 2020, Art. no. 100228.
- [20] M. Bayat, M. Pournaghi, M. Rahimi, and M. Barmshoori, “NERA: A new and efficient RSU based authentication scheme for VANETs,” *Wireless Netw.*, vol. 26, no. 5, pp. 3083–3098, Jun. 2019.
- [21] A. Lei et al., “A blockchain based certificate revocation scheme for vehicular communication systems,” *Future Gener. Comput. Syst.*, vol. 110, pp. 892–903, Sep. 2020.
- [22] Y. C. E. Adja, B. Hammi, A. Serhrouchni, and S. Zeadally, “A blockchain-based certificate revocation management and status verification system,” *Comput. Secur.*, vol. 104, May 2021, Art. no. 102209.
- [23] Q. Wang, D. Gao, C. H. Foh, H. Zhang, and V. C. M. Leung, “Decentralized CRL management for vehicular networks with permissioned blockchain,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 11408–11420, Nov. 2022.
- [24] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, 1996.
- [25] *The AVISPA Project*. Accessed: May16, 2022. [Online]. Available: https://www.ercim.eu/publication/Ercim_News/enw64/armando.html
- [26] C. Sommer, R. German, and F. Dressler, “Bidirectionally coupled network and road traffic simulation for improved IVC analysis,” *IEEE Trans. Mobile Comput.*, vol. 10, no. 1, pp. 3–15, Jan. 2011.
- [27] Crypto++ Library 8.6 | Free C++ Class Library Cryptograph. Schemes. *Crypto++ Library 8.6*. Accessed: Apr. 1, 2022. [Online]. Available: <https://www.cryptopp.com/>
- [28] C. Lyu, D. Gu, Y. Zeng, and P. Mohapatra, “PBA: Prediction-based authentication for vehicle-to-vehicle communications,” *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 1, pp. 71–83, Jan. 2016.
- [29] M. Azees, P. Vijayakumar, and L. J. Deboarh, “EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2467–2476, Sep. 2017.
- [30] S. Wang and N. Yao, “LIAP: A local identity-based anonymous message authentication protocol in VANETs,” *Comput. Commun.*, vol. 112, pp. 154–164, Nov. 2017.
- [31] S. M. Pournaghi, B. Zahednejad, M. Bayat, and Y. Farjami, “NECPA: A novel and efficient conditional privacy-preserving authentication scheme for VANET,” *Comput. Netw.*, vol. 134, pp. 78–92, Apr. 2018.
- [32] Q. Mei, H. Xiong, J. Chen, M. Yang, S. Kumari, and M. K. Khan, “Efficient certificateless aggregate signature with conditional privacy preservation in IoV,” *IEEE Syst. J.*, vol. 15, no. 1, pp. 245–256, Mar. 2021.
- [33] J. Qi, T. Gao, X. Deng, and C. Zhao, “A pseudonym-based certificateless privacy-preserving authentication scheme for VANETs,” *Veh. Commun.*, vol. 38, Dec. 2022, Art. no. 100535.



Abdulaziz Alshaeri (Graduate Student Member, IEEE) received the M.S. degree in computer science from George Washington University. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC), Baltimore, MD, USA. His research interests include security and privacy in the Internet of Things and vehicular communications.



Mohamed Younis (Fellow, IEEE) is currently a Professor with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County (UMBC), Baltimore, MD, USA. Before joining UMBC, he was with the Advanced Systems Technology Group, an aerospace electronic systems research and development organization of Honeywell International Inc., where he led multiple projects for building integrated fault-tolerant avionics and dependable computing infrastructure. He also participated in the development of the redundancy management system, which is a key component of the vehicle and mission computer for NASA's X-33 space launch vehicle. He has published over 350 technical papers in refereed conferences and journals. He has seven granted and three pending patents. His research interests include network architecture and protocols, the IoT, embedded systems, fault-tolerant computing, and secure communication. In addition, he serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences.