Authentication Framework with Enhanced Privacy and Batch Verifiable Message Sharing in VANETs

Sujash Naskar, *Member, IEEE,* Carlo Brunetta, *Member, IEEE,*
Tingting Zhang, *Member, IEEE,* Gerhard Hancke, *Fellow Member, IEEE,* Mikael Gidlund, *Senior Member, IEEE*

***Abstract*—Vehicular Ad Hoc Networks (VANETs) are the backbone for intelligent transport and enhanced passenger safety, but they face significant challenges related to authentication, security, and privacy. Existing distributed VANET authentication protocols struggle with issues like privacy preservation during vehicle handovers and inefficiency in the presence of a massive amount of verification to be made. This paper proposes a novel authentication framework designed to address these limitations. First, we introduce zero-knowledge guarantees for Vehicle-to-Infrastructure (V2I) authentication and improve anonymity and unlinkability in authentication by eliminating explicit vehicle handover, thus enhancing privacy. Second, we propose a batch-verifiable Vehicle-to-Vehicle (V2V) message-sharing method utilizing an elliptic curve digital signatures scheme (ECDSA*). Unlike others, we provide a complete computational and efficiency analysis of batch verification in the presence of faulty signatures. A formal security analysis and proven security in the Scyther security verification tool promise the security guarantees of our proposed scheme. A thorough efficiency analysis shows that our scheme can perform at least 5-times more V2I authentication and can batch verify at least 2-times more V2V messages than other related schemes within a time threshold of 300ms.**

***Index Terms*—Vehicular Ad Hoc Networks (VANETs), Privacy Preservation, Authentication Protocol, Batch Verification.**

I. Introduction

THE Vehicular Ad Hoc Networks (VANETs) are interconnected networks of vehicles and infrastructure that facilitate advanced traffic management, support autonomous driving, and enhance passenger safety. Despite its merits, one of the critical challenges in VANETs is to develop efficient authentication techniques that guarantee secure and reliable vehicular communication while dealing with high mobility, time-critical responses, and network security vulnerabilities. Authentication protocols in VANETs are crucial for addressing security and privacy issues inherent in public-channel vehicular communications. These protocols [1] typically fall into two categories: centralized authentication with a single certification authority (CA) [2], and distributed authentication with multiple interconnected CAs [3]. The latter offers significant advantages, as discussed in [4, 5], including improved scalability, reliability, fault tolerance, robustness, and the elimination of single points of failure, particularly in regions with high vehicle density.

Vehicular communications can be broadly categorized into Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) interactions [6]. In V2I communication, vehicles authenticate with trusted verifiers governed by CA and obtain session-specific keys, enabling them to access local traffic information (LTI) from local authorities [7]. This authentication must be anonymous and unlinkable, ensuring the vehicle's identity and location-based privacy is preserved from adversaries. In decentralized VANETs, when vehicles transit between CA regions, an explicit handover protocol is required to be performed before the vehicle can request authentication to the local verifier [8]. This handover is necessary to delegate the authentication credentials of the vehicle to a new verifier in a new CA region. However, traditional handover requests can compromise privacy by revealing movement patterns to external potential adversaries and the verifier itself. Therefore, developing a V2I authenticated key-sharing scheme that eliminates the need for explicit handover requests can significantly enhance a vehicle's privacy.

In V2V communication, vehicles exchange sensitive driving assistance information (DAI) such as speed, turning, braking, and emergency alerts with nearby vehicles, ensuring the legitimacy of the information [9]. In traffic-congested metropolitan areas, this can result in hundreds of vehicles simultaneously generating and broadcasting DAI. A receiver vehicle must check each received message's authenticity, confidentiality, integrity, and freshness before accepting it as valid. According to existing research, each transmitted message in VANET must be verified within approximately 300 milliseconds (ms) [10] from the time a sender transmits it to the network. This defines a threshold for the lifespan of a transmitted message in the network, after which the message is no longer valid. Consequently, receiving vehicles must perform numerous verifications within a short time frame, as multiple senders can simultaneously transmit DAI messages into the network. Individually verifying all received messages before they expire can be significantly challenging with a standard single verification method. One cryptographic solution to this challenge proposed in the literature is to use a batch verification technique where a batch containing a set of messages can be verified with a single verification. Therefore, batch verification of message signatures can substantially improve efficiency by reducing the number of verifications needed within a given time frame [11].

While batch verification methods have been widely discussed and adopted in VANET literature, they often assume that all messages within a batch are error-free and correct. This assumption is impractical as message signatures can be corrupted due to transmission errors, random noise, adversarial interference, or the injection of faulty signatures. The presence of faulty message signatures in a batch is unavoidable and can

S. Naskar, T. Zhang, M. Gidlund are with the Department of Computer and Electrical Engineering, Mid Sweden University, 851 70 Sundsvall, Sweden (e-mail: {sujash.naskar, tingting.zhang, mikael.gidlund}@miun.se).

G Hancke is with the Department of Computer Science, City University of Hong Kong, China (e-mail: gp.hancke@cityu.edu.hk)

C Brunetta was in Simula UiB, Bergen, 5006, Norway. He is currently an Independent Researcher in Paris, France (e-mail: carlob@simula.no).

increase both the number and time required for verifications [12]. Cryptographic batch-verification algorithms are generally unable to pinpoint the location of faulty messages within a batch if a verification fails. Therefore, if a batch contains L messages with f faulty messages randomly distributed, the efficiency of existing batch verification methods remains uncertain. Although existing research in other application domains [13, 14] addresses this challenge, these approaches are not effectively designed to meet the stringent privacy and security requirements of VANETs. In contrast, the batch verification scheme proposed in [15] identifies invalid signatures by individually verifying each message in a batch when batch verification fails. However, this strategy can be inefficient, as the presence of a single invalid signature in a batch forces the verifier to resort to individual verification for all messages in the batch. To our knowledge, no current batch verification schemes in VANETs account for the presence of randomly distributed faulty messages within each batch. Also, the existing batch verification techniques mostly follow a bilinear-pairing-based batch verification method, which requires significantly high computation time and resources, making them inefficient.

Inspired by the potential to advance VANET authentication with stronger security and privacy, we proposed a novel V2I authenticated key-sharing scheme without explicit handover in this paper. Alongside, to overcome the batch-verification limitations of current schemes, a novel V2V batch-verifiable message-sharing method is proposed using lightweight elliptic curve cryptography (ECC) and considering the presence of randomly distributed faulty messages in each batch.

A. *Contributions*

Addressing the privacy and security challenges and the limitations present in current schemes, our proposed scheme significantly improves the authentication efficiency in VANETs with the following contributions:

- **Enhanced Privacy in V2I Communication**: Our V2I authenticated key-sharing scheme uses lightweight elliptic curve-based non-interactive-zero-knowledge proves (NIZK) [16] to eliminate explicit handover requests, protecting vehicle travel patterns and identities. This ensures anonymous and unlinkable V2I authentication, safeguarding vehicle privacy even from semi-trusted verifiers.
- **Optimized Authentication with Revocation**: Our scheme executes authenticated key sharing between a vehicle and an infrastructure unit in $< 2ms$, significantly faster than existing methods. Additionally, it is designed to promptly revoke malicious vehicles upon dispute reports, preventing the spread of malicious messages in the network.
- **Handling Faulty Signatures in V2V Batch Verification**: Proposed V2V batch verification method uniquely accounts for faulty message signatures, reflecting real-world VANET deployment. By randomizing the distribution of faulty messages, we provide a comprehensive performance analysis, unlike previous schemes that assume error-free messages and signature accuracy.
- **Comprehensive Batch Efficiency**: Our analysis of batch efficiency, including the impact of faulty signatures, shows that the best performance occurs with $< 50\%$ faulty signatures contiguously distributed in a batch, and the worst performance when faulty signatures are $\geq 50\%$. Using the linearity property of $\mathsf{ECDSA}*$ batch-verification [17] and precomputation outsourcing, our method ensures verification time does not exponentially increase with batch failures, maintaining complexity $\leq \mathcal{O}(L)$.

B. *Paper Organization*

Section II review related schemes, discussing their advantages and limitations. Section III outlines the system model, security and adversarial assumptions, and the cryptographic protocols used in the proposed scheme presented in Section IV. Section V exclusively analyzes the efficiency of the proposed batch verification scheme. Section VI provides formal and informal security analyses. Section VII presents performance comparisons. Finally, Section VIII concludes with discussions and insights.

II. Related Works

Yadav et al. [6] proposed a V2I authenticated key sharing scheme in a fog-based centralized VANET. Using the ECC-based certification method, the scheme is computationally lightweight. Other very similar authentication schemes proposed by Li et al. [3], Liu et al. [18] and Zhou et al. [19] also follows ECC-based certification. However, all these schemes require an iterative registration process for all vehicles, fog nodes, and roadside units (RSUs) to acquire certificates from a central CA at all times. A V2I authenticated key-sharing scheme is proposed by Tahir et al. [2] using lightweight hash functions and boolean xor operations. The authentication process is centralized and the protocol is vulnerable to reply attacks and location-based privacy attacks. Chen et al. [20], Shen et al. [21], Feng et al. [22], and Sikarwar [23] has proposed bilinear pairing-based distributed authentication scheme that can facilitate V2I or V2V communication. Allowing authentication requests to get batch verified, these schemes have tried to make authentication faster. However, they have assumed the presence of error-free messages and signatures in each batch, which makes their batch verification scheme non-realistic for public channel communication where an error in transmission is unavoidable. Another batch verification scheme proposed by Yan et al. [15] uses ECC-based certificates. However, their verification time increases exponentially if a batch contains an invalid certificate. Other distributed trust-based authentication mechanisms proposed by Wang et al. [24], Zhang et al. [9], Ma et al. [25], Inedjaren et al. [26] and A. Ghaleb et al. [27] uses a reputation mechanism with a blockchain-based trust table for each surrounding vehicle. Even though these protocols facilitate direct message sharing between vehicles, they are computationally heavy and can not facilitate batch verification of received messages. Table I compares various authentication schemes previously proposed in the literature. Many existing schemes focus solely on authenticated V2V communication without specifying how vehicles will communicate with infrastructure units (V2I). To achieve batch verifiability, these schemes often rely on resource-intensive

TABLE I: Comparison of VANET authentication schemes.

Schemes	V2I+V2V	Batch	Revoke	Light	Distributed
Yadav et al. [6]	✗	✗	✗	✓	✓
Naskar et al. [5]	✓	✗	✓	✓	✓
Bayat et al. [28]	✓	✗	✓	✗	✓
Wang et al. [24]	✗	✗	✗	✗	✓
Sikarwar [23]	✗	✓	✗	✗	✓
Yan et al. [15]	✗	✓	✗	✗	✗
Chen et al. [20]	✗	✓	✗	✗	✓
Shen et al. [21]	✓	✓	✗	✗	✗
Feng et al. [22]	✓	✓	✗	✗	✓
Our Scheme	✓	✓	✓	✓	✓

TABLE II: Notation Descriptions used in the Scheme.

Notations	Definitions
$\leftarrow$	Deterministic assignment
$\leftarrow\$$	Probabilistic assignment
$x \leftarrow \mathsf{Alg}$	Executes Alg and returns x
$(x, y, z) \rightarrow \Delta$	Stores (x, y, z) into the database
$y \leftarrow \Delta(x)$	Load entry indexed by x from the database
$x \in_? \Delta$	Checks if Δ contains an entry with index x
$\mathsf{H}(data)$	General hash function
$\mathsf{H}_s(data)$	Seeded hash function.
$\mathsf{E}(\mathsf{k}, \mathsf{m}), \mathsf{D}(\mathsf{k}, \mathsf{ct})$	AES Encryption and Decryption.
pp	Set of public-parameters
G, H	Generators of elliptic curve group $\mathbb{G}$
$\vec{S_i}$	NIZK Statement
$(\vec{R}, \pi)$	NIZK prove for state meant $\vec{S_i}$
m, ct	Original message, encrypted message
σ	An $\mathsf{ECDSA}*$ signature
S	A sender Vehicle
R	A receiver vehicle
s_i	Secret of vehicle shared with parent CA
h_i	Hashed value of secret s_i shared with LI.
i	value selected from set of indices R
ρ	Auxiliary information
t	Timestamp
$\mathcal{R}$	Dispute report request
ek	An epoch key
$\mathsf{pk_{CA}}, \mathsf{sk_{CA}}$	Public and private key of a CA
$\mathsf{pk}_{\mathcal{V}}, \mathsf{sk}_{\mathcal{V}}$	Long term public and private key of a vehicle
pk, sk	Short-term public and private key of a vehicle
$\mathsf{pk_{LI}}, \mathsf{sk_{LI}}$	Public and private key of an LI

bilinear pairing cryptography, compromising their lightweight nature. Additionally, most lack a revocation strategy to prevent malicious vehicles from being part of an ongoing communication. In contrast, while remaining lightweight and distributed, the proposed framework in this paper addresses all essential VANET communication needs, such as V2V, V2I, batch verifiability, and revocation.

III. Preliminaries

We adapt the distributed VANET system model proposed by Naskar et al. [5]. All the notations used in the proposed scheme are presented with descriptions in Table II.

A. Communication Model

As presented in Figure 1, the system consists of the following scheme entities:

- **Certification Authorities**, or CAs are trusted authorities responsible for deciding the initial security parameters and registering vehicles with their original identities [22].

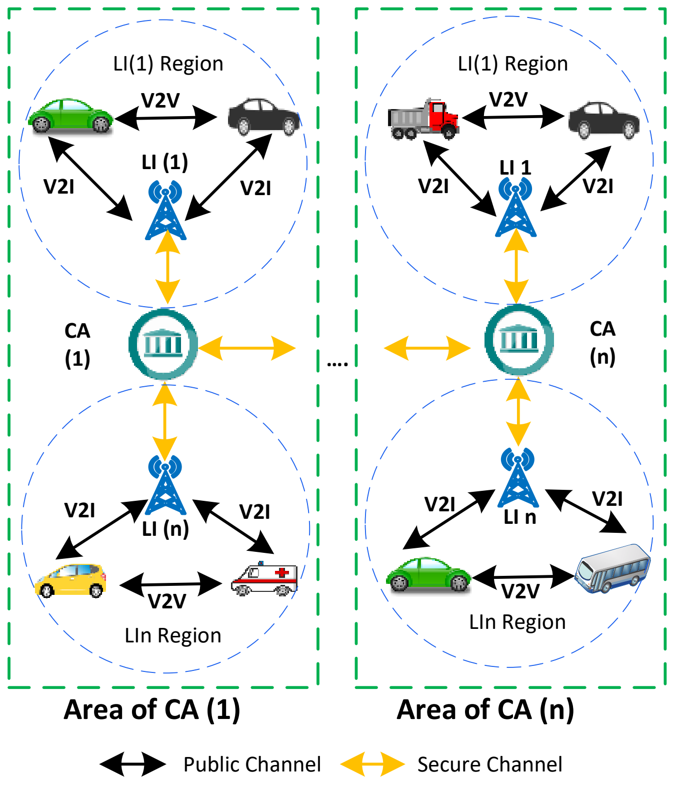

Fig. 1: Distributed VANET Communication Model

- **Local Inspectors** or LIs are locally placed units connected to a CA. Each LI communicates with its CA to facilitate authenticated key sharing and revocation locally [5].
- **Vehicles**, are equipped with tamper-protected On-Board-Unit (OBU) [21] for all security computations and communications. Each vehicle has a parent CA to which it gets registered initially and communicates with LI and other vehicles to acquire LTI and DAI.
- **Communication Channel**, are typically a dedicated short-range communication or DSRC such as the IEEE 802.11p, IEEE 802.11px or C-V2X [29]. We assume vehicles communicate with LI using the public channel, whereas an LI communicates with its CA using an already established, authenticated, secure channel. This assumption is practical because CAs and LIs are trusted or semi-trusted entities.

Following the decentralized VANET model (Figure 1), the CAs provide authentication services for a large geographic area. Each CA communicates with multiple connected LIs that provide verification services in comparatively small areas within the CA's region. Vehicles receive LTI information from the current local LI and communicate with other surrounding vehicles by sharing authenticated messages. Vehicles can report disputes to the local LI, which can revoke the disputed vehicle and prevent the propagation of malicious messages in the network.

Similar to a session, an epoch is a short time frame with a specified starting and ending time decided by an LI for its communication region. In each epoch, vehicles get verified by the LI and receive an epoch key ek securely. Epoch keys from different epoch times are unlikable, maintaining forward and backward secrecy of epochs.

B. Security Assumptions

We follow the well-established VANET security model, widely adopted by the community [5, 30, 22, 26]. The security requirements are:

- **Privacy Guarantees**: The scheme must ensure a vehicle user's identity and location-based privacy is preserved during all communications. Therefore, authentication must be unlinkable and anonymous.
- **Preventing Modification Attacks**: To ensure the integrity and confidentiality of messages, the scheme must protect the communication from modification attacks.
- **No Replay Attacks**: The legitimacy and freshness of each authentication message need to be checked by a verifier to eliminate the possibility of a reply attack.
- **MitM Free**: An adversary place itself in between two communicating parties to establish a secure channel without the party noticing its malicious presence.
- **No Impersonation**: Each entity in the scheme has a unique identifier that must be validated to ensure the authenticity of entities in communication.
- **Preventing Forgery Attacks**: The scheme must ensure that batch-verification of vehicle's certificates and message signatures are not vulnerable to signature forgery attacks.
- **Sybil Free**: Each entity in the network must have only one instance at any specific time.
- **Non-Repudiation**: A sender can not deny sending a message, and a receiver can not deny receiving it; this ensures non-repudiation in the network.
- **Minimized DoS Attacks**: Even though Denial-of-service is unavoidable in public channels, it is possible to minimize its impact on the network. The verification process must ensure that if a significantly high number of false messages arrives, it can quickly identify and discard them at an early verification stage.

C. Adversarial Assumptions

Following the same adversarial model mentioned by other related works [5, 31], i.e. the adversary in our scheme, represented as $\mathcal{A}$ can be an external unauthorized entity (external adversary) or an internal dishonest entity (internal adversary) performing several security and privacy attacks. We assume that $\mathcal{A}$ follows the Dolev-Yao adversarial model [32], having comprehensive knowledge of the network structure, communications, and significant computational power. Following the model, $\mathcal{A}$ can:

- **Intercept and Analyze**: $\mathcal{A}$ can intercept any message from the public network and analyze it for possible valuable information that can help $\mathcal{A}$ to find privacy and security vulnerabilities in the scheme.
- **Modification and Injection**: $\mathcal{A}$ can partially or completely modify an intercepted message and inject it into the communication flow to achieve unauthorized access to the network.
- **Impersonation and spoofing**: $\mathcal{A}$ can impersonate any legitimate entity in the network to manipulate the communication flow and inject malicious commands.
- **Protocol State Manipulation**: $\mathcal{A}$ can manipulate the designed protocol states to disrupt the expected sequence messages flow.

An unauthorized entity can perform all possible passive and active attacks in the public wireless networks. Vehicles in our proposed scheme are untrusted entities that can perform privacy theft or security attacks such as modification, replay, impersonation, and repudiation. The LIs are semi-trusted entities as they are assumed not to perform security attacks. However, they can be curious to learn about the original identities of vehicles and their travel pattern from one LI to another. However, the OBU of vehicles are tamper-protected units, and it is assumed that all the secret information stored in OBU can not be accessed or shared externally.

D. Cryptographic Primitives

All the cryptographic functions used in our authentication scheme are presented as algorithms in Figure 2.

Authenticated Encryption Scheme. For our scheme, we consider a secure authenticated symmetric encryption scheme (E, D) such as AES in Galois counter-mode Such a scheme encrypts a message m under a secret key k to obtain $\mathsf{ct} \leftarrow \mathsf{E}(k, \mathsf{m})$ which effectively contains a ciphertext plus an authentication tag. The decryption algorithm $\mathsf{D}(k, \mathsf{ct})$ will output the original message if the authentication tag is correctly verified otherwise the decryption fails, i.e. the decryption of the ciphertext might be possible but the provided tag is not valid, similar to a bad signature.

Chaum-Pedersen NIZK. The Chaum-Pedersen commitment scheme [33], together with the Fiat-Shamir heuristic [34], is used to design non-interactive-zero-knowledge (NIZK) prove protocols. The commitment scheme allows a prover to convince a verifier that a certain statement is true without revealing any additional information. The Fiat-Shamir heuristic makes the statement proof non-interactive by generating the coin flip challenge c as the output of a hash function. The Chaum-Pedersen NIZK proof using an elliptic curve proves that two elliptic curve points share the same discrete logarithm with respect to two different bases without revealing the discrete logarithm itself. As presented in Figure 2, the algorithm $\mathsf{NIZKGen}$ first selects the elliptic-curve group $\mathbb{G}$ of prime order p with generators G and H. The algorithm $\mathsf{Prove}_\delta(\mathsf{pp}, \vec{S}, s)$ generates a randomized prove statement $\vec{R}, \pi$ and finally the algorithm $\mathsf{NIZKVer}_\delta(\mathsf{pp}, \vec{R}, \vec{S}, \pi)$ verifies the prove.

DH-Key Generation. Using elliptic-curve cryptography (ECC), first a key pair $(\mathsf{sk}, \mathsf{pk})$ is generated using $\mathsf{KGen}(\mathsf{pp})$ after initializing the curve parameters Init. Then, the Diffie-Hellman (DH) [35] symmetric shared key between two communicating entities is generated using $\mathsf{KA}(\mathsf{sk}, \mathsf{pk})$ algorithm where sk and pk are from different entities.

$\mathsf{ECDSA}*$ *Signatures.* The $\mathsf{ECDSA}*$ algorithm is a variety of Elliptic Curve Digital Signature Algorithm (ECDSA) [36], that

```
$\mathsf{Init}$
$\mathsf{pp} \leftarrow (\mathbb{G}, p, G)$
**return** $\mathsf{pp}$

$\mathsf{KGen}(\mathsf{pp})$
$r \leftarrow\!\!\$\; \mathbb{Z}_p$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow (r, r \cdot \mathsf{pp}.G)$
**return** $(\mathsf{sk}, \mathsf{pk})$

$\mathsf{KA}(\mathsf{sk}, \mathsf{pk})$
**return** $\mathsf{sk} \cdot \mathsf{pk}$

$\mathsf{NIZKGen}$
$\mathsf{pp} \leftarrow (\mathbb{G}, p, G, H)$
**return** $\mathsf{pp}$

$\mathsf{Prove}_\delta(\mathsf{pp}, \vec{S}, s)$
$r \leftarrow\!\!\$\; \mathbb{Z}_p$
$\vec{R} \leftarrow (r \cdot \mathsf{pp}.G, r \cdot \mathsf{pp}.H)$
$c \leftarrow \mathsf{H}(\mathsf{pp}, \vec{R}, \vec{S}, \delta)$
$\pi \leftarrow r - s \cdot c$
**return** $(\vec{R}, \pi)$

$\mathsf{NIZKVer}_\delta(\mathsf{pp}, \vec{R}, \vec{S}, \pi)$
$c \leftarrow \mathsf{H}(\mathsf{pp}, \vec{R}, \vec{S}, \delta)$
**return true if**
    $\begin{cases} R_1 - c \cdot S_1 \stackrel{?}{=} \pi \cdot \mathsf{pp}.G \\ R_2 - c \cdot S_2 \stackrel{?}{=} \pi \cdot \mathsf{pp}.H \end{cases}$

$\mathsf{SignInit}(l, T)$
$\mathsf{pp} \leftarrow (\mathbb{G}, p, G)$
**for** $i \in \{1, \ldots, T\}$ **do**
        / Pairwise co-prime
    $b_i \leftarrow\!\!\$\; \mathbb{Z}_p$
**return** $(\mathsf{pp}, b_I)$

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m})$
$r \leftarrow\!\!\$\; \{2, \ldots, p-2\}$
$R \leftarrow r \cdot \mathsf{pp}.G$
$x \leftarrow x(R)$
$c \leftarrow r^{-1}(\mathsf{H}(\mathsf{m}) + \mathsf{sk} \cdot x)$
$\sigma \leftarrow (R, c)$
**return** $\sigma$

$\mathsf{BatchVer}((\mathsf{pp}, b_I), \mathsf{pk}, \sigma_I, \mathsf{m}_I)$
**for** $i \in I$ **do**
    $x_i \leftarrow x(R_i)$
    $w_i \leftarrow c_i^{-1}$
    $u_i \leftarrow \mathsf{H}(\mathsf{m}_i) \cdot w_i$
    $v_i \leftarrow x_i \cdot w_i$
$\overline{R} \leftarrow \sum_{i \in I} b_i \cdot R_i$
$\overline{U} \leftarrow \left(\sum_{i \in I} b_i u_i\right) \cdot \mathsf{pp}.G$
$\overline{V} \leftarrow \left(\sum_{i \in I} b_i v_i\right) \cdot \mathsf{pk}$
**return** $\overline{R} \stackrel{?}{=} \overline{U} + \overline{V}$

$\mathsf{Ver}(\mathsf{pp}, \mathsf{pk}, (R, c), \mathsf{m})$
$x \leftarrow x(R)$
$w \leftarrow c^{-1}$
$u \leftarrow \mathsf{H}(\mathsf{m}) \cdot w$
$v \leftarrow x \cdot w$
$\overline{R} \leftarrow u \cdot \mathsf{pp}.G + v \cdot \mathsf{pk}$
**return** $x(\overline{R}) \stackrel{?}{=} x$
```

Fig. 2: Algorithms for the Chaum-Pedersen ZK protocol using Fiat-Shamir transformation, Diffie-Hellman key-agreement protocol and ECDSA$*$ signature scheme, with key generation shared with DH, plus the batch verification algorithm.

uses the pairwise relative co-primes b_i. After initialized by the $\mathsf{SignInit}(l, T)$ algorithm, it generates a verifiable signature σ on a message m using the $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{m})$ algorithm. Unlike ECDSA, where a signature consists of an elliptic curve point and only the x-coordinate of another, ECDSA$*$ signatures consist of two elliptic curve points with all coordinates. Standard single verification of a signature is performed using $\mathsf{Ver}(\mathsf{pp}, \mathsf{pk}, (R, c), \mathsf{m})$ algorithm.

ECDSA$*$ *Batch Verification.* We have adapted the ECDSA$*$ batch verification algorithm proposed by Kittur and Pais [17] for its efficient linearity in verification considering the presence of faulty messages in each batch. The ECDSA$*$ signatures can be batch-verified for an efficient verification time. Following the algorithm $\mathsf{BatchVer}((\mathsf{pp}, b_I), \mathsf{pk}_I, \sigma_I, \mathsf{m}_I)$ presented in Figure 2, all signatures in a batch of $i \in \{1, \ldots, L\}$ is first precomputed to generate $\widehat{R}, \widehat{U}, \widehat{V}$ and finally the check $\overline{R} \stackrel{?}{=} \overline{U} + \overline{V}$ ensures the correctness of the batch verification.

IV. Proposed Authentication Scheme

This section provides a complete presentation of our scheme. We provide the formal algorithm definition in Figure 3 and discuss the high-level design rationale throughout the section. The proposed authentication scheme consists of five phases: initialization of the parameters, registration to CA, authentication of V, V2I/V2V communication and revocation procedure.

Initialization. The CAs initialize the scheme by sampling the public parameters $\mathsf{pp}_{\mathsf{CA}}$ which identify the elliptic curve group $\mathbb{G}$ and generators G and H. Finally, each CA generates a key-pair $\mathsf{sk}_{\mathsf{CA}}, \mathsf{pk}_{\mathsf{CA}}$ used for signing LIs certificates and for direct secure communication from users to the correspondent CA.

Registration. During the registration phase, all the parties register with their region-specific certification authority, thus creating a network of multiple CAs (e.g. CA and $\widehat{\mathsf{CA}}$) able to securely communicate with each other. Both the LI and the user V registration processes are assumed to be performed in a trusted environment where the CA can assure the lawful correctness of the registration data.

The $\widehat{\mathsf{CA}}$ registers LI by providing a certificate σ_{LI} that contains information regarding LI's public key $\mathsf{pk}_{\mathsf{LI}}$, auxiliary information like its deployment location ρ. The reason is that the certificate allows a connecting party to verify the legitimacy of LI which is necessary for authenticating the communication between V and LI.

The registration process for the vehicle V aims at establishing a set of shared secrets used as hints for the authentication of V to LI. Namely, V registers its information (VI, UI) together with a generated public key pk_{V} which V must prove to CA it knows the related secret key sk_{V} using the NIZK primitive. If this is the case, CA provides several shared secrets (s, R) used for generating NIZK authentication proofs and a database Δ_{V} containing different CAs (e.g. $\widehat{\mathsf{CA}}$) trusted by CA.

Authentication. The authentication procedure aims at authenticating V against LI with the help of the certification authorities. A high-level diagram of the authentication procedure is provided in Figure 4. Assume the vehicle V is registered with CA and LI is registered to $\widehat{\mathsf{CA}}$ and V is currently in LI's locality and desires to authenticate after receiving LI's broadcasted public key and certificate.

After verifying the validity of LI's certificate and the trustworthiness of $\widehat{\mathsf{CA}}$, the vehicle V generates an ephemeral key-pair and an authentication NIZK proof π using randomly sampled values from the shared secrets (s, R). Together with these values, V appends the ciphertext ct_c which contains an identifier for the correct authority CA, decryptable only by $\widehat{\mathsf{CA}}$, and ct_x containing an identifier to V's information and obtainable only by CA and sends them to LI.

After verifying the correctness of the received data, LI is tasked to authenticate V by verifying the NIZK proof of which statement $(\mathsf{pp}, \vec{S}_i)$ might be unknown if not previously stored in the database $\Delta_{\mathsf{LI}}(h_i)$. In such a scenario, LI queries its certification authority $\widehat{\mathsf{CA}}$ regarding the challenge h_i and provides $(\mathsf{ct}_c, \mathsf{ct}_x)$ which are used to identify the correct authority CA and to reconstruct the statement $(\mathsf{pp}, \vec{S}_i)$.

In fact, $\widehat{\mathsf{CA}}$ decrypts and verifies if it knows CA and forwards the statement request to them. CA can decrypt ct_x which provides enough information to retrieve the secret

```
Registration of LI to Authority $\widehat{CA}$
After $\widehat{CA}$ verifies LI's legitimacy
$\widehat{CA} \to LI : pp_{\widehat{CA}}, pk_{\widehat{CA}}$
    LI exec $(pp_{\widehat{CA}}, pk_{CA}, \rho)$
    $(sk_{LI}, pk_{LI}) \leftarrow\$ KGen(pp_{CA})$
    $\sigma \leftarrow\$ Sign(pp_{\widehat{CA}}, sk_{LI}, (pk_{\widehat{CA}}, pk_{LI}, \rho))$
$LI \to \widehat{CA} : (pk_{LI}, \rho, \sigma)$
    $\widehat{CA}$ exec $(pp_{\widehat{CA}}, sk_{\widehat{CA}}, pk_{LI}, \rho, \sigma)$
    $Ver(pp_{\widehat{CA}}, pk_{LI}, \sigma, (pk_{\widehat{CA}}, pk_{LI}, \rho))$
    $\sigma_{LI} \leftarrow\$ Sign(pp_{\widehat{CA}}, sk_{\widehat{CA}}, (pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho))$
$\widehat{CA} \to LI : \sigma_{LI}$
$LI : Ver(pp_{\widehat{CA}}, pk_{\widehat{CA}}, \sigma_{LI}, (pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho))$
$LI$ : store $(pp_{\widehat{CA}}, sk_{LI}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI})$
$CA : (pk_{LI}, \rho, \sigma_{LI}) \to \Delta_{\widehat{CA}}$
```

```
Revocation req. R $(m, pk_S, \sigma_S, t, \sigma_m)$
load $(pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI}, \epsilon, ek, sk_R, pk_R, \sigma_R)$
$R \leftarrow (m, pk_S, \sigma_S, t, \sigma_m, pk_R, \sigma_R)$
$\sigma \leftarrow\$ Sign(pp_{\widehat{CA}}, sk_R, R)$
$k_{LI} \leftarrow KA(sk_R, pk_{LI})$
$ct \leftarrow E(k_{LI}, (R, \sigma))$
send $(pk_R, ct)$
```

```
V2V communication S send $(m, t)$
load $(pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI}, \epsilon, ek, sk_S, pk_S, \sigma_S)$
$\sigma_m \leftarrow\$ Sign(pp_{\widehat{CA}}, sk_S, (m, pk_S, t))$
$ct \leftarrow E(ek, (m, pk_S, \sigma_S, t, \sigma_m))$
send $(pk_S, ct)$
```

```
V2V communication R receive $(pk_{S_i}, ct_i)_I$
load $(pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI}, \epsilon, ek, sk_R, pk_R, \sigma_R)$
load $\Delta_{pk}$;   init $\Delta, \Delta_m$
for $i \in I$ do
    $(m_i, pk_{S_i}, \sigma_{S_i}, t_i, \sigma_i]) \leftarrow D(ek, ct_i)$
    if $t_i$ invalid then skip
    if $m_i$ meaningless then skip
    if $(pk_{S_i}, \sigma_{S_i}) \in \Delta_{pk}$ then $(m_i, pk_{S_i}, t_i, \sigma_i) \to \Delta_m$
    else $(pk_{S_i}, \sigma_{S_i}) \to \Delta$
$BatchVer(pp_{\widehat{CA}}, pk_{LI}, (\sigma_{S_i})_{i\in\Delta}, (pk_{S_i}, pk_{LI}, \epsilon)_{i\in\Delta})$
for $(pk_{S_i}, \dots) \in \Delta$ then $(m_i, pk_{S_i}, t_i, \sigma_i) \to \Delta_m$
for $pk_{S_i} \in \Delta_m$ then
    $\Delta_i \leftarrow \Delta_m(pk_{S_i})$
    $BatchVer(pp_{\widehat{CA}}, pk_{S_i}, (\sigma_{m_i})_{i\in\Delta_i}, (m_i, pk_{S_i}, t_i)_{i\in\Delta_i})$
return $\Delta_m$
```

```
Initialization of CA
init $\Delta_{CA}$ ;   $pp_{CA} \leftarrow\$ NIZKGen$
$(sk_{CA}, pk_{CA}) \leftarrow\$ KGen(pp_{CA})$
store $(pp_{CA}, sk_{CA}, pk_{CA})$
```

```
Registration of Vehicle (V) to Authority CA
After CA verifies V's legitimacy
$CA \to V : pp, pk_{CA}$
    V exec $(pp, pk_{CA}, UI, VI)$
    $(sk_V, pk_V) \leftarrow\$ KGen(pp)$
    $\eta \leftarrow sk_V \cdot pp.H$
    $(\vec{R}, \pi) \leftarrow\$ Prove(pp, (pk_V, \eta), sk_V)$
$V \to CA : (UI, VI, \vec{R}, (pk_V, \eta), \pi)$
    CA exec $(pp, UI, VI, (pk_V, \eta), \pi)$
    $NIZKVer(pp, \vec{R}, (pk_V, \eta), \pi)$
    init $\Delta_V$ ;   $y \leftarrow\$ \{0,1\}^*$
    $R \leftarrow\$ \mathcal{P}(\mathbb{Z}_p)$ s.t. $|R| = r$
    $s \leftarrow H(UI, VI, y)$
$CA \to V : (s, R), \Delta_V$
$V$ : store $(pp, pk_{CA}, sk_V, pk_V, s, R), \Delta_V$
$CA : (pp, pk_V, UI, VI, y, s, R) \to \Delta_{CA}$
```

```
Revocation req. LI $(pk_R, ct)$
$k_{LI} \leftarrow KA(sk_{LI}, pk_R)$
$(R, \sigma) \leftarrow D(k_{LI}, ct)$
$(m, pk_S, \sigma_S, t, \sigma_m, \widehat{pk}_R, \sigma_R) \leftarrow R$
if $pk_R \neq \widehat{pk}_R$ then abort
$Ver(pp_{\widehat{CA}}, pk_{LI}, \sigma_S, (pk_S, pk_{LI}, \epsilon))$
$Ver(pp_{\widehat{CA}}, pk_{LI}, \sigma_R, (pk_R, pk_{LI}, \epsilon))$
$Ver(pp_{\widehat{CA}}, pk_R, \sigma, R)$
$Ver(pp_{\widehat{CA}}, pk_S, \sigma_m, (m, pk_S, t))$
Act accordingly on $(m, t)$
```

```
Auth. LI input $(pk, ct)$
$LI$ : load $(pp_{\widehat{CA}}, sk_{LI}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI})$
$k_{LI} \leftarrow KA(sk_{LI}, pk)$
$(\widehat{pk}, (h_i, \vec{R}_i, \pi), ct_x, ct_c, t) \leftarrow D(k_{LI}, ct)$
if $t$ invalid or $\widehat{pk} \neq pk$ then abort
if $h_i \overset{?}{\in} \Delta_{LI}$ then $(pp, \vec{S}_i, \dots) \leftarrow \Delta_{LI}$
else $LI \to \widehat{CA} : (pk, pk_{LI}, h_i, ct_x, ct_c)$
    $\widehat{CA} \to LI : (pp, \vec{S}_i)$
$NIZKVer_t(pp, \vec{R}_i, \vec{S}_i, \pi)$
$\sigma \leftarrow\$ Sign(pp_{\widehat{CA}}, sk_{LI}, (pk, pk_{LI}, \epsilon))$
$\widehat{ct} \leftarrow E(k_{LI}, (\epsilon, ek, \sigma))$
$(h_i, pp, S_i, \pi, pk, \sigma) \to \Delta_{LI}$
send $\widehat{ct}$ to V
```

```
Auth. V input $(pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI})$
$V$ : load $(pp_{CA}, pk_{CA}, sk_V, s, R), \Delta_V$
if $H(pk_{\widehat{CA}}) \notin \Delta_V$ then abort
$Ver(pp_{\widehat{CA}}, pk_{\widehat{CA}}, \sigma_{LI}, (pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho))$
$i \leftarrow\$ R$;   $t$ timestamp
$s_i \leftarrow H_s(pk_{LI}, i)$
$h_i \leftarrow H(s_i)$
$\vec{S}_i \leftarrow (s_i \cdot pp_{CA}.G, s_i \cdot pp_{CA}.H)$
$(\vec{R}_i, \pi) \leftarrow\$ Prove_t(pp_{CA}, \vec{S}_i, s_i)$
$\sigma \leftarrow\$ Sign(pp_{CA}, sk_V, (pk_{CA}, i, H(pk_V)))$
$(sk, pk) \leftarrow\$ KGen(pp_{\widehat{CA}})$
    $k_{CA} \leftarrow KA(sk, pk_{CA})$
    $ct_x \leftarrow E(k_{CA}, (i, H(pk_V), \sigma))$
    $k_{\widehat{CA}} \leftarrow KA(sk, pk_{\widehat{CA}})$
    $ct_c \leftarrow E(k_{\widehat{CA}}, h_{pk_{CA}})$
    $k_{LI} \leftarrow KA(sk, pk_{LI})$
    $ct \leftarrow E(k_{LI}, (pk, (h_i, \vec{R}_i, \pi), ct_x, ct_c, t))$
send $(pk, ct) \to LI$
receive $\widehat{ct}$
$(\epsilon, ek, \sigma) \leftarrow D(k_{LI}, \widehat{ct})$
$Ver(pp_{\widehat{CA}}, pk_{LI}, \sigma, (pk, pk_{LI}, \epsilon))$
store $(pp_{\widehat{CA}}, pk_{LI}, pk_{\widehat{CA}}, \rho, \sigma_{LI}, \epsilon, ek, sk, pk, \sigma)$
```

```
Auth. $\widehat{CA}$ input $(pk, pk_{LI}, h_i, ct_x, ct_c)$
$\widehat{CA}$ : load $(pp_{\widehat{CA}}, sk_{\widehat{CA}}, pk_{\widehat{CA}}), \Delta_{\widehat{CA}}$
$k_{\widehat{CA}} \leftarrow KA(sk_{\widehat{CA}}, pk)$
$h_{pk_{CA}} \leftarrow D(k_{\widehat{CA}}, ct_c)$
if $h_{pk_{CA}}$ unknown then abort
if $\widehat{CA} = CA$ then
    exec $CA(pk, pk_{LI}, h_i, ct_x)$
else $\widehat{CA} \neq CA$ then
    $\widehat{CA} \to CA : (pk, pk_{LI}, h_i, ct_x)$
    $CA$ : exec $CA(pk, pk_{LI}, h_i, ct_x)$
    $CA \to \widehat{CA} : (pp, \vec{S}_i)$
return $(pp, \vec{S}_i)$
```

```
Procedure $CA(pk, pk_{LI}, h_i, ct_x)$
$CA$ : load $(pp_{CA}, sk_{CA}, pk_{CA})\Delta_{CA}$
$k_{CA} \leftarrow KA(sk_{CA}, pk)$
$(i, h_{pk_V}, \sigma) \leftarrow D(k_{CA}, ct_x)$
$(pp, pk_V, \dots, s, R) \leftarrow \Delta_{CA}(h_{pk_V})$
if $i \notin R$ then abort
$Ver(pp_{CA}, pk_V, \sigma, (pk_{CA}, i, h_{pk_V}))$
$s_i \leftarrow H_s(pk_{LI}, i)$
if $h_i \neq H(s_i)$ then abort
$\vec{S}_i \leftarrow (s_i \cdot pp.G, s_i \cdot pp.H)$
return $(pp, \vec{S}_i)$
```

Fig. 3: CA's initialization and registration procedures for both the users V and LI LI. Red texts represent aborts resolved by re-executing the authentication procedure. Authentication procedures for the user V involving LI, the authority $\widehat{CA}$ and the procedure when the correct authority is identified. Cyan texts represent aborts which terminate the authentication. Authenticated V2V (expandable to V2I) communication algorithms between a sender S and a receiver R. Yellow texts represent message filtering, i.e. the procedure drops the messages that do not pass the highlighted test. Revocation request from R to LI. Green texts denote possible protocol aborts. At the end of its procedure, LI acts according to the reported message (m, t).

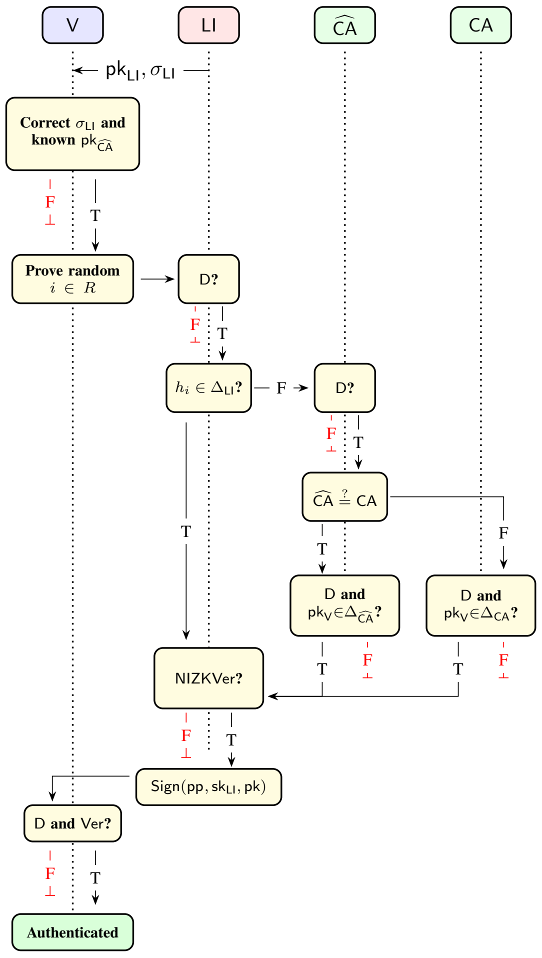
V
LI
$\widehat{\mathsf{CA}}$
CA
$\mathsf{pk}_{\mathsf{LI}}, \sigma_{\mathsf{LI}}$
Correct σ_{LI} and known $\mathsf{pk}_{\widehat{\mathsf{CA}}}$
Prove random $i \in R$
D?
$h_i \in \Delta_{\mathsf{LI}}$?
D?
$\widehat{\mathsf{CA}} \overset{?}{=} \mathsf{CA}$
D and $\mathsf{pk}_{\mathsf{V}} \in \Delta_{\widehat{\mathsf{CA}}}$?
D and $\mathsf{pk}_{\mathsf{V}} \in \Delta_{\mathsf{CA}}$?
NIZKVer?
Sign(pp, $\mathsf{sk}_{\mathsf{LI}}$, pk)
D and Ver?
Authenticated
T
F
$\perp$

Fig. 4: A high-level diagram of the authentication flow.

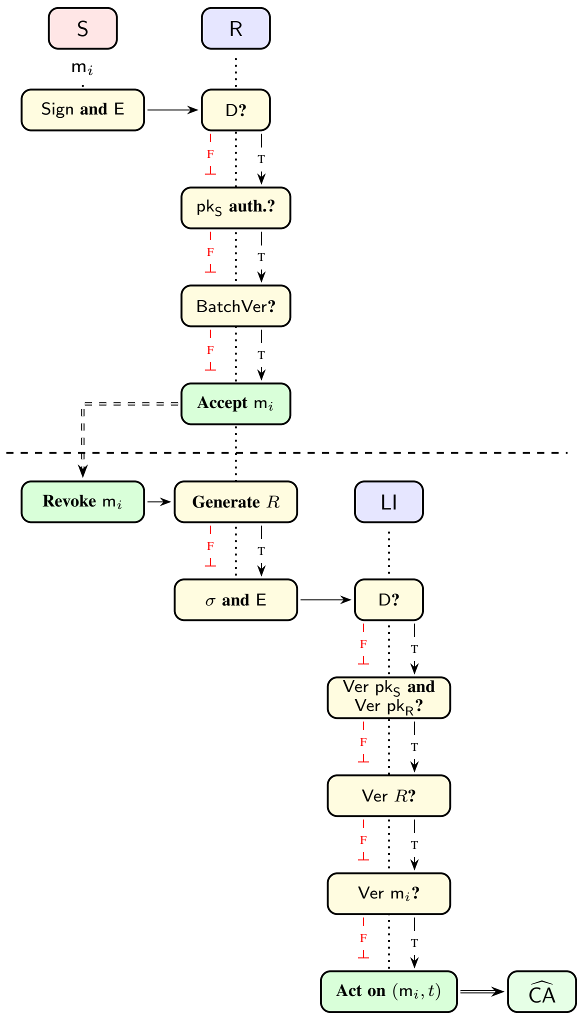

Fig. 5: A high-level diagram of the V2V communication between sender S and receiver R with revocation procedure initiated by the receiver R.

values (s, R) and allows the reconstruction of $(\mathsf{pp}, \vec{S_i})$ which is sent back to first $\widehat{\mathsf{CA}}$ and later forwarded back to LI. At this point, LI can verify the NIZK proof and, if correct, authenticate V by providing the epoch key ek and an epoch certificate σ. For security, each communication is encrypted using an authenticated symmetric encryption scheme $\mathsf{E}(k, \cdot)$ with a secret key obtained by computing the Diffie-Hellman key agreement and/or signed. If any verification fails, the protocol is aborted forcing the authentication procedure to be repeated.

Communication. Authenticated vehicles in an LI's region broadcast DAI messages to all its surrounding vehicles directly and follows the procedure displayed in Figure 5.

A sender S sends a message m_i by signing it obtaining σ_{m_i} and encrypting it together with its epoch certificate σ_{S}. The receiver R verifies the senders authentication's correctness by verifying σ_{S} and later verifies the message integrity σ_{m_i}. Further verifications are done which are fully detailed in the formal algorithm of Figure 3.

Batch verification can be used to optimize the receiver's computational complexity. We assume that R maintains a temporary record of verified sender's public keys in a database Δ_{pk}, a list of received and unverified public keys in Δ and a list of L signed messages received from the same sender in Δ_{m}. These buffers are used to store previously verified public keys to skip re-verifying them, block-listed keys of which messages are immediately discarded, managing the amount L of messages of which verification is still pending. When a new epoch starts, all temporary records in Δ_{pk}, Δ_{m} and Δ are deleted.

Revocation. If a vehicle S sends malicious DAI or other information to surrounding vehicles that can cause driving safety or security threats for other vehicle users; the proposed scheme allows any receiver R to report S to the LI as represented in Figure 5.

Briefly, R generates a report $\mathcal{R}$ containing any information from the sender and the incriminated signed message, together with the receiver's certificate and public key. The report is signed by R and provided to LI.

Once LI receives the report $\mathcal{R}$, there are specific verifications to be executed: the legitimacy of R and S must be verified to check if these are authenticated vehicles, the report's integrity to avoid any receiver's malicious reporting action, and finally the message signature to verify the reported content. While verifying the validity of a report $\mathcal{R}$, if any of these verification stages fail, LI can discard the report as malicious. Only if all these verification results are true, LI confirms the legitimacy of the report and must act accordingly. For example, LI can revoke S certificate and broadcast the block-listing of the public key pk_S to authenticated vehicles. Moreover, in scenarios where a malicious receiver R is continuously sending malicious reports or trying to frame a sender S, the LI can take action against R by revoking the certificate and excluding it from further participating in the protocol. Although the authenticity of the report is validated through our proposed revocation procedure, determining whether the message m is genuinely disputed lies beyond the scope of the communication protocol and this research. From the provided and the authentication information, LI can escalate the revocation to $\widehat{\mathsf{CA}}$ (respectively CA) which can de-anonymise the sender's identity and push for more serious consequences.

V. Batch Efficiency and Strategy

At first glance, batch verification appears more efficient than executing multiple standard single-signature verifications however, it is not clear how to handle the presence of wrong message-signature pairs in the batch. Curiously, the unofficial consensus is to avoid the usage of batch verification because of the increased difficulty in analysing the computation timing in the presence of errors: if in a batch of L signatures there are f invalid ones which the algorithm must correctly identify, is the batch verification still more efficient? The intuition behind such a question is that batch verification algorithmically costs like a standard verification plus some arithmetic computation that depends on the batch size and which computes the aggregated signature. However, both the aggregation and the number of standard verifications increases in the presence of error, motivating the community's doubt.

This section clarifies such a question by providing an optimal algorithmic strategy to correctly identify all the f invalid signatures out of a batch size of L. Such a strategy never computes more than L standard verification. We provide a thorough analysis of different scenarios and highlight when batch verification is more efficient. For the sake of clarity, we assume a batch of size $L = 2^\ell$ since these results can be used to extend to any size L.

A. Optimal Strategy for Batch Verification

The key observation on the batch verification algorithm (Figure 2) is that the verification has a linear system structure meaning that we can partition the indexes $I = I_1 \cup I_2$ and obtain two batch verifications for a smaller index set,

$$\sum_{i\in I}(\cdots) = \sum_{i\in I_1}(\cdots) + \sum_{i\in I_2}(\cdots)$$

allowing us to obtain the verification of I_2 by subtracting the verification of I by the one of I_1.

Consider the strategy Φ as in Figure 6 which is instantiated by considering all the indices $I = \{1,\ldots,L\}$ and $C = \mathsf{BatchVer}((\mathsf{pp}, b_I), \mathsf{pk}, \sigma_I, \mathsf{m}_I)$ where $C = \overline{R} - \overline{U} - \overline{V}$.

The strategy checks if the current verification is valid and returns all the current indices. If $C \neq 0$ then there are some

```
Strategy $\Phi(I, C)$
if $C = 0$ then return $I$
    / From here, $C \neq 0$
if $|I| = 1$ then return $\{\}$
    / From here, $C \neq 0$ and $|I| \neq 1$
$I_1 \cup I_2 \leftarrow I$   / Split the indices
  / Verify the first index set
$C_1 \leftarrow \mathsf{BatchVer}((\mathsf{pp}, b_{I_1}), \mathsf{pk}, \sigma_{I_1}, \mathsf{m}_{I_1})$
  / Linearly compute the second verification
$C_2 \leftarrow C - C_1$
  / Return the results of the recursive calls
return $\Phi(I_1, C_1) \cup \Phi(I_2, C_2)$
```

Fig. 6: Optimal verification strategy, recursive algorithm.

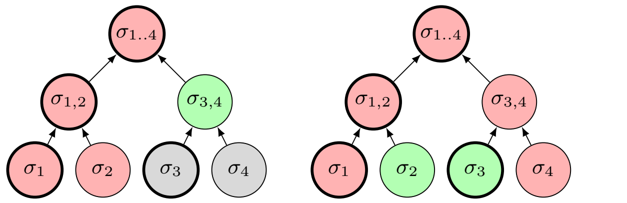

Fig. 7: Errors positions influencing the number of verifications required. Thicker border nodes are the strategies' verifications, red nodes have $C_i \neq 0$, green nodes have $C_i = 0$ and grey nodes are not considered by the strategy.

errors thus the partitioning and executing the verification on I_1 obtaining C_1 which is used to compute C_2 without executing the verification algorithm. The strategy dichotomically splits the problem into two smaller ones and maintains correctness because of the linearity property. The recursion terminates in maximum ℓ steps. The output of this recursive batch verification strategy is a list of indices for which the signatures and the corresponding messages are valid, as well as a list of indices for which the signatures are invalid and, therefore, the corresponding messages can be discarded.

The strategy gains in efficiency by obtaining the C_2 value via subtraction and not by executing a verification plus, by cleverly storing the precomputed values during the initial starting aggregation values for I, all the batch verification can be computed as standard verifications.

B. Errors, Best and Worst Scenario

The position of the invalid signatures influences the number of verification required by the strategy. An example with $L = 4$ signatures is highlighted in Figure 7 where the errors are either contiguous (σ_1, σ_2) or not (σ_1, σ_4).

The best scenario happens whenever all the errors are contiguous either at the beginning or the end of the batch in such a way as to minimise the number of intermediate subtrees affected. This amount is computed as $1 + \sum_{i=0}^{\ell} m_i(f)$ with $m_i(f)$ defined as:

$$m_i(f) = \left\lfloor \frac{m_{i+1}(f)+1}{2} \right\rfloor \qquad m_l(f) = \left\lfloor \frac{f+1}{2} \right\rfloor$$

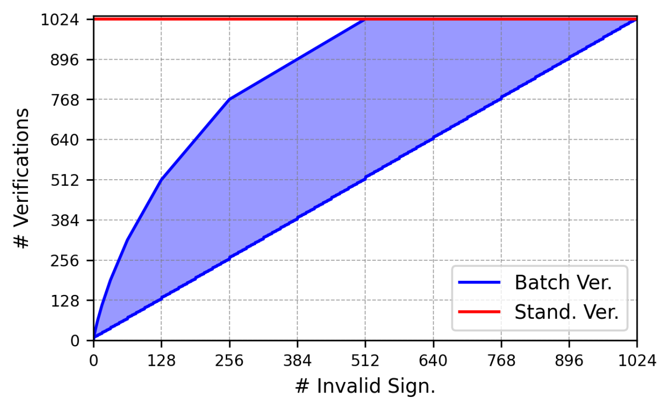

Fig. 8: Number of standard verifications for $L = 2^{10}$ signatures for non-batch and batch verification, both in the best and worst-error positioning scenarios.

The worst scenario happens whenever the errors maximise the number of subtrees affected. This curve can be computed as $1 + \sum_{i=0}^{\ell} M_i(f)$ with $M_i(f)$ defined as:

$$M_i(f) = \begin{cases} f & \text{if } f < 2^i \\ 2^i & \text{otherwise} \end{cases}$$

We plot in Figure 8, the number of standard verifications required both in the best and worst-error positioning scenarios and the amount required without batching. Batching never incurs additional standard verifications indicating that the aggregating pre-computations is the only limiting factor.

C. Analytical Efficiency of Batch Computation

Specifically for our application, we must take into consideration all the additional computational/space cost introduced by the aggregation and verification strategy. We evaluate the precise number of field and elliptic curve operations for the verification of $L = 2^\ell$ signatures for both the standard and the batch verification algorithms, considering the presence of f errors. With the formulas, we compare them and identify which verification is better and when.

Assumptions and Notations. Consider $\mathcal{T}_+, \mathcal{T}_\times, \mathcal{T}_{\mathbb{E}+}, \mathcal{T}_{\mathbb{E}\times}$ the timing to execute a finite field and an elliptic curve sum and product respectively while $\mathcal{M}_\mathbb{F}, \mathcal{M}_\mathbb{E}$ denotes the space necessary to store a field or curve element respectively and $\mathcal{T}_\mathsf{h}$ the timing for computing a hash. We assume the equality test to not have any cost while subtractions/inversions have the same cost as additions/multiplications. Since both algorithms must compute (u_i, v_i) from any message-signature pair received, we assume a receiver-unit computes and stores such values in a buffer used later for verification with total of cost of $L \cdot (\mathcal{T}_\mathsf{h} + 3\mathcal{T}_\times)$. We assume the receiver's unit to potentially be separated from the verification's unit thus these costs might not be considered in the comparison.

Standard Verification. For the standard verification, the total computations are trivial to compute. For each of the L signature, the algorithm must check $R \overset{?}{=} u_i{\cdot}G + v_i{\cdot}\mathsf{pk}$ obtained by computing two scalar curve multiplication and one curve addition, meaning,

$$L \cdot (2 \cdot \mathcal{T}_{\mathbb{E}\times} + \mathcal{T}_{\mathbb{E}+}) \tag{1}$$

Batch Verification. The batch verification can be split into three phases: a preparatory phase, the pre-computation of all the partial aggregations and the effective single verifications that depend on the error's amount and position.

The preparatory phase consists in the independent multiplication of each (R_i, u_i, v_i) by b_i costing a total of:

$$L \cdot (2 \cdot \mathcal{T}_\times + \mathcal{T}_{\mathbb{E}\times}) \tag{2}$$

The pre-computation requires the computation of the sum of all the L tuples (b_iR_i, b_iu_i, b_iv_i) meaning that $L-1$ sums must be made for each component for a total cost of:

$$(L-1) \cdot (2 \cdot \mathcal{T}_+ + \mathcal{T}_{\mathbb{E}+}) \tag{3}$$

To avoid re-computing the partial aggregation, the aggregations can cleverly store L partial aggregations used later by the verification strategy if an error occurs with a memory-cost of $L \cdot (2\mathcal{M}_\mathbb{F} + \mathcal{M}_\mathbb{E})$. These partial aggregations are the left branches (I_1) of all the possible verifications.

Regarding verification, the strategy executes one standard verification on the total aggregated values and, if there are errors, the number of operations depends on the amount of additional single verification required. We denote with c_f the total amount of verifications, including the first one.

For each additional verification, the algorithm computes a standard verification on half the indices and a curve subtraction between the verification results for a total of:

$$c_f \cdot (2 \cdot \mathcal{T}_{\mathbb{E}\times} + \mathcal{T}_{\mathbb{E}+}) + (c_f - 1) \cdot \mathcal{T}_{\mathbb{E}+} \tag{4}$$

where the right addendum denotes the additional subtractions computed for each verification except the first one.

1) Comparisons of Single vs Batch Verification: The comparison is done by verifying when Equation (1) is bigger than the sum of Equations (2) to (4). With some minor algebraic manipulation, we get the inequality:

$$2(L-1)\mathcal{T}_+ + 2L\mathcal{T}_\times + 2(c_f - 1)\mathcal{T}_{\mathbb{E}+} < (L - 2c_f)\mathcal{T}_{\mathbb{E}\times} \tag{5}$$

The left-hand side is always positive while the right-side can be negative. For the inequality to make sense, it must hold $c_f < \frac{L}{2}$. From the plot of Figure 8, we can consider the worst-positioning of the errors which implies a maximum error rate of $\sim \frac{1}{8}$. Furthermore, the inequality suggests a maximum size for the batch L too. Consider the worst-case scenario of $\frac{1}{8}$ error rate and $(L - 2c_f) = 1$. The right-hand side is a constant while all the factors on the left side depend on the batch size L, comprising c_f.

An easy optimization consists of outsourcing the preparation computations to the receiver's unit, meaning that whenever receiving a message-signature pair, the receiver not only prepares the values (R_i, u_i, v_i) but multiplies by b_i too thus offloading many computations from the verifier. This can be done if the verification and receiver are operating on two different controllers on the vehicle. As before, we compare Equation (1) and Equations (3) and (4):

$$2(L-1)\mathcal{T}_+ + 2(c_f - 1)\mathcal{T}_{\mathbb{E}+} < 2 \cdot (L - c_f)\mathcal{T}_{\mathbb{E}\times} \tag{6}$$

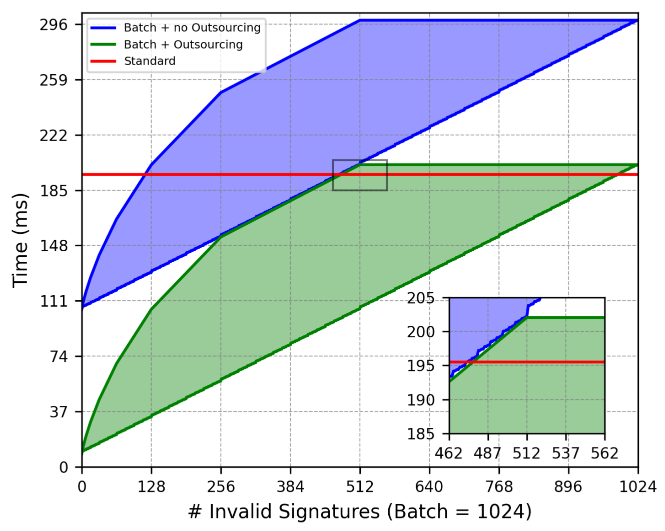

Fig. 9: Timing comparisons between the standard verification, batch with and without preparatory phase for and considering worst and best error positioning. The batch is of size $L = 2^{10}$ and Table IV is used to evaluate the curves.

obtaining $c_f < L$ implying, in the worst-case scenario, a maximum error rate of $\sim \frac{1}{2}$.

D. Empirical Analysis

To complete our analysis, we plot in Figure 9 the timing of the two verification methods using empirically measured timings for the operations of Table IV. We consider a batch size of $L = 2^{10}$ message-signature pairs, evaluate and plot the timing costs with a dependency on the error rate thus allowing to highlight both the worst and best error positioning scenarios, for both outsourcing the preparatory computations or not.

As the plot highlights, the preparatory phase has an enormous cost which quickly turns the batch verification to be less efficient than the standard verification and for an error rate higher than $\sim \frac{1}{2}$, it is always more expensive. However, outsourcing the preparatory phase allows the batch verification to always be more efficient or be slower by $2(L-1)(\mathcal{T}_{+}+\mathcal{T}_{\mathbb{E}+})$ at maximum. Despite such an amount depends on the batch size L, the difference only appears when almost half the signatures are invalid. For our example, the difference is $\sim 3.35\%$ additional computational cost.

VI. Security and Privacy Analysis

This section collects a formal security proof of our scheme together with a mechanically validated proof using Scyther, a tool for protocol's security verification [6, 37]. Lastly, we informally discuss further privacy and security guarantees.

A. Formal Security Proof

The security of our scheme is proved secure by considering an experiment $\mathsf{Exp}^{\mathsf{G}}(\mathcal{A})$ where an adversary $\mathcal{A}$ that interacts against a challenger $\mathcal{C}$ following the definition of a game G which is designed to describe a security property, e.g. creating an invalid signature, and terminates by returning if $\mathcal{A}$ wins or not the experiment. All parties involved in the experiment are probabilistic polynomial-time Turing machine [2]. We define the advantage of winning as $\mathsf{Adv}^{\mathsf{G}}(\mathcal{A})$ which can be computed as the probability $\mathsf{Pr}(\mathsf{G}(\mathcal{A}) \rightarrow \mathsf{win})$ of $\mathcal{A}$ winning the game G or as the difference:

$$\left|\mathsf{Pr}(\mathsf{G}(\mathcal{A}) \rightarrow \mathsf{win}) - \mathsf{Pr}(\mathsf{G}(\mathcal{A}) \rightarrow \mathsf{lose})\right|$$

Assumptions. We assume our scheme to be defined in the random oracle model (ROM) [38, 6, 9] meaning that we model our hash function H (and keyed H_s) as an ideal oracle that returns a consistent random output. This assumption is mandatory for the existence of NIZK.

Our scheme is defined using the Elliptic Curve Diffie-Hellman key agreement (DH) [35], ECDSA$*$ [36] and a keyed hash function H of which assumptions we assume to hold, i.e. their advantage is negligible:

- the DH's assumption measures the advantage ϵ_{DH} of computing $(a \cdot b)P$ from (P, aP, bP);
- ECDSA$*$ assumes that any adversary has advantage ϵ_{Sign} to create a valid message-signature pair without knowing the signing secret key;
- while ϵ_{H_s} is the advantage for $\mathcal{A}$ to compute the digest $\mathsf{H}_s(x)$ without knowing the secret seed s. The explicit evaluation of such an advantage relates to distinguishing between the hash and the random oracle.

We omit the formal description of the reductions for conciseness and instead directly develop the probabilities required to compute the advantage. At first, we compute the probability advantage for individual scheme-specific operations as games, and finally, a cumulative probability advantage is computed to establish the security strength of the entire scheme.

Authentication. Define the authentication game Auth where the adversary $\mathcal{A}$, playing the role of a malicious vehicle V, can authenticate and obtain the epoch key ek by providing wrong data to the challenger, acting as LI.

Theorem 1. The advantage of $\mathcal{A}$ in the Auth game is,

$$\mathsf{Adv}^{\mathsf{Auth}}(\mathcal{A}) \leq \eta_{q_{\Delta},q} + \eta_{R,q} \cdot \epsilon_{\mathsf{H}_s}$$

where p is the elliptic group's order, q_{Δ} is the (polynomial) amount of (allowed) authenticated vehicle, q is the (polynomial) number of authentication attempts and the coefficients $(\eta_{q_{\Delta},q}, \eta_{R,q})$ is computed as described in the proof.

Proof. All communications are encrypted thus providing no additional information that $\mathcal{A}$ can used to maliciously authenticate. As highlighted in Figure 4, to obtain ek, $\mathcal{A}$ must correctly pass the NIZKVer for the index $h_i \leftarrow \mathsf{H}(s_i)$ computed from the secret s_i which is a random element of $\mathbb{Z}_p$. Since s_i can be randomly selected by $\mathcal{A}$ from which a correct proof π can be computed, if $\mathcal{A}$ provides a $h_i \leftarrow \mathsf{H}(s_i)$ such that $h_i \in \Delta_{\mathsf{LI}}$, $\mathcal{A}$ maliciously obtains the epoch key. Formally,

$$\mathsf{Pr}(h_i \in \Delta_{\mathsf{LI}}) \cdot \mathsf{Pr}(\mathsf{win} | h_i \in \Delta_{\mathsf{LI}}) = \mathsf{Pr}(h_i \in \Delta_{\mathsf{LI}}) \cdot 1$$

which is equal to sampling the same secret as one of the already authenticated vehicles which are q_Δ. The probability of guessing h_i in q queries is the combinatorial amount,

$$\eta_{q_\Delta,q} = 1 - \frac{\binom{p-q_\Delta}{q}}{\binom{p}{q}} = 1 - \prod_{i=0}^{q-1} \frac{p - q_\Delta - i}{p - i}$$

from which one gets that,

$$\mathsf{Pr}(h_i \in \Delta_{\mathsf{LI}}) \le \eta_{q_\Delta,q}$$

In the event of $h_i \notin \Delta_{\mathsf{LI}}$, $\mathcal{A}$ must provide valid ciphertext $(\mathsf{ct}_c, \mathsf{ct}_x)$ and to guarantee the output of the statement, the indicated public key $h_{\mathsf{pk_V}}$ must be present in Δ_{CA}, i.e. the vehicle with public key $\mathsf{pk_V}$ is correctly registered with CA, the index is correct $i \in R$ and the secret $s_i = \mathsf{H}_s(\mathsf{pk_{LI}}, i)$ since the hash check can always be passed. Formally,

$$\mathsf{Pr}(h_i \notin \Delta_{\mathsf{LI}}) \cdot \mathsf{Pr}(\mathsf{win}|h_i \notin \Delta_{\mathsf{LI}}) \le \mathsf{Pr}(\mathsf{win}|h_i \notin \Delta_{\mathsf{LI}})$$

that can be computed as,

$$\begin{aligned}\mathsf{Pr}(\mathsf{win}|h_i \notin \Delta_{\mathsf{LI}}) &= \mathsf{Pr}(h_{\mathsf{pk_V}}) \cdot \mathsf{Pr}(i \in R) \cdot \mathsf{Pr}(s_i)\\ \mathsf{Pr}(\mathsf{win}|h_i \notin \Delta_{\mathsf{LI}}) &\le 1 \cdot \eta_{R,q} \cdot \epsilon_{\mathsf{H}_s}\end{aligned}$$

where ϵ_{H_s} indicates the probability for $\mathcal{A}$ to correctly guess the digest of a given input without knowing s and the coefficient $\eta_{R,q}$ is similarly as before computed as,

$$\eta_{R,q} = 1 - \frac{\binom{p-|R|}{q}}{\binom{p}{q}} = 1 - \prod_{i=0}^{q-1} \frac{p - |R| - i}{p - i}$$

By putting everything together, we obtain,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Auth}}(\mathcal{A}) &= \mathsf{Pr}(\mathsf{Auth}(\mathcal{A}) \to \mathsf{win})\\ &\le \mathsf{Pr}(h_i \in \Delta_{\mathsf{LI}}) + \mathsf{Pr}(\mathsf{win}|h_i \notin \Delta_{\mathsf{LI}})\\ &\le \eta_{q_\Delta,q} + \eta_{R,q} \cdot \epsilon_{\mathsf{H}_s}\end{aligned}$$

concluding the proof. □

The parameter q_Δ is important for the authentication security guarantee of the scheme and is an intrinsic trade-off between practicality and security. Allowing many vehicles to be authenticated in the same epoch facilitates $\mathcal{A}$'s advantage in guessing h_i with fewer authentication attempts thus the epoch must be updated more often to avoid such a possibility. In practice, this concern is mainly theoretical since the number of vehicles q_Δ and authentication attempts q are polynomial which is negligible when compared with the order p of the group.

Verification. Define the forgery game Forgery where the adversary $\mathcal{A}$, playing the role of a malicious sender S, forces an invalid message-signature pair to be accepted by the challenger, playing the role of the receiver R.

Theorem 2. The advantage of $\mathcal{A}$ in the Forgery game is,

$$\mathsf{Adv}^{\mathsf{Forgery}}(\mathcal{A}) \le \epsilon_{\mathsf{Sign}}$$

Proof. As highlighted in Figure 5, $\mathcal{A}$ must at least provide a bad signature σ_i for the message m_i since we can assume the adversary $\mathcal{A}$ to be authenticated. The advantage of winning Forgery is bounded by the advantage ϵ_{Sign} of providing a valid forgery. □

Revocation. Define the revocation game Revoke where the adversary $\mathcal{A}$, playing the role of a malicious receiver R, forces an invalid report $\mathcal{R}$ to be accepted by the challenger, playing the role of LI.

Theorem 3. The advantage of $\mathcal{A}$ in the Revoke game is,

$$\mathsf{Adv}^{\mathsf{Revoke}}(\mathcal{A}) \le \epsilon_{\mathsf{Sign}}$$

Proof. Same reasoning as in Theorem 2. □

From the above theorems, the explicit probability advantages of $\mathcal{A}$ are computed to provide the security strength of the proposed scheme from the designed games. However, to understand the overall security strength of the entire protocol, a cumulative theorem can be formulated as follows:

Theorem 4. The cumulative probability advantage $\mathsf{Adv}^{\mathsf{Scheme}}(\mathcal{A})$ of the proposed authentication framework is given by

$$\mathsf{Adv}^{\mathsf{Scheme}}(\mathcal{A}) \le \eta_{q_\Delta,q} + \eta_{R,q} \cdot \epsilon_{\mathsf{H}_s} + 2\epsilon_{\mathsf{Sign}}$$

Proof. By combining Theorem 1,Theorem 2 and Theorem 3 we get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{Scheme}}(\mathcal{A}) &\le \mathsf{Adv}^{\mathsf{Auth}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{Forgery}}(\mathcal{A}) + \mathsf{Adv}^{\mathsf{Revoke}}(\mathcal{A})\\ &\le \eta_{q_\Delta,q} + \eta_{R,q} \cdot \epsilon_{\mathsf{H}_s} + 2\epsilon_{\mathsf{Sign}}\end{aligned}$$

□

This negligible cumulative probability advantage protects our scheme against feasible polynomial-time attacks performed by $\mathcal{A}$.

B. Scyther Security Proof

We validated our protocol's security using Scyther, a widely accepted tool for formal security verification [6, 37]. Scyther rigorously and automatically verifies protocols against predefined security goals like secrecy, authenticity, integrity, non-repudiation, freshness, session key forward and backward secrecy, aliveness, and agreement.

Protocols are modeled as `spdl` code, which defines roles, message flows, actions, and security claims [39]. Scyther then generates a protocol state-tree and analyzes for possible security vulnerabilities. Scyther's fundamental security properties follow the Dolev-Yao, CK, and eCK security models that include:

- **Secret:** ensures a parameter's confidentiality remains exclusive to its specified role.
- **Alive:** ensures all protocol parties perform all transitions and events, simulating availability.
- **Weakagree:** ensures the protocol is safe from replay attacks.
- **Niagree:** checks against man-in-the-middle, impersonation, non-repudiation, and modification attacks.
- **Nisynch:** verifies message integrity to prevent non-injective synchronization issues.

For our scheme, we modeled the V2I authentication and V2V message-sharing phase. In initial simulations, our protocol passed all predefined security claims for specific attack scenarios, ensuring the confidentiality and integrity of messages

TABLE III: Formal security simulated in Scyther verification tool.

Simulation Type: Predefined security claim check					
Operation	**Alive**	**Weakagree**	**Niagree**	**Nisynch**	**Secret**
V2I- V	✓	✓	✓	✓	✓
V2I-LI	✓	✓	✓	✓	✓
V2V-S	✓	✓	✓	✓	✓
V2V-R	✓	✓	✓	✓	✓
Simulation Type: Scyther Automated security claim check					
V2I- V	✓	✓	✓	✓	✓
V2I-LI	✓	✓	✓	✓	✓
V2V-S	✓	✓	✓	✓	✓
V2V-R	✓	✓	✓	✓	✓
Simulation Type: Modeled Security Attack and Vulnerability Test					
Secret key (sk) leaked	✓	✗	✗	✗	✗
Session key (ek)leaked	✗	✗	✗	✗	✓
Signature Forging	✗	✗	✗	✗	✓

and transition states. In further tests, we have used Scyther's automated checks to simulate various attack scenarios and confirmed our protocol's robustness against various threats. As shown in Table III, our protocol is provably secured against all predefined claims and automated security checks by Scyther. Additionally, we have intentionally leaked protocol-specific secrets to analyze further how the scheme behaves under a designed attack scenario. It is important to note that modeled attack scenarios are intentionally created to simulate a vulnerable situation and study the protocol behavior; it does not mean that the protocol is vulnerable to these attacks. Therefore, we test the protocol by introducing intentional leaks to the adversary, thus simulating an adversary's ability to obtain such leaked data, which has a negligible probability of happening. We intentionally leak the vehicle's secret key and timestamps to simulate confidentiality vulnerability. Under this attack, Scyther detects that all the other security properties are failed apart from `alive`. In the next scenario, we intentionally leak a session key; following that, we also simulate an impersonation attack by forging a signature or intentionally bypassing the signature verification step. For both these tests, Scyther detects that the vehicle's secret parameters still remain confidential as it passes the `secret` property while all the other security properties are violated.

C. Informal Privacy and Security Analysis

Following the security and adversarial assumptions presented in Section III, we argue about the scheme's ability to defend against the following attacks and vulnerabilities:

- **Privacy Theft**: to trace a vehicle's traffic routes or obtain identity-based information from the communication, an adversary $\mathcal{A}$ must link authentication messages from different instances. Privacy theft hinges on the linkability of authentication requests from the same vehicle in different epochs. However, since the vehicle generates a new key-pair $(\mathsf{sk},\mathsf{pk})$ for each epoch and uses a randomized r, the authentication ciphertext ct and pk are unlinkable across epochs. The only linkable information is $h_{\mathsf{pk}_\mathsf{V}}$ which contained in the ciphertext ct_x and only the correct certification authority can obtain it. Any other potentially malicious party would be required to break the symmetric encryption scheme E or the key agreement, both assumed secure. Also, no specific handover request is sent when vehicles change regions. Thus, the proposed scheme prevents $\mathcal{A}$ from linking messages, preserving the vehicle's location and identity privacy. Within an epoch, vehicles are traceable by other verified vehicles by sharing DAI, which is allowed in VANET. Given the short epoch duration, this traceability does not compromise vehicle privacy.
- **Modification Attacks**: an adversary can try to modify an authentication request to an LI or a shared V2V broadcast from a vehicle. To alter any message, the adversary must break the key agreement scheme to obtain the shared secret used to decrypt the ciphertext. This is formally described and proven infeasible by Theorems 1 and 2.
- **Replay Attack**: each authentication request to an LI and authenticated V2V message is signed and encrypted with a timestamp. An adversary replaying an authentication request will fail since the LI decrypts and must validate the timestamp t. For V2V messages, the receiver vehicle checks the freshness of t and the validity of $(\mathsf{pk}_\mathsf{S}, \sigma_\mathsf{S})$ within the current epoch, preventing replay attacks in the same or different epochs.
- **Man-in-the-Middle Attacks (MitM)**: MitM attacks are prevented because of the usage of certificates σ_{LI} forcing an adversary to gain/forge them to execute the attack.
- **Impersonation Attack**: an adversary $\mathcal{A}$ may attempt to impersonate a valid vehicle or an LI to maliciously deviate from the honest execution of the protocol. However, $\mathcal{A}$ would be required to provide several $\mathsf{ECDSA}*$ signatures which means $\mathcal{A}$ must obtain the secret key sk from a known public key pk. This is assumed secure since it is based on the discrete logarithm problem.
- **Forgery Attack**: during V2V batch verification at any receiver R,$\mathcal{A}$ can try to forge false signatures σ_m to pass the batch verification process. This is formally described and proven infeasible by Theorem 2.
- **Sybil Attack**: a malicious vehicle may attempt to generate multiple identities within the same LI. Since LI cannot distinguish if two authentication attempts come from the same vehicle, this protection must be provided by CA which is the only (trusted) authority that can identify different attempts for the same vehicle. To solve this, CA might limit the amount of authentication to force a malicious vehicle to reuse previously provided values h_i thus highlighting to LI the attempted Sybil attack. This technique would allow Sybil-free authentication in the network.
- **Non-repudiation**: a sender vehicle S might deny sending a malicious message reported to the LI by a receiver R. However, in our scheme, each V2V message m is signed by the sender S as σ_m and broadcasted with its public key pk_S and certificate σ_S. If m is reported, any receiver R or the LI can verify the signature in σ_m to confirm it was signed by S thus disallowing the message sending denial.
- **Denial-of-Service (DoS)**: an adversary $\mathcal{A}$ may flood the network with randomly generated authentication messages to overwhelm the LI and disrupt its services. When

TABLE IV: Execution times of cryptographic operations.

Symbol	Description	Timing (ms)
$\mathcal{T}_{+}$	Addition of two field elements	≈ 0.0009
$\mathcal{T}_{\times}$	Multiplication of two field elements	≈ 0.0011
$\mathcal{T}_{\mathsf{exp}}$	Exponentiation operation	≈ 0.0083
$\mathcal{T}_{\mathsf{E}+}$	Point addition on curve secp256k1	≈ 0.0023
$\mathcal{T}_{\mathsf{E}\times}$	Point multiplication on curve secp256k1	≈ 0.0921
$\mathcal{T}_{\mathsf{h}}$	Hash operation	≈ 0.0011
$\mathcal{T}_{\mathsf{E}}$	AES encryption	≈ 0.0026
$\mathcal{T}_{\mathsf{D}}$	AES decryption	≈ 0.0032
$\mathcal{T}_{\mathsf{Sign}}$	ECDSA signature	≈ 0.0944
$\mathcal{T}_{\mathsf{Ver}}$	ECDSA verification	≈ 0.1865
$\mathcal{T}_{\mathsf{bp}}$	Bilinear-pairing of two groups	≈ 3.8721
$\mathcal{T}_{\mathsf{bp}*}$	Pairing-based multiplication operation	≈ 0.6621
$\mathcal{T}_{\mathsf{bp}+}$	Pairing-based addition operation	≈ 0.0461

TABLE V: Computation costs of the proposed protocol.

Operations	Computations	Timing (ms)
V Auth. (Best case)	$\mathcal{T}_{\mathsf{h}} + 6\mathcal{T}_{\mathsf{E}\times} + \mathcal{T}_{\mathsf{Sign}} + 3\mathcal{T}_{\mathsf{E}}$	≈ 0.64
V Auth. (Worst case)	$3\mathcal{T}_{\mathsf{h}} + 8\mathcal{T}_{\mathsf{E}\times} + \mathcal{T}_{\mathsf{Sign}} + 3\mathcal{T}_{\mathsf{E}}$	≈ 0.82
LI Auth.	$7\mathcal{T}_{\mathsf{E}\times} + 2\mathcal{T}_{\mathsf{E}+} + 3\mathcal{T}_{\mathsf{D}} + 2\mathcal{T}_{\mathsf{h}} + +\mathcal{T}_{\mathsf{Sign}} + \mathcal{T}_{\mathsf{Ver}} + \mathcal{T}_{\mathsf{E}}$	≈ 0.91
V Acknow.	$\mathcal{T}_{\mathsf{D}} + \mathcal{T}_{\mathsf{Ver}}$	≈ 0.18
S Send	$\mathcal{T}_{\mathsf{Sign}} + \mathcal{T}_{\mathsf{E}}$	≈ 0.09
R Verify (Standard)	$\mathcal{T}_{\mathsf{D}} + 2\mathcal{T}_{\mathsf{Ver}}$	≈ 0.38
R Verify (Batch)	$\approx 0.09 * L + 0.18 * c_f$	
R Reports	$\mathcal{T}_{\mathsf{Sign}} + \mathcal{T}_{\mathsf{E}\times} + \mathcal{T}_{\mathsf{E}}$	≈ 0.18
LI Verify	$5\mathcal{T}_{\mathsf{Ver}} + \mathcal{T}_{\mathsf{E}\times} + \mathcal{T}_{\mathsf{D}}$	≈ 1.02

TABLE VI: Communication overhead in our scheme.

Communication	Message Size (Bytes)
Authentication Request	$230 = (33*2+32*4+16*2+4)$
Acknowledgement Tuple	$112 = (16*3+64)$
V2V Tuple (without m)	$179 = (64*2+33+14+4)$
Revocation Request (no m)	$342 = (64*4+33*2+16+4)$

an LI decrypts an authentication request, it verifies the received h_i against its database. If h_i is absent, the LI forwards the request to the CA. Verification proceeds only upon receiving a valid response from the CA; otherwise, the request is discarded immediately. This proactive approach minimizes vulnerability to DoS attacks by detecting and preventing malicious requests early on.

VII. Performance Analysis

To assess our scheme's performance efficiency, we adopted the methodology used by several prior studies including [40, 4, 20, 21]. This approach focuses exclusively on evaluating the computational and execution costs of cryptographic operations within the scheme and the communication overhead involved in message exchange during various stages of the authentication process. Our analysis excludes parameters such as end-to-end channel latency, transmission delays, and other physical layer considerations, as these aspects are beyond the scope of our study.

Our timings are obtained using an Intel i7-6500U @ 2.50GHz CPU, 16GB of RAM. The cryptographic functions were implemented in C using the OpenSSL library [41] and the PBC library [42] on a Linux virtual environment. We use a 128-bit AES symmetric key and the secp256k1 curve [43]. The secp256k1 curve is defined on a 256-bit prime field with a 256-bit order. We repeatedly execute the cryptographic computations performed and collect the timings in Table IV.

A. Proposed Scheme's Costs

The computation costs of our scheme are evaluated based on the cryptographic executions required for V2I authentication, V2V batch verification, and dispute reports. Table V details these computations and approximates the total execution times across all phases. During V2I authentication, two verification scenarios are considered: a best-case scenario where LI already knows the statement for h_i and a worst-case scenario where the statement must be queried to $\widehat{\mathsf{CA}}$. In the best-case scenario, authenticated V2I key-sharing completes within ≈ 1.73 ms while in the worst-case scenario, it takes ≈ 1.91 ms. Notably, most authentications align with the best-case scenario as once h_i is established, it can be used for a long period. For a sender, S generates and broadcasts a V2V message within ≈ 0.09 ms, while a receiver R needs 0.38ms to perform single-message verification or $0.096L + 0.18$ ms for batch-verifying L messages. R generates report in ≈ 0.18ms and LI takes ≈ 1.02ms for report verification.

The communication costs are based on the sizes of messages exchanged. Since the scheme uses ECDSA$*$ signatures and AES encryption/decryption, we calculate message sizes by considering the encrypted signatures and authentication secrets exchanged during V2I authentication, V2V message sharing, and dispute reporting. Each ECDSA$*$ signature is 64 bytes, the private key is 32 bytes, and the compressed public key is 33 bytes. For AES in Galois counter-mode, the ciphertext size is the plaintext size plus 16 bytes for the authentication tag. The SHA256 hash outputs 32 bytes and each timestamp is 4 bytes. Table VI lists the message sizes for all communication phases. In both best and worst-case scenarios, the size of security-related messages in the communication process is fixed. An authentication request from a vehicle is always 230 bytes, determined by a fixed set of parameters. The acknowledgment tuple sent by the LI to the vehicle is 112 bytes. For V2V communication, each shared message includes 179 bytes of security data in addition to the original message m. During disputes, the receiver R generates a revocation request of 342 bytes. The size of the original message (m) is excluded from these calculations, as only security-related parameters are considered.

B. Comparative Efficiency

We compare the computation and communication efficiency of V2I authentication and V2V batch verification of our scheme against several similar schemes from the literature. For the authentication comparison, we consider pairing-based authentication schemes of which costs are shown in Table VII. As pairing is computationally heavier than executing ECC-multiplications, these schemes require much longer computation times to perform authentication. With a single bilinear pairing operation, the schemes Azees et al. [44], Bayat et al.

TABLE VII: Computational and communication overhead for V2I authenticated key sharing. The value k_1 depends on the number of parameters used for the assessment of vehicles' trustworthiness in a blockchain-based trust management [24], k_2 denote the edge nodes involved in the authentication process [40].

Scheme	Computation Overhead	Timing (ms)	Communication Overhead (Bytes)	
			Request	Acknowledge
Azees et al. [44]	$\mathcal{T}_{\text{bp}} + 9\mathcal{T}_{\text{bp}*} + 2\mathcal{T}_{\text{h}}$	≈ 9.81	576	512
Wang et al. [24]	$2\mathcal{T}_{\text{bp}} + 8\mathcal{T}_{\text{bp}*} + 4\mathcal{T}_{\text{h}}$	≈ 13.02	$64 * k_1$	452
Bayat et al. [28]	$\mathcal{T}_{\text{bp}} + 9\mathcal{T}_{\text{bp}*} + 9\mathcal{T}_{\text{h}} + 2\mathcal{T}_{\text{E}}$	≈ 9.82	192	616
Yang et al. [40]	$7\mathcal{T}_{\text{bp}} + 6\mathcal{T}_{\text{bp}*} + 8\mathcal{T}_{\text{h}}$	≈ 31.07	228	$128 * k_2 + 32$
Our Scheme (Best)	$9\mathcal{T}_{\text{E}\times} + 2\mathcal{T}_{\text{E}+} + 2\mathcal{T}_{\text{Sign}} + 4\mathcal{T}_{\text{E}} + \mathcal{T}_{\text{D}} + \mathcal{T}_{\text{h}}$	≈ 1.73	230	112
Our Scheme (Worse)	$15\mathcal{T}_{\text{E}\times} + 2\mathcal{T}_{\text{E}+} + 2\mathcal{T}_{\text{Sign}} + \mathcal{T}_{\text{Ver}} + 4\mathcal{T}_{\text{E}} + 3\mathcal{T}_{\text{D}} + 5\mathcal{T}_{\text{h}}$	≈ 1.91	230	112

TABLE VIII: The computational costs of V2V single and batch verification of different schemes for a batch size of $L = 128$.

Scheme	Signing Costs		Verification Computation Costs		
	Overhead	Timing (ms)	Algorithm	Overhead	Timing (ms)
Yan et al. [15]	$4\mathcal{T}_{\text{E}\times} + 3\mathcal{T}_{\text{h}}$	≈ 0.37	Ver	$18\mathcal{T}_{\text{E}\times} + 9\mathcal{T}_{+} + 9\mathcal{T}_{\times} + 8\mathcal{T}_{\text{h}}$	≈ 1.68
			BatchVer	$(3L+2)\mathcal{T}_{\text{E}\times} + 3L\mathcal{T}_{+} + 3L\mathcal{T}_{\times} + 2L\mathcal{T}_{\text{h}}$	$0.28L + 0.1842$
Chen et al. [20]	$2\mathcal{T}_{\text{bp}*} + \mathcal{T}_{\text{bp}+}$	≈ 1.37	Ver	$2\mathcal{T}_{\text{bp}} + \mathcal{T}_{\text{bp}*} + \mathcal{T}_{\text{bp}+} + 2\mathcal{T}_{\text{E}\times} + \mathcal{T}_{\text{exp}}$	≈ 8.64
			BatchVer	$2\mathcal{T}_{\text{bp}} + L\mathcal{T}_{\text{bp}*} + 2L\mathcal{T}_{\text{E}\times} + (3L-2)\mathcal{T}_{\text{E}+} + L\mathcal{T}_{\times} + L\mathcal{T}_{\text{exp}}$	$\approx 0.86L + 7.56$
Shen et al. [21]	$\mathcal{T}_{\text{bp}} + \mathcal{T}_{\text{bp}*} + \mathcal{T}_{\text{exp}}$	≈ 4.54	Ver	$2\mathcal{T}_{\text{bp}} + \mathcal{T}_{\times} + \mathcal{T}_{\text{exp}}$	≈ 7.75
			BatchVer	$2L\mathcal{T}_{\text{bp}} + L\mathcal{T}_{\times} + L\mathcal{T}_{\text{exp}}$	$\approx 7.75L$
Feng et al. [22]	$2\mathcal{T}_{\text{bp}} + 11\mathcal{T}_{\text{bp}*} + 12\mathcal{T}_{\text{exp}}$	≈ 15.12	Ver	$4\mathcal{T}_{\text{bp}} + 10\mathcal{T}_{\text{bp}*} + 10\mathcal{T}_{\text{exp}}$	≈ 22.19
			BatchVer	$4\mathcal{T}_{\text{bp}} + (6L-1)\mathcal{T}_{\text{bp}*}$	$\approx 3.97L + 14.82$
Our Scheme	$\mathcal{T}_{\text{Sign}} + \mathcal{T}_{\text{E}}$	≈ 0.09	Ver	$2\mathcal{T}_{\text{Ver}} + \mathcal{T}_{\text{D}}$	≈ 0.37
			BatchVer	$2L\mathcal{T}_{\times} + 2(L-1)\mathcal{T}_{+} + (L+2)\mathcal{T}_{\text{E}\times} + L\mathcal{T}_{\text{E}+}$	$\approx 0.09L + 0.18$

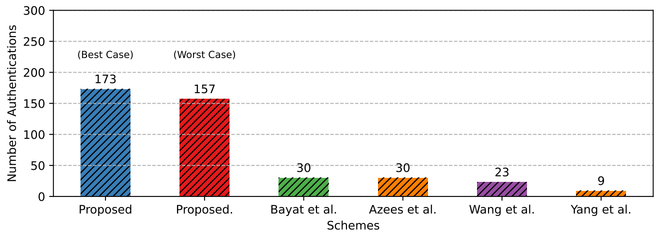

Fig. 10: Number of V2I Authentications performed within 300 ms.

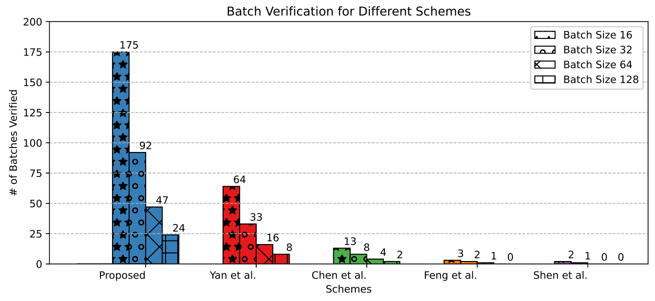

Fig. 11: Number of V2V batches verified for batch sizes $L \in \{16, 32, 64, 128\}$ with a time-threshold 300ms in different schemes

[28] require $\approx$ 9.8ms to complete authentication. With two pairing operations, Wang et al. [24] proposed an authentication scheme that requires $\approx$ 13ms. With a distributed edge-based authentication framework, the scheme proposed by Yang et al. [40] requires $\approx$ 31ms to perform a single authentication. Compared to these schemes, our proposed scheme completes vehicle verification significantly faster with a total authentication timing of approximately 1.79 to 1.91 ms and lower total communication overhead. A simulation of the number of authentications each scheme can execute within 300 ms is reported in Figure 10. Our scheme executes 173 in best case and 157 V2I authentications in worst-case scenarios, which is significantly higher than the other comparative schemes that only execute between 9 to 30 authentications only, highlighting its superior efficiency in execution times.

We compare the V2V signing and (batch) verification efficiency of our scheme against related schemes from the literature. As reported in Table VIII, our scheme requires fewer computations, allowing it to outperform all the other schemes to sign and perform standard verification of V2V messages. Relying on ECDSA$*$ operations, the proposed batch scheme is significantly faster than the bilinear pairing-based batch verifications by Chen et al. [20], Shen et al. [21], Feng et al. [22] or even the ECC-based batch verification proposed by Yan et al. [15]. Unlike the comparative schemes, our scheme facilitates the identification of faulty signatures while also being computationally fast and efficient. As reported in Figure 11, when considering the amount of batch verified within a 300 ms threshold, our method can verify a maximum of $\approx$ 2800 V2V messages (175$*$16) within 300ms significantly higher than the others.

VIII. Conclusions

In this paper, we presented an innovative authentication framework for VANETs that addresses the critical challenges of privacy, security, and efficiency. The proposed distributed V2I authenticated key-sharing scheme eliminates the need for explicit handover requests when vehicles change regions, thus preserving the vehicle's location and identity-based privacy during transit between different CA regions. Additionally, our V2V batch-verification approach effectively handles faulty message signatures, enhancing the reliability and efficiency of VANET communications. Through extensive analysis, we demonstrated that our proposed methods outperform similar existing protocols, particularly in scenarios with high vehicle density and frequent message exchanges. The use of lightweight ECC and a focus on practical deployment conditions make our framework highly applicable to real-world VANET systems, offering a robust solution for secure and efficient vehicular communication.

References

[1] H. Hasrouny *et al.*, "VANET Security Challenges and Solutions: A Survey," *Vehicular Communications*, vol. 7, pp. 7–20, 2017.

[2] H. Tahir *et al.*, "Lightweight and secure multi-factor authentication scheme in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14 978–14 986, 2023.

[3] X. Li *et al.*, "BDRA: Blockchain and Decentralized Identifiers Assisted Secure Registration and Authentication for VANETs," *IEEE Internet of Things Journal*, 2022.

[4] L. Wei *et al.*, "A Lightweight and Conditional Privacy-Preserving Authenticated Key Agreement Scheme with Multi-TA Model for Fog-based VANETs," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[5] S. Naskar *et al.*, "A Scheme for Distributed Vehicle Authentication and Revocation in Decentralized VANETs," *IEEE Access*, 2024.

[6] A. K. Yadav *et al.*, "iVFAS: An Improved Vehicle-to-Fog Authentication System for Secure and Efficient Fog-based Road Condition Monitoring," *IEEE Transactions on Vehicular Technology*, 2024.

[7] O. B. Piramuthu and M. Caesar, "VANET Authentication Protocols: Security Analysis and A Proposal," *The Journal of Supercomputing*, vol. 79, no. 2, pp. 2153–2179, 2023.

[8] S. Son *et al.*, "Design of Blockchain-Based Lightweight V2I Handover Authentication Protocol for VANET," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 3, pp. 1346–1358, 2022.

[9] J. Zhang *et al.*, "Cvar: Distributed and Extensible Cross-Region Vehicle Authentication with Reputation for VANETs," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[10] M. Arif *et al.*, "A Survey on Security Attacks in VANETs: Communication, Applications and Challenges," *Vehicular Communications*, vol. 19, p. 100179, 2019.

[11] P. Vijayakumar *et al.*, "An Anonymous Batch Authentication and Key Exchange Protocols for 6G Enabled VANETs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1630–1638, 2021.

[12] Z. Liu *et al.*, "Efficient Small-Batch Verification and Identification Scheme with Invalid Signatures in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 836–12 846, 2021.

[13] Chen, Jing and He, Kun and Yuan, Quan and Xue, Guoliang and Du, Ruiying and Wang, Lina, "Batch Identification Game Model for Invalid Signatures in Wireless Mobile Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 6, pp. 1530–1543, 2016.

[14] Xiong, Hu and Jin, Chuanjie and Alazab, Mamoun and Yeh, Kuo-Hui and Wang, Hanxiao and Gadekallu, Thippa Reddy and Wang, Weizheng and Su, Chunhua, "On the Design of Blockchain-Based ECDSA with Fault-Tolerant Batch Verification Protocol for Blockchain-Enabled IoMT," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1977–1986, 2021.

[15] C. Yan *et al.*, "Edge-Assisted Hierarchical Batch Authentication Scheme for VANETs," *IEEE Transactions on Vehicular Technology*, 2023.

[16] M. Akil *et al.*, "A Privacy-Preserving Approach to Vehicle Renting and Driver Accountability in VANETs," in *IFIP International Summer School on Privacy and Identity Management*. Springer, 2023, pp. 192–210.

[17] A. S. Kittur and A. R. Pais, "A New Batch Verification Scheme for ECDSA* Signatures," *Sādhanā*, vol. 44, no. 7, p. 157, 2019.

[18] X. Liu *et al.*, "PTAP: A Novel Secure Privacy-Preserving & Traceable Authentication Protocol in VANETs," *Computer Networks*, vol. 226, p. 109643, 2023.

[19] Y. Zhou *et al.*, "An Efficient and Provably Secure Identity Authentication Scheme for VANET," *IEEE Internet of Things Journal*, vol. 10, no. 19, pp. 17 170–17 183, 2023.

[20] S. Chen *et al.*, "BASRAC: An Efficient Batch Authentication Scheme with Rule-based Access Control for VANETs," *Vehicular Communications*, vol. 40, p. 100575, 2023.

[21] J. Shen *et al.*, "Secure Real-Time Traffic Data Aggregation with Batch Verification for Vehicular Cloud in VANETs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 807–817, 2019.

[22] X. Feng *et al.*, "P2BA: A Privacy-Preserving Protocol with Batch Authentication Against Semi-Trusted RSUs in Vehicular Ad Hoc Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3888–3899, 2021.

[23] D. D. Sikarwar, Himani, "Towards Lightweight Authentication and Batch Verification Scheme in IoV," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3244–3256, 2021.

[24] C. Wang *et al.*, "B-TSCA: Blockchain Assisted Trustworthiness Scalable Computation for V2I Authentication in VANETs," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1386–1396, 2020.

[25] Z. Ma *et al.*, "An Efficient Decentralized Key Management Mechanism for VANET with Blockchain," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5836–5849, 2020.

[26] Y. Inedjaren *et al.*, "Blockchain-based Distributed Management System for Trust in VANET," *Vehicular Communications*, vol. 30, p. 100350, 2021.

[27] F. A. Ghaleb *et al.*, "Misbehavior-Aware on-Demand Collaborative Intrusion Detection System Using Distributed Ensemble Learning for VANET," *Electronics*, vol. 9, no. 9, p. 1411, 2020.

[28] M. Bayat *et al.*, "NERA: A New and Efficient RSU based Authentication Scheme for VANETs," *Wireless Networks*, vol. 26, pp. 3083–3098, 2020.

[29] X. Wang *et al.*, "An Overview of 3GPP Cellular Vehicle-to-Everything Standards," *GetMobile: Mobile Computing and Communications*, vol. 21, no. 3, pp. 19–25, 2017.

[30] J. Cui *et al.*, "Extensible Conditional Privacy Protection Authentication Scheme for Secure Vehicular Networks in a Multi-Cloud Environment," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1654–1667, 2019.

[31] M. N. Mejri *et al.*, "Survey on VANET Security Challenges and Possible Cryptographic Solutions," *Vehicular Communications*, vol. 1, no. 2, pp. 53–66, 2014.

[32] D. Dolev and A. Yao, "On The Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[33] D. Chaum and T. P. Pedersen, "Wallet Databases with Observers," in *Annual international cryptology conference*. Springer, 1992, pp. 89–105.

[34] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," in *Conference on The Theory and Application of Cryptographic Techniques*. Springer, 1986, pp. 186–194.

[35] E. Bresson *et al.*, "Provably Secure Authenticated Group Diffie-Hellman Key Exchange," *ACM Transactions on Information and System Security (TISSEC)*, vol. 10, no. 3, pp. 10–es, 2007.

[36] A. Antipa *et al.*, "Accelerated Verification of ECDSA Signatures," in *International Workshop on Selected Areas in Cryptography*. Springer, 2005, pp. 307–318.

[37] S. Shunmuganathan, “A Reliable Lightweight Two Factor Mutual Authenticated Session Key Agreement Protocol for Multi-Server Environment,” *Wireless Personal Communications*, vol. 121, no. 4, pp. 2789–2822, 2021.

[38] Y. Zhou *et al.*, “A Secure Authentication and Key Agreement Scheme with Dynamic Management for Vehicular Networks,” *Connection Science*, vol. 35, no. 1, p. 2176825, 2023.

[39] M. Nikooghadam and H. Amintoosi, “Secure Communication in CloudIoT Through Design of A Lightweight Authentication and Session Key Agreement Scheme,” *International Journal of Communication Systems*, vol. 36, no. 1, p. e4332, 2023.

[40] A. Yang *et al.*, “Delegating Authentication to Edge: A Decentralized Authentication Architecture for Vehicular Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1284–1298, 2020.

[41] E. Käsper, “Fast Elliptic Curve Cryptography in OpenSSL,” in *Financial Cryptography and Data Security: FC 2011 Workshops, RLCPS and WECSR 2011, Rodney Bay, St. Lucia, February 28-March 4, 2011, Revised Selected Papers 15*. Springer, 2012, pp. 27–39.

[42] L. Ben, “On The Implementation On Pairing-Based Cryptosystems,” *Department of Computer Science, Stanford University Ph. D. Thesis*, 2007.

[43] J. W. Bos *et al.*, “Elliptic Curve Cryptography in Practice,” in *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer, 2014, pp. 157–175.

[44] M. Azees *et al.*, “EAAP: Efficient Anonymous Authentication with Conditional Privacy-Preserving Scheme for Vehicular Ad Hoc Networks,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 9, pp. 2467–2476, 2017.

Sujash Naskar (Member, IEEE) received a B.Sc. degree in computer science in 2017 and an M.Sc. degree in 2019 from University of Calcutta, Kolkata, India. He is currently pursuing a Ph.D. degree in IoT Security and Privacy at STC research group, Mid Sweden University, Sundsvall, Sweden.

From November 2022 to July 2023, he was a guest research fellow at Simula UiB, Bergen, Norway. His research interests include privacy and security management for Vehicular ad hoc Networks (VANET), especially developing lightweight privacy-preserving authentication schemes for VANET.

Mr. Sujash's awards and honors include receiving the prestigious Ericsson Grant for a long-term research visit.

Carlo Brunetta received his BSc and MSc in mathematics from University of Trento, Trento, Italy, in 2014 and 2016 respectively. He obtained a PhD in cryptology and privacy-preserving protocols from Chalmers University of Technology, Gothenburg, Sweden, in 2021. From 2021 to 2023, he was a researcher in theoretical cryptography and side-channel attacks in cryptography-oriented applications at Simula UiB, Bergen, Norway. He is currently an independent researcher in several aspects of cryptology, applied and not.

Gerhard Hancke (M'1999,SM'2011,F'2022) is a Professor in the Department of Computer Science at City University of Hong Kong. He received B.Eng and M.Eng degrees in Computer Engineering from the University of Pretoria, South Africa, in 2002 and 2003, and a PhD in Computer Science from the University of Cambridge, United Kingdom, in 2009. Previously he worked as a researcher with the Smart Card and IoT Security Centre and as teaching fellow with the Department of Information Security, both at Royal Holloway, University of London. His research interests are in security, reliable communication and distributed sensing for the Industrial Internet-of-Things.

Tingting Zhang received the B.Sc. degree and M.Sc. degree in computer science and engineering from Fudan University, Shanghai, China, in 1982 and 1984, respectively, and the Ph.D. degree in computer science and engineering from Linköping University, Linköping, Sweden, in 1993. She is a Professor of computer engineering with the Department of Information and Communication Systems, Mid Sweden University, Sundsvall, Sweden. Her current research work is in the areas of wireless sensor networks and security.

Mikael Gidlund (M'98-SM'16) received the Licentiate of Engineering degree in radio communication systems from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2004, and the Ph.D. degree in electrical engineering from Mid Sweden University, Sundsvall, Sweden, in 2005. From 2008 to 2015, he was a Senior Principal Scientist and Global Research Manager of Wireless Technologies with ABB Corporate Research, Västerås, Sweden. From 2007 to 2008, he was a Project Manager and a Senior Specialist with Nera Networks AS, Bergen, Norway. From 2006 to 2007, he was a Research Engineer and a Project Manager with Acreo AB, Hudiksvall, Sweden. Since 2015, he has been a Professor of Computer Engineering at Mid Sweden University. He holds more than 20 patents (granted and pending) in the area of wireless communication. His current research interests include wireless communication and networks, wireless sensor networks, access protocols, and security. Dr. Gidlund is an Associate Editor of the *IEEE Transactions on Industrial Informatics.* and *IEEE Journal of Emerging and Selected Topics in Industrial Electronics*