AuthUp: A Privacy-Preserving V2I Authentication Scheme for Vehicular Ad Hoc Networks

Xincheng Li, Xinchun Yin, Jianting Ning *Member, IEEE,* and Tianwei Zhang *Member, IEEE,*

***Abstract*—Vehicular ad hoc networks (VANETs) play a crucial role in enhancing modern transportation systems, leading to the development of various authentication schemes to meet the stringent security and privacy requirements of vehicular communications. Many existing schemes rely on tamper-proof devices (TPDs) that store extensive secret parameters for identity authentication, making them susceptible to Sybil and side-channel attacks. This paper introduces a sophisticated privacy-preserving authentication protocol, referred to as Authentication and Update (AuthUp), specifically designed to solve the problem. A trusted authority (TA) issues one-time access tokens to vehicles during the registration process. After each vehicle-to-infrastructure (V2I) authentication process, cryptographic key material is updated without compromising privacy. Consequently, Sybil attacks are promptly identified and thwarted, as manipulated tokens fail authentication. Additionally, adversaries are prevented from accessing sensitive data stored in TPDs via side-channel attacks because of the dynamic updating of secret keys. A formal security proof validates the robustness of AuthUp, and extensive evaluations demonstrate its superior efficiency compared to alternative schemes, highlighting its suitability for VANET deployment.**

***Index Terms*—Vehicular ad hoc network (VANET), Sybil attack, security, privacy-preserving, authentication.**

I. Introduction

THE deployment of vehicular ad hoc networks (VANETs) [1], [2] holds the potential to significantly enhance traffic efficiency while affording considerable convenience to pedestrians and drivers through the widespread propagation of time-critical traffic information. In compliance with the dedicated short-range communications (DSRC) standard, vehicles cyclically transmit beacon messages (e.g., every 100 ms) through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication channels. Upon reception of identification and contextual data (e.g., vehicle location, speed, braking status, etc.), receivers promptly adapt their movements. VANETs facilitate the implementation of various applications, encompassing safety-critical functions (e.g., collision avoidance, emergency braking) and value-added services (e.g., toll payment, traffic announcement dissemination, entertainment).

Security and privacy are fundamental concerns that hinder the large-scale practical deployment of VANET [3]. Open-access wireless channels in VANET are susceptible to various attacks, i.e., modification, impersonation, illusion, replay, and collision induction [4]. Fabricated or tampered messages can lead to traffic congestion, accidents, or even fatalities for high-speed vehicles. Preserving the privacy of vehicle users is another critical challenge. Location-based services necessitate vehicles continuously transmitting their location data to surrounding entities or a remote server as needed. However, they are reluctant to share sensitive information with third parties, especially malicious adversaries. Protecting their private information and minimizing the transmitted data would be highly beneficial.

A. Motivation

Among the existing privacy-preserving authentication schemes for VANETs [5], [6], vehicles typically pre-store extensive pseudonyms and key pairs for signature generation to ensure identity privacy through redundancy. However, malicious or compromised vehicles can exploit this redundancy to deceive service providers, gaining unfair advantages. This exploitation is commonly referred to as Sybil attacks [7], [8]. Douceur [7] introduced the concept of Sybil attacks, where attackers forge multiple pseudonymous identities to disrupt the reputation system of services in open-access networks. Consequently, service providers or authorities may incur significant economic losses if the threat of Sybil attacks is not effectively mitigated. Moreover, the verification of certificate revocation lists (CRLs) is costly, even though abused certificates are revoked by a trusted authority (TA).

Another formidable challenge that poses a significant threat to the safety of drivers and their property is side-channel attacks [9]. Typically, each on-board unit (OBU) is equipped

Manuscript received xx, xxxx; revised xx, xxxx. This work was supported in part by the National Natural Science Foundation of China under Grants 61972094, 62032005, in part by the Open Fund of Henan Key Laboratory of Network Cryptography Technology under Grant LNCT2022-A17, in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX22_3501, in part by the Yangzhou University International Academic Exchange Fund under Grant YZUF2023106, and in part by the Young Talent Promotion Project of Fujian Science and Technology Association. *(Corresponding author: Xinchun Yin; Jianting Ning.)*

Xincheng Li is with the College of Information Engineering, Yangzhou University, Yangzhou 225127, China, and also with the Henan Key Laboratory of Network Cryptography Technology, Information Engineering University, Zhengzhou 450001, China (e-mail: 18752782261@163.com).

Xinchun Yin is with Guangling College, Yangzhou University, Yangzhou 225128, China, the College of Information Engineering, Yangzhou University, Yangzhou 225127, China, and also with the Henan Key Laboratory of Network Cryptography Technology, Information Engineering University, Zhengzhou 450001, China (e-mail: xcyin@yzu.edu.cn)

Jianting Ning is with the Key Laboratory of Analytical Mathematics and Applications (Ministry of Education), Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China, and also with the Faculty of Data Science, City University of Macau, Macau 999078, China (e-mail: jtning88@gmail.com).

Tianwei Zhang is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: tianwei.zhang@ntu.edu.sg).

with a tamper-proof device (TPD) to safeguard private secrets from being compromised. However, physical cryptosystems inadvertently and continuously leak sensitive information, such as timing, power consumption, and electromagnetic emissions. Side-channel attacks exploit these leaks to extract secret information using either single-shot or multi-shot traces [10]. Typically, single-shot attacks are limited by factors such as noise, the attack environment, and network conditions [11]. In this paper, we focus on more advanced and powerful multi-shot attacks. Once adversaries gain access to the private secrets stored in OBUs, launching Sybil attacks becomes effortless. Hence, the development of an effective authentication scheme capable of withstanding side-channel attacks holds paramount importance in practical applications.

Several existing works have explored Sybil attack-resistant authentication schemes for VANETs [12], [13], [14], [15], [16], [17], [18], [19], [20], [21]. However, these schemes either lack resilience against the aforementioned side-channel attacks or exhibit limited flexibility for VANETs due to their high computational and communication overheads. Given the significant threat posed by both Sybil attacks and side-channel attacks to the VANET environment, our objective is to *devise a secure and privacy-preserving V2I authentication scheme for VANETs that can effectively combat Sybil attacks and side-channel attacks simultaneously.*

B. Contributions

To fulfill our objective, we propose a robust and privacy-preserving V2I authentication scheme named Authentication and Update (AuthUp). In AuthUp, onboard units (OBU) can anonymously submit requests for value-added services without storing any long-term secret key. Upon achieving successful V2I mutual authentication, the trusted authority (TA) authorizes a new local secret key pair and access token using advanced updatable cryptographic techniques and chameleon hash (CH). Due to the vulnerability of long-term keys to side-channel attacks, this dynamic update process, known as key rotation, significantly enhances AuthUp's ability to resist such attacks. Notably, each token authorized for a specific OBU possesses the ability to be updated and verified only a limited number of times. Moreover, the security features of existential unforgeability under chosen message attacks (EUF-CMA) and identity traceability embedded within our proposed scheme ensure the detection of Sybil attacks. The main contributions of our work can be summarized as follows:

1) A secure and privacy-preserving authentication scheme named AuthUp is proposed for VANET. The technologies of CH and UPKE are utilized to facilitate one-time token authorization and authentication. The EUF-CMA security and dynamic user key rotation empower AuthUp to effectively withstand Sybil and side-channel attacks simultaneously.
2) The rigorous security proof of our AuthUp scheme shows that it intuitively captures EUF-CMA security in the random oracle model. Furthermore, we provide detailed illustrations demonstrating how AuthUp satisfies fundamental security and privacy-preserving requirements, ensuring its practicality and feasibility in real-world deployment scenarios.
3) We conduct extensive benchmarking of various cryptographic operations and provide a comprehensive comparison between AuthUp and existing anonymous authentication schemes for VANETs in terms of computational and communication overhead. Experimental results conclusively demonstrate the superior efficiency and feasibility of AuthUp compared to alternative schemes.

C. Organization

The remainder of this paper is structured as follows: in Section II, we review existing schemes designed to resist Sybil attacks and side-channel attacks. Section III provides an overview of essential building blocks relevant to AuthUp. Subsequently, Section V defines the syntax, system model, and other key concepts. Section IV elaborates on the detailed design of AuthUp, outlining its key components and functionalities. We proceed to present the security proof of AuthUp in Section VI, followed by the performance evaluation of AuthUp in Section VII. Finally, Section VIII offers concluding remarks on our work and future works.

II. Related works

A. Sybil attack resistance in VANET

The Sybil attack detection and resistance scheme can be classified into three main categories: cryptography-based authentication approaches, location verification-based approaches, and trajectory-based approaches.

1) Cryptography-based authentication approaches: Cryptography-based authentication approaches are centered around constructing authentication schemes using various cryptographic components to detect Sybil attacks by authenticating messages and vehicles.

Wu et al. [12] proposed a controllable message-linkable scheme leveraging group signature technology. Group signature technology enables group members to authenticate anonymously, while the message endorser can be revealed to prevent Sybil attacks if duplicate signatures are generated on the same message. However, this scheme may fail to detect collusion among malicious vehicles signing messages with switched credentials. Zhou et al. [14] introduced a lightweight and privacy-preserving Sybil attack resistance scheme where the department of motor vehicle (DMV) manages anonymous certificate distribution, vehicle registration, tracking, and administrative policies. Pseudonyms from the same vehicle can be detected by both road-side units (RSUs) and the DMV using a two-stage collusion-resistant hash function, ensuring anonymity throughout the detection process. Nonetheless, collusion attacks and pseudonym exchange can still evade detection in this scheme. Lin [13] developed a local Sybil resistance (LSR) scheme aimed at resisting Sybil attacks. This technique employs group signatures for event report generation and verification, linking vehicles that generate multiple signatures on the same event to detect Sybil attacks. However, both schemes are susceptible to potential malfunction, as malicious

vehicles might forge different signatures with fabricated events to their advantage. Moreover, the broadcast method employed in these schemes complicates the differentiation of Sybil attacks, particularly in high-density traffic scenarios.

2) Location verification-based approaches: Location verification-based approaches [15], [16], [17], [19], [18] leverage physical measurements using hardware equipment such as radars and special sensors to distinguish Sybil nodes occupying the same position. While effective, these approaches are often associated with high costs in terms of hardware, computation, and bandwidth due to their reliance on additional equipment.

In [16], the authors proposed a lightweight Sybil attack detection solution for VANET based on received signal strength indicator (RSSI) and Voiceprint. Another similar approach named PCISAD was proposed by Yao et al. [17]. Both of these methods are capable of identifying malicious Sybil nodes by gathering and comparing received RSSI time series, even in scenarios where vehicles have the ability to manipulate their transmission power intensity. However, it's worth noting that the detectors in both schemes may still be susceptible to deception by malicious vehicles equipped with multiple radios. More recently, Benadla et al. [18] introduced a collaborative Sybil attack detection approach utilizing blockchain technology for vehicular fog computing (VFC) networks. In this approach, the positions of vehicles are initially authenticated by fog nodes using the RSSI technique. Subsequently, fog nodes generate a proof of position and store it in a blockchain for real-time detection. These proofs collectively form a trajectory of a vehicle, thereby mitigating Sybil vulnerability from a global perspective.

3) Trajectories-based approaches: Trajectory-based approaches in VANET utilize a vehicle's trajectory as a pseudonym for identity authentication. A trajectory comprises multiple position and timestamp pairs along the road, providing a unique signature for each vehicle's movement pattern. These trajectories are challenging to fabricate as RSUs provide witnesses of position and timestamp for vehicles, ensuring the integrity of trajectory information. However, multiple vehicles sharing the same trajectories may receive the same authorized messages, complicating the detection process. Additionally, any security issue such as RSU compromise may lead to complete privacy disclosure by querying a vehicle's trajectory history, as trajectories inherently contain the user's private information. Furthermore, a compromised RSU can generate numerous indistinguishable valid trajectories for vehicles, which is especially problematic in time-sensitive traffic-related applications.

One notable trajectory-based scheme for VANETs is Footprint [20]. In Footprint, RSUs distribute signatures on location-hidden authorized messages, constructing location-hidden trajectories. Verifiers can then authenticate the legitimacy of an anonymous trajectory without compromising privacy. Legitimate trajectories consist of multiple messages signed by the same RSU during the same period, enabling the detection of Sybil nodes attempting to forge trajectories. Baza et al. [21] proposed a Sybil attack-resistant scheme based on proof of work (PoW) and location. Specifically, RSUs, in conjunction with vehicle trajectories, are tasked with signing and disseminating time-stamped tags as evidence of vehicles' anonymous locations. This process establishes a verifiable trajectory through the collaboration of multiple RSUs positioned alongside the roads, employing a threshold signature scheme. Meanwhile, vehicles are required to compute a solution to a PoW puzzle to generate randomness for the subsequent RSU. Due to the impracticality of malicious vehicles compromising multiple RSUs simultaneously, the scheme effectively mitigates Sybil attacks. However, the PoW poses a significant computational burden on vehicles and may result in intolerable delays. Similarly, the compromise of a single RSU exposes vehicles' privacy to adversaries without limitation. Furthermore, the length of a trajectory increases linearly as the vehicle progresses forward.

B. Side-channel attack resistance in VANET

Side-channel attacks allow adversaries to observe and analyze various exploited characteristics to recover sensitive information using both cryptographic and non-cryptographic techniques [10], [11]. Typically, timing side-channel attacks [22] (e.g., cache timing attacks, branch timing attacks, meltdown) exploit the time a device takes to perform certain operations to extract sensitive information. Power side-channel attacks [23], including simple power analysis (SPA), differential power analysis (DPA), and correlation power analysis (CPA), leverage power consumption statistics to extract sensitive information. Other attacks, such as electromagnetic side-channels, acoustic side-channels, and optical side-channels, also pose significant security threats.

To resist side-channel attacks in VANET, some schemes leverage the physical unclonable function (PUF) [24] in prior research [25], [26]. PUF is a hardware function implemented within a circuit that captures inherent process parameter deviations during chip manufacturing. The unique characteristics of uniqueness and randomness facilitate a distinctive correspondence between challenge and response signals, thus fortifying the schemes against side-channel attacks. However, the reliance on hardware entails significant economic costs.

Another strategy to mitigate side-channel attacks involves the periodic updating of security parameters stored in OBUs, as discussed in previous schemes [27], [28], [29]. In [27], a privacy-preserving TPD-based authentication scheme was proposed for cloud-based VANETs. An offline self-updating method is employed to periodically update the data in the TPD to enhance resistance against side-channel attacks. Additionally, a two-layer error location mechanism is introduced to detect invalid signatures within problematic aggregate signature scenarios. In [29], a novel message authentication code (MAC)-based authentication scheme named NoMAS was proposed to withstand side-channel attacks. Specifically, NoMAS offers simultaneous hard key update (HKU) and soft key update (SKU) to prevent data leakages. SKU is conducted regularly by OBUs using the original system key received from a Registration Authority (RA). In the event of an attacker accessing the system key, HKU is initiated by the RA.

In summary, the self-update mechanism serves to deter adversaries from pilfering long-term security parameters stored

in OBUs. However, due to the challenging detection of coarse-grained self-updates aimed at preserving privacy, malicious vehicles orchestrating Sybil attacks pose a formidable challenge to withstand.

III. Preliminaries

This section mainly introduces some preliminaries, including updatable message authentication code (UMAC), updatable public key encryption (UPKE), and chameleon hash (CH).

A. Updatable message authentication code

The concept of UMAC was first proposed by Cini et al. [30], emphasizing sound key management practices and the periodic switching of keys.

Definition 1: **(Updatable message authentication code, UMAC).** A UMAC scheme $\mathcal{UMAC}$ is a tuple of five probabilistic polynomial-time (PPT) algorithms (Setup, Next, Sig, Update, Ver):

1) $\mathcal{UMAC}$.Setup$(1^\lambda) \rightarrow (pp, k)$: Given a security parameter $\lambda \in \mathbb{N}$, Setup outputs public parameters pp and a key k.
2) $\mathcal{UMAC}$.Next$(k_e) \rightarrow (k_{e+1}, \delta_{e+1})$: The key update algorithm takes in a key k_e for epoch e. It outputs a new key k_{e+1} for epoch $e+1$ and an update token δ_{e+1}.
3) $\mathcal{UMAC}$.Sig$(k_e, M) \rightarrow \sigma_e$: Sig takes in a key k_e and a message $M \in \mathcal{M}$ to output a tag σ_e.
4) $\mathcal{UMAC}$.Update$(\delta_{e+1}, \sigma_e) \rightarrow \sigma_{e+1}$: Update receives an update token δ_{e+1} and a tag σ_e. It outputs an updated tag σ_{e+1}.
5) $\mathcal{UMAC}$.Ver$(k_e, M, \sigma_e) \rightarrow b$: Ver receives a key k_e, a message M, and a tag σ_e. It outputs a verdict $b \in \{0, 1\}$.

Definition 2: **(Correctness of UMAC).** Let $\lambda, n \in \mathbb{N}$ and $(pp, k) \leftarrow \mathcal{UMAC}$.Setup$(1^\lambda)$. Define $(k_{e+1}, \delta_{e+1}) \leftarrow \mathcal{UMAC}$.Next$(k_e)$ and $\sigma_{e+1} \leftarrow \mathcal{UMAC}$.Update$(\delta_{e+1}, \sigma_e)$ for $e \in [n-1]$. A UMAC scheme provides correctness if, for all $M \in \mathcal{M}$ and σ_e with $\mathcal{UMAC}$.Ver$(k_e, M, \sigma_e) = 1$: Pr$[\mathcal{UMAC}$.Ver$(k_{e+1}, M, \sigma_{e+1}) = 1] = 1$.

B. Updatable public key encryption

The notion of UPKE was introduced by Jost et al. [31] and further extended by Alwen et al. [32] and Dodis et al. [33]. Below, we give the syntax of UPKE in [33].

Definition 3: **(Updatable public key encryption, UPKE).** An UPKE scheme $\mathcal{UPKE}$ is a tuple of six PPT algorithms (Setup, KeyGen, Enc, Dec, Upd-Pk, Upd-Sk):

1) $\mathcal{UPKE}$.Setup$(1^\lambda) \rightarrow (pp)$: Given a security parameter $\lambda \in \mathbb{N}$, Setup outputs public parameters pp.
2) $\mathcal{UPKE}$.KeyGen$(pp) \rightarrow (sk_0, pk_0)$: KeyGen takes in pp and outputs a fresh key pair (sk_0, pk_0).
3) $\mathcal{UPKE}$.Enc$(pk_e, m) \rightarrow c$: Given public key pk_e and a message $m \in \mathcal{M}$, Enc produces a ciphertext c.
4) $\mathcal{UPKE}$.Dec$(sk_e, c) \rightarrow m/\perp$: Dec inputs a secret key sk_e and a ciphertext c to produce message m or $\perp$.
5) $\mathcal{UPKE}$.Upd-Pk$(pk_e, up_{e+1}) \rightarrow pk_{e+1}$: Upd-Pk takes in the public key pk_e and an update ciphertext up_{e+1} to produce a new public key pk_{e+1}.

Exp$^{IND-CPA}_{\mathcal{A},\mathcal{UPKE}}(1^\lambda)$
$pp \leftarrow \mathcal{UPKE}$.Setup$(1^\lambda)$;
$(sk_0, pk_0) \leftarrow \mathcal{UPKE}$.KeyGen$(pp), b \leftarrow \{0, 1\}$;
$(m_0^*, m_1^*, state) \leftarrow \mathcal{A}(pk_0)$;
$c^* \leftarrow \mathcal{UPKE}$.Enc$(pk_l, m_b^*)$; // l is the current epoch
$state \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Update}}(\cdot)}(c^*, state)$;
$b' \leftarrow \mathcal{A}(pk^*, sk^*, up^*, state)$;
return 1 iff $b = b'$.

Oracle $\mathcal{O}_{\text{Update}}(pk_e)$
Randomly choose up_{e+1};
$(pk_{e+1}) \leftarrow \mathcal{UPKE}$.Upd-Pk$(pk_e, up_{e+1})$;
$(sk_{e+1}) \leftarrow \mathcal{UPKE}$.Upd-Sk$(sk_e, up_{e+1})$;
return pk_{e+1}.

Fig. 1: Experiment of IND-CPA security in $\mathcal{UPKE}$.

6) $\mathcal{UPKE}$.Upd-Sk$(sk_e, up_{e+1}) \rightarrow sk_{e+1}$: Upd-Sk takes in the secret key sk_e and an update ciphertext up_{e+1} to produce a new secret key sk_{e+1}.

Definition 4: **(Correctness of UPKE).** Let $\lambda, n \in \mathbb{N}$, $pp \leftarrow \mathcal{UPKE}$.Setup$(1^\lambda)$, $(sk_0, pk_0) \leftarrow \mathcal{UPKE}$.KeyGen$(pp)$. Define $(pk_{e+1}) \leftarrow \mathcal{UPKE}$.Upd-Pk$(pk_e, up_{e+1})$, $sk_{e+1} \leftarrow \mathcal{UPKE}$.Upd-Sk$(sk_e, up_{e+1})$, for $e \in [n-1]$. A UPKE scheme provides correctness if, for all message $m \in \mathcal{M}$ and all $j \in [n]$: Pr$[\mathcal{UPKE}$.Dec$(sk_j, \mathcal{UPKE}$.Enc$(pk_j, m)) = m] = 1$.

Definition 5: **(IND-CPA security of UPKE).** The indistinguishability under chosen-plaintext attacks (IND-CPA) security of a UPKE scheme $\mathcal{UPKE}$ is based on the experiment defined in Fig. 1.

An UPKE scheme $\mathcal{UPKE}$ is IND-CPA secure, if for any PPT adversary $\mathcal{A}$, advantage $Adv^{IND-CPA}_{\mathcal{A},\mathcal{UPKE}} = Pr[Exp^{IND-CPA}_{\mathcal{A},\mathcal{UPKE}}(1^\lambda) = 1]$ is negligible.

C. Chameleon hash

Chameleon hash (CH), also referred to as a trapdoor hash function, enables an individual with a trapdoor to generate arbitrary collisions within the domain of the function. This concept was introduced by Krawczyk and Rabin [34] and builds upon the notion of chameleon commitments [35].

Definition 6: **(Chameleon hash, CH).** A CH scheme $\mathcal{CH}$ is a tuple of five PPT algorithms (Setup, KeyGen, Hash, Verify, Adapt):

1) $\mathcal{CH}$.Setup$(1^\lambda) \rightarrow pp$: Setup takes in security parameter $\lambda \in \mathbb{N}$ and outputs public parameters pp.
2) $\mathcal{CH}$.KeyGen$(pp) \rightarrow (sk, pk)$: Given public parameters pp, KeyGen outputs a key pair (sk, pk).
3) $\mathcal{CH}$.Hash$(pk, m, r) \rightarrow h$: Hash takes in a public key pk, a message $m \in \mathcal{M}$, and a random coin r to produce a chameleon hash value h.
4) $\mathcal{CH}$.Verify$(pk, m, h, r) \rightarrow b$: Verify takes in a public key pk, a message m, a hash value h, and a random coin r to output a bit $b \in \{0, 1\}$.
5) $\mathcal{CH}$.Adapt$(sk, m, r, m') \rightarrow r'$: Adapt receives a secret key sk, a message m, a random coin r, and a message $m' \in \mathcal{M}$ and outputs a randomness r'.

TABLE I: The main notations defined in AuthUp

Notations	Definition
TA	Trusted authority
RSU	Road-side unit
OBU	On-board unit
λ	A security parameter
$\mathbb{G}, q$	A cyclic multiplication group with order q
g	A generator of group $\mathbb{G}$
(msk, mpk)	Master key pair of TA
ID_r, sk_r, pk_r	Key pair of RSU with the identity ID_r
ID_v, sk_i, pk_i	Key pair of vehicle with the identity ID_v in time interval i
$Cert_r$	A PKI-based certificate for RSU with ID_r
k_r	A secret key of RSU with ID_r
n	The access time an OBU is authorized
$HMAC$	A general HMAC function
H_1, H_2, H_3	Three collision-resistant hash functions
$tk_i = \{\alpha_i, r_i, r_{i+1}\}$	Access token for interval i
c_{tk}	Ciphertext of tokens
req_i	A request for server in time interval i
$\mathcal{L}$	A list storing information of vehicles
σ_r	An updatable tag
$\mathcal{L}_{HMAC}, \mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{user}, \mathcal{L}_{corr}, \mathcal{L}_{tk}$	Lists store the outputs of $HMAC, H_1, H_2$ functions, and information about legitimate users, corrupted users, and tokens
$q_{H_1}, q_{H_2}, q_{HMAC}, q_{Setup_v}, q_{corrupt}, q_{TKGen}$	The times to invoke $\mathcal{O}_{\mathrm{H}_1}, \mathcal{O}_{\mathrm{H}_2}, \mathcal{O}_{\mathrm{HMAC}}, \mathcal{O}_{\mathrm{Setup_v}}, \mathcal{O}_{\mathrm{corrupt}}$, and $\mathcal{O}_{\mathrm{TKGen}}$

Definition 7: **(Correctness of CH).** Let $\lambda \in \mathbb{N}$, $pp \leftarrow \mathcal{CH}.\text{Setup}(1^\lambda)$, and $(sk, pk) \leftarrow \mathcal{CH}.\text{KeyGen}(pp)$. A CH provides correctness if, for any $m, m' \in \mathcal{M}$ and $r' \leftarrow \mathcal{CH}.\text{Adapt}(sk, m, r, m')$: $\Pr[\mathcal{CH}.\text{Verify}(pk, m, h, r) = \mathcal{CH}.\text{Verify}(pk, m', h, r') = 1] = 1$.

Definition 8: **(Collision resistance of CH).** Let $\lambda \in \mathbb{N}$, $pp \leftarrow \mathcal{CH}.\text{Setup}(1^\lambda)$, and $(sk, pk) \leftarrow \mathcal{CH}.\text{KeyGen}(pp)$. A CH function provides collision resistance if the probability $\Pr[(h, (m_0, r_0), (m_1, r_1)) \leftarrow \mathcal{A}(pk) : \mathcal{CH}.\text{Verify}(pk, m_0, h, r_0) = \mathcal{CH}.\text{Verify}(pk, m_1, h, r_1) = 1 \wedge (m_0 \neq m_1)]$ is negligible.

IV. Scheme Overview

A. Notation and Syntax

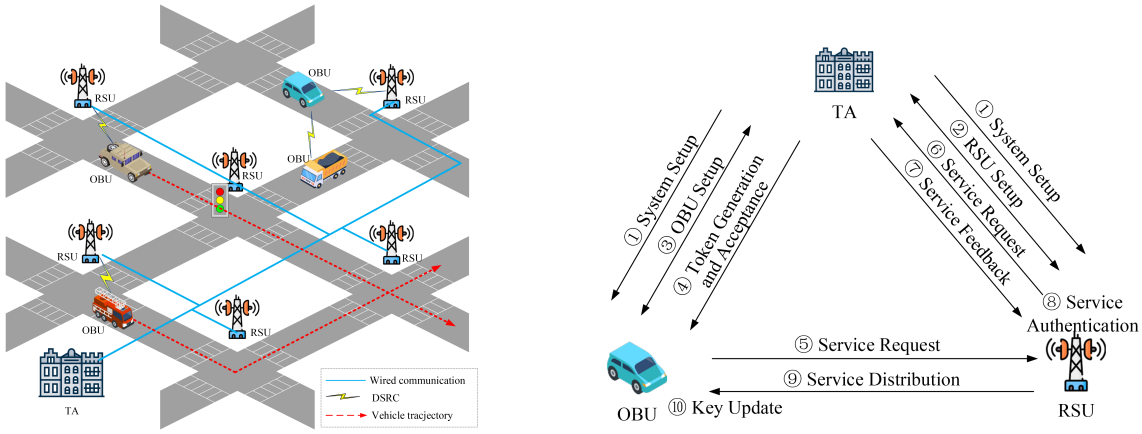
The main notations used in AuthUp are defined in Table I. Formally, AuthUp consists of nine algorithms, i.e., Setup, Setup_r, Setup_v, TKGen, TKAcc, Request, Feedback, Authentication, and KeyUpdate.

1) $\text{AuthUp.Setup}(1^\lambda) \rightarrow (pp, msk, mpk, \mathcal{L})$: Given a security parameter λ, TA runs the probabilistic Setup algorithm to produce public parameters pp, a master key pair (msk, mpk), and an empty user list $\mathcal{L}$.
2) $\text{AuthUp.Setup}_\text{r}(pp, ID_r) \rightarrow (sk_r, pk_r, Cert_r, k_r)$: Given public parameters pp and the identity of RSU ID_r, TA runs the probabilistic RSU setup algorithm to output a key pair (sk_r, pk_r), a certificate $Cert_r$ based on public key infrastructure (PKI), and a secret token k_r for RSU.
3) $\text{AuthUp.Setup}_\text{v}(pp, ID_v) \rightarrow (sk_i, pk_i)$: Given public parameters pp and the identity of OBU ID_v, OBUs run the probabilistic vehicle setup algorithm to output a key pair (sk_i, pk_i).
4) $\text{AuthUp.TKGen}(ID_v, pk_i, n, msk, mpk, \mathcal{L}) \rightarrow (c_{tk}, \mathcal{L}')$: Given the identity of vehicle ID_v, the public key of vehicle pk_i, a maximum access time n, master key pair (msk, mpk), and user list $\mathcal{L}$, TA runs the probabilistic TKGen algorithm to produce a ciphertext of token c_{tk} and an updated user list $\mathcal{L}'$.
5) $\text{AuthUp.TKAcc}(sk_i, c_{tk}) \rightarrow tk$ / $\perp$: Given the secret key of vehicle sk_i and a ciphertext of token, OBU runs the deterministic token acceptance algorithm to output a token tk or $\perp$.
6) $\text{AuthUp.Request}(sk_i, pk_i, tk_i, mpk) \rightarrow req_i$: Given a key pair (sk_i, pk_i), token tk_i, and master public key mpk, OBU runs the probabilistic service request algorithm to output a request req_i.
7) $\text{AuthUp.Feedback}(req_i, msk) \rightarrow (\sigma_r, c)$ / $\perp$: The probabilistic service feedback algorithm is run by TA. Given a request req_i and a master secret key msk as input, the algorithm outputs an updatable tag σ_r and ciphertext c.
8) $\text{AuthUp.Authentication}(k_r, \sigma_r) \rightarrow b$: The probabilistic service authentication algorithm is run by RSU. Given the secret key of RSU k_r and an updatable tag σ_r as input, the algorithm outputs a verdict $b \in \{0, 1\}$, indicating the validity of the response from TA.
9) $\text{AuthUp.KeyUpdate}(sk_i, pk_i, c) \rightarrow tk_{i+1}$ / $\perp$: The deterministic key update algorithm is run by OBU. Given a secret key sk_i, a public key pk_i, and a ciphertext c as input, the algorithm outputs a new key pair and an updated access token tk_{i+1} or $\perp$.

B. System model

As shown in Fig. 2, the main components in AuthUp consist of a trusted authority (TA), RSUs, and OBUs. The responsibilities of each entity are shown as follows:

- TA represents a trusted authority or enterprise that provides diverse value-added vehicular services to vehicles that have subscribed to the services.
  1) TA initializes the AuthUp system and distributes public parameters to RSUs and vehicles (①).
  2) Upon receiving a setup request from either an RSU or an OBU, TA is responsible for verifying the legitimacy of these entities and initializing key pairs and certificates (specific to RSUs) for legitimate RSUs and OBUs (② and ③). Meanwhile, the TA generates access tokens for OBUs, which are authenticated by the OBUs and utilized to access value-added services in VANET (④).
  3) Once receiving a service request from an OBU (⑤ and ⑥), TA first authenticates the legitimacy of the request and then generates feedback to instruct the RSU to disseminate services if the request is valid (⑦). Subsequently, vehicular services, along with a key update message, will be distributed to the vehicle to update its access token (⑨).
- RSUs are honest-but-curious communication and vehicular service transmission stations. In AuthUp, RSUs relay service requests received from OBUs (⑥) and key update



(a) System model. (b) Main process flow.

Fig. 2: Overview of AuthUp.

messages from TA (⑨). They will deliver vehicular services to OBUs only if the feedback received from TA is positive (⑧).

- OBUs request value-added services once they pass through the communication range of a new RSU (⑤). With the existence of TPDs, OBUs can prevent adversaries from stealing secret keys. However, OBUs must update their keys periodically to withstand side-channel attacks (⑩).

C. Design goals

We aim to build a Sybil attack-resistant scheme for VANET. This scheme can not only provide secure and privacy-preserving authentication but also withstand potential Sybil attacks on VANET. Specifically, the following goals are crucial to keep in mind:

1) Message integrity and authenticity: Since the messages are broadcast on open-access wireless channels, integrity and authenticity must be guaranteed. Specifically, the scheme must be authenticable and unforgeable.
2) Conditional privacy-preserving: The contradiction between privacy-preserving and traceability must be appropriately resolved. Specifically, (1) no identity information can be inferred from broadcast messages (anonymity). (2) Adversaries cannot link any two messages sent from the same OBU (unlinkability). (3) Only TA can reveal the identity of a vehicle once accused (traceability).
3) Resist different attacks: The authentication scheme should have the ability to effectively resist various attacks, i.e., Sybil, side-channel, modification, and replay attacks.

D. Security model

CH is leveraged in the proposed scheme to prevent unauthorized service access. Specifically, the trapdoor of the CH serves as the system master secret key, enabling the generation of legitimate tokens for vehicles. We simplify the security model and notion of EUF-CMA security for AuthUp.

Definition 9: **(EUF-CMA security of AuthUp).** The EUF-CMA security of AuthUp is based on the probabilistic experiment in Fig. 3. AuthUp is EUF-CMA secure if, for any PPT adversary $\mathcal{A}$, the advantage $Adv^{EUF-CMA}_{\mathcal{A},AuthUp}(1^\lambda) = Pr[Exp^{EUF-CMA}_{\mathcal{A},AuthUp}(1^\lambda) = 1]$ is negligible.

$\mathbf{Exp}^{EUF-CMA}_{\mathcal{A},AuthUp}(1^\lambda)$
$(pp, msk, mpk) \leftarrow AuthUp.\text{Setup}(1^\lambda)$;
$\mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{HMAC}, \mathcal{L}_{user}, \mathcal{L}_{corr}, \mathcal{L}_{tk} \leftarrow \emptyset$;
$(pk_i, \alpha_i, r_i, r_{i+1}) \leftarrow \mathcal{A}^{\mathcal{O}}(pp, mpk)$;
return 1 iff the $(pk_i, \alpha_i, r_i, r_{i+1})$ passes authentication $\wedge pk_i = pk_i^* \wedge pk_i^* \notin \mathcal{L}_{corr}$.

Oracle $\mathcal{O}_{\text{H}_1}(i)$
$\tau_{i1} \leftarrow H_1(i)$;
$\mathcal{L}_{H_1} \leftarrow \mathcal{L}_{H_1} \cup \{(i, \tau_{i1})\}$;
return τ_{i1}.

Oracle $\mathcal{O}_{\text{H}_2}(pk_i, \alpha_i)$
$\tau_{i2} \leftarrow H_2(pk_i, \alpha_i)$;
$\mathcal{L}_{H_2} \leftarrow \mathcal{L}_{H_2} \cup \{(pk_i, \alpha_i, \tau_{i2})\}$;
return τ_{i2}.

Oracle $\mathcal{O}_{\text{HMAC}}(i)$
$j \leftarrow HMAC(msk, i)$;
$\mathcal{L}_{HMAC} \leftarrow \mathcal{L}_{HMAC} \cup \{(i, j)\}$;
return j.

Oracle $\mathcal{O}_{\text{Setup}_\text{v}}(ID_i)$
$(sk_i, pk_i) \leftarrow \mathcal{UPKE}.\text{KeyGen}(pp)$;
$\mathcal{L}_{user} \leftarrow \mathcal{L}_{user} \cup \{(ID_i, sk_i, pk_i)\}$;
return pk_i.

Oracle $\mathcal{O}_{\text{Corrupt}}(ID_i, pk_i)$
$\mathcal{L}_{corr} \leftarrow \mathcal{L}_{corr} \cup \{(ID_i, pk_i)\}$;
return sk_i.

Oracle $\mathcal{O}_{\text{TKGen}}(ID_i, pk_i)$
$(c_{tk}, *) \leftarrow AuthUp.\text{TKGen}(ID_i, pk_i, msk, mpk, *)$;
$\mathcal{L}_{tk} \leftarrow \mathcal{L}_{tk} \cup \{(ID_i, pk_i)\}$;
return c_{tk}.

Fig. 3: Experiment for EUF-CMA security of AuthUp.

V. Problem Formulation

A. High level description

The main intuition is to authorize vehicles with one-time tokens generated with CH and dynamically update their keys and tokens. Each time a vehicle passes through an RSU, it sends a service request containing the one-time token. RSU will only provide vehicular services if the request is validated by TA. Since the token can only be used once, tokens abused by Sybil nodes will be detected and denied by TA. Therefore, a vehicle can only achieve vehicular services within the communication range of a single RSU simultaneously. Note that since the vehicle cannot obtain a complete collision of CH, the confidentiality of the master secret key can be guaranteed [36], [37]. AuthUp consists of two phases, i.e., system initialization and service access. During system initialization, TA initializes the system and distributes keys to the participants. In the service access phase, vehicles request vehicular services from RSUs after undergoing legitimacy authentication.

B. System initialization

AuthUp.Setup$(1^\lambda) \rightarrow (pp, msk, mpk, \mathcal{L})$: Let $\mathcal{UMAC}$ = {Setup, Next, Sig, Update, Ver} be a UMAC scheme, and $\mathcal{UPKE}$={Setup, Enc, Dec, Upd-Pk, Upd-Sk} be a UPKE scheme, and $\mathcal{CH}$ = {Setup, KeyGen, Hash, Verify, Adapt} be a CH scheme.

The setup algorithm initializes public parameters of a UMAC $pp_{\mathcal{UMAC}} \leftarrow \mathcal{UMAC}$.Setup$(1^\lambda)$, a UPKE $pp_{\mathcal{UPKE}} \leftarrow \mathcal{UPKE}$.Setup$(1^\lambda)$, a CH $pp_{\mathcal{CH}} \leftarrow \mathcal{CH}$.Setup$(1^\lambda)$ and generates a master key pair $(sk, pk) \leftarrow \mathcal{CH}$.KeyGen$(pp_{\mathcal{CH}})$. Then, it picks three collision-resistant hash functions $H_1 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_2 : \mathbb{G} \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G} \times \mathbb{Z}_q^* \times \mathbb{G} \times \{0,1\}^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$, a HMAC function $HMAC : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, and sets a user list $\mathcal{L} \leftarrow \emptyset$. The algorithm outputs $pp = (pp_{\mathcal{UMAC}}, pp_{\mathcal{UPKE}}, pp_{\mathcal{CH}}, H_1, H_2, H_3, HMAC)$, a master key pair (msk, mpk), and an empty user list $\mathcal{L}$.

AuthUp.Setup$_\mathrm{r}(pp, ID_r) \rightarrow (sk_r, pk_r, Cert_r, k_r)$: The RSU setup algorithm generates a key pair $(sk_r, pk_r) \leftarrow \mathcal{UPKE}$.KeyGen$(pp_{\mathcal{UPKE}})$ and a PKI-based certificate $Cert_i$ for RSU with the identity ID_r. Meanwhile, it computes token $k_r = HMAC_{msk}(pk_r)$ for the RSU.

AuthUp.Setup$_\mathrm{v}(pp, ID_v) \rightarrow (sk_i, pk_i)$: The vehicle setup algorithm generates a key pair $(sk_i, pk_i) \leftarrow \mathcal{UPKE}$.KeyGen$(pp_{\mathcal{UPKE}})$ for the vehicle with the identity ID_v.

AuthUp.TKGen$(ID_v, pk_i, n, msk, mpk, \mathcal{L}) \rightarrow (c_{tk}, \mathcal{L}')$: The detailed token generation process is shown as Algorithm 1. A maximum access time of n may be authorized for the vehicle according to specific applications.

AuthUp.TKAcc$(sk_i, c_{tk}) \rightarrow tk_0 = \{\alpha_0, r_0, r_1\}$ or $\perp$: The vehicle may utilize its seckey key sk_i to decrypt the token ciphertext. The vehicle obtains $tk_0 = \{\alpha_0, r_0, r_1\} \leftarrow \mathcal{UPKE}$.Dec$(sk_i, c)$ if decryption succeeds. The vehicle stores tk_0 for subsequent service access. Otherwise, return $\perp$.

Algorithm 1: Algorithm TKGen

Input: An identity of vehicle ID_v, initial public key of vehicle pk_0, a maximum access time n, master key pair (msk, mpk), and user list $\mathcal{L}$.
Output: A ciphertext of token c_{tk}, and an updatable user list $\mathcal{L}$.

$\alpha_0 \xleftarrow{\$} \mathbb{Z}_q^*$;
$\alpha_1 \leftarrow HMAC_{msk}(\alpha_0)$;
$r_0 \leftarrow (ID_v \parallel n) \oplus H_1({pk_0}^{msk})$;
$m_0 \leftarrow H_2(pk_0, \alpha_0)$;
$h \leftarrow \mathcal{CH}$.Hash$(mpk, m_0, r_0)$;
$pk_1 \leftarrow pk_0 * g^{r_0}$;
$m_1 \leftarrow H_2(pk_1, \alpha_1)$;
$r_1 \leftarrow \mathcal{CH}$.Adapt$(msk, m_0, r_0, m_1)$;
$c_{tk} \leftarrow \mathcal{UPKE}$.Enc$(pk_0, tk_0 = \{\alpha_0, r_0, r_1\})$;
$\mathcal{L}' \leftarrow \mathcal{L} \cup \{(ID_v, n)\}$;
return c_{tk} and $\mathcal{L}'$.

Algorithm 2: Algorithm Request

Input: Key pair of vehicle (sk_i, pk_i), a token $tk_i = \{\alpha_i, r_i, r_{i+1}\}$, and master public key mpk.
Output: A request message req_i.

$k_i \leftarrow H_1(mpk^{sk_i})$;
$t_i \leftarrow Col(time)$;
$\tau_i \leftarrow H_3(pk_i, \alpha_i, t_i, r_i, r_{i+1})$;
$\sigma_i \leftarrow \mathcal{UMAC}$.Sig$(k_i, \tau_i \parallel ID_r)$;
return $req_i = \{pk_i, \alpha_i, t_i, r_i, r_{i+1}, \sigma_i\}$.

C. Service access

AuthUp.Request$(sk_i, pk_i, tk_i, mpk) \rightarrow req_i$: The detailed process of the service request algorithm is shown as Algorithm 2.

AuthUp.Feedback$(req_i, msk) \rightarrow (\sigma_r, c)$ / $\perp$: The feedback algorithm is shown as Algorithm 3. Intuitively, the algorithm first authenticates the legitimacy of OBU (Lines 1-13). If the request is from a valid OBU, then TA generates feedback for RSU (Lines 14-17). RSU will provide various services only when the feedback can be authenticated effectively. Besides, TA generates a new token for OBU to authorize his next service access (Lines 18-27).

AuthUp.Authentication$(k_r, \sigma_r) \rightarrow b$: RSU returns c to the vehicle iff $\mathcal{UMAC}$.Ver$(k_r, \tau_i \parallel ID_r, \sigma_r) = 1$. The pk_i can further be utilized for a handshake and disseminate vehicular services to the OBU.

AuthUp.KeyUpdate$(sk_i, pk_i, c) \rightarrow (sk_{i+1}, pk_{i+1}, tk_{i+1})$ or $\perp$: The key update is shown as Algorithm 4. Intuitively, OBU first decrypts the ciphertext, and the algorithm returns 0 if decryption fails. Otherwise, OBU updates tokens for the next service access.

VI. Security analysis

In this section, we formally demonstrate that AuthUp achieves the security and privacy preservation goals mentioned in Section IV-C.

Algorithm 3: Algorithm Feedback

Input: A request message $req_i = \{pk_i, \alpha_i, t_i, r_i, r_{i+1}, \sigma_i\}$ and master secret key msk.
Output: A pair of updated tag and ciphertext (σ_r, c) or $\perp$

. // Legitimacy authentication
$t_i' \leftarrow Col(time)$;
if $|t_i' - t_i| \leq \Delta t$ **then**
  $(ID_v \parallel n) \leftarrow r_i \oplus H_1({pk_i}^{msk})$;
  Check $\mathcal{L}$ for access time n;
  $k_i \leftarrow H_1({pk_i}^{msk})$;
  $\tau_i \leftarrow H_3(pk_i, \alpha_i, t_i, r_i, r_{i+1})$;
  if $\mathcal{UMAC}$.Verify$(k_i, \tau_i \parallel ID_r, \sigma_i) == 1$ **then**
    $m_i \leftarrow H_2(pk_i, \alpha_i)$;
    $\delta_i \leftarrow HMAC_{msk}(r_i)$;
    $pk_{i+1} \leftarrow pk_i g^{\delta_i}$;
    $\alpha_{i+1} \leftarrow HMAC_{msk}(\alpha_i)$;
    $m_{i+1} \leftarrow H_2(pk_{i+1}, \alpha_{i+1})$;
    if $\mathcal{CH}$.Hash$(mpk, m_i, r_i) ==$ $\mathcal{CH}$.Hash(mpk, m_{i+1}, r_{i+1}) **then**
      Update $\mathcal{L}[ID_v]$ to $n-1$;
      // Feedback to RSU
      $k_r \leftarrow HMAC_{msk}(ID_r)$;
      $\Delta \leftarrow k_r / k_i$;
      $\sigma_r \leftarrow \mathcal{UMAC}$.Update$(\Delta, \sigma_i)$;
      // Authorize for new token
      $r_{i+1}' \leftarrow (ID_v \parallel n-1) \oplus H_1({pk_{i+1}}^{msk})$;
      $\delta_{i+1}' \leftarrow HMAC_{msk}(r_{i+1}')$;
      $pk_{i+2} \leftarrow pk_{i+1} g^{\delta_{i+1}'}$;
      $\alpha_{i+1}' \xleftarrow{\$} \mathbb{Z}_q^*$;
      $m_{i+1}' \leftarrow H_2(pk_{i+1}, \alpha_{i+1}')$;
      $\alpha_{i+2} \leftarrow HMAC_{msk}(\alpha_{i+1}')$;
      $m_{i+2} \leftarrow H_2(pk_{i+2}, \alpha_{i+2})$;
      $r_{i+2} \leftarrow \mathcal{CH}$.Adapt$(msk, m_{i+1}', r_{i+1}', m_{i+2})$;
      $M \leftarrow \{\alpha_{i+1}' \parallel r_{i+2}\}$;
      $c \leftarrow \mathcal{UPKE}$.Enc$(pk_i, M \parallel \delta_{i+1}')$;
      return (σ_r, c).
    end
  end
end
return $\perp$.

Algorithm 4: Algorithm KeyUpdate

Input: Key pair of vehicle (sk_i, pk_i) and a response ciphertext c.
Output: A new key pair and an updated token tk_{i+1} or $\perp$.

// Ciphertext decryption
if *Decryption succeed* **then**
  // Update token
  $(M \parallel \delta_{i+1}') \leftarrow \mathcal{UPKE}.Dec(sk_i, c)$;
  $\{\alpha_{i+1}' \parallel r_{i+2}\} \leftarrow M$;
  $pk_{i+1} \leftarrow \mathcal{UPKE}$.Upd-Pk$(pk_i, \delta_{i+1}')$;
  $sk_{i+1} \leftarrow \mathcal{UPKE}$.Upd-Sk$(sk_i, \delta_{i+1}')$;
  $r_{i+1}' = (ID_v \parallel n-1) \oplus H_1(mpk^{sk_{i+1}})$;
  return $(sk_{i+1}, pk_{i+1}, tk_{i+1})$, where $tk_{i+1} = \{\alpha_{i+1}', r_{i+1}', r_{i+2}\}$.
else
  return $\perp$.
end

A. Security proof

Theorem 1: **EUF-CMA security of AuthUp.** Under the assumptions that the HMAC function $\mathcal{HMAC}$ and the CH scheme are collision-resistant and the UPKE scheme $\mathcal{UPKE}$ is IND-CPA secure, the proposed AuthUp provides EUF-CMA security with the advantage: $Adv_{\mathcal{A},AuthUp}^{EUF-CMA}(1^\lambda) = \epsilon_{\mathcal{UPKE}} + \epsilon_{\mathcal{HMAC}} * \epsilon_{\mathcal{CH}}$, where $\epsilon_{\mathcal{HMAC}}$, $\epsilon_{\mathcal{UPKE}}$, and $\epsilon_{\mathcal{CH}}$ are the advantages of breaking the collision resistance of $\mathcal{HMAC}$, the IND-CPA security of $\mathcal{UPKE}$ and the collision resistance of $\mathcal{CH}$.

Proof: If any PPT adversary $\mathcal{A}$ can break AuthUp with a non-negligible advantage, we can define a simulator $\mathcal{S}$ to break the collision resistance of $\mathcal{HMAC}$, the IND-CPA security of $\mathcal{UPKE}$, or the collision resistance of $\mathcal{CH}$. The process of the game is as follows:

1) Setup: $\mathcal{S}$ first computes system public parameters involving public keys of a UMAC $pp_{\mathcal{UMAC}} \leftarrow \mathcal{UMAC}$.Setup$(1^\lambda)$, a UPKE $pp_{\mathcal{UPKE}} \leftarrow \mathcal{UPKE}$.Setup$(1^\lambda)$, and a CH $pp_{\mathcal{CH}} \leftarrow \mathcal{CH}$.Setup$(1^\lambda)$ generates a key pair $(msk, mpk) \leftarrow \mathcal{CH}$.KeyGen$(pp_{\mathcal{CH}})$. $\mathcal{S}$ returns public parameters $pp = (pp_{\mathcal{UMAC}}, pp_{\mathcal{UPKE}}, pp_{\mathcal{CH}})$ and public key mpk to $\mathcal{A}$. $\mathcal{S}$ initializes six empty sets $\mathcal{L}_{H_1}, \mathcal{L}_{H_2}, \mathcal{L}_{HMAC}, \mathcal{L}_{user}, \mathcal{L}_{corr}, \mathcal{L}_{tk} \leftarrow \emptyset$.

2) Query: $\mathcal{A}$ performs the legal operations by querying the following oracles:

Oracle $\mathcal{O}_{\text{H}_1}(i)$: $\mathcal{A}$ can query the H_1 oracle with a group element i at most q_{H_1} times. $\mathcal{S}$ randomly selects a number $\tau_{i1} \leftarrow \mathbb{Z}_q^*$, updates the list $\mathcal{L}_{H_1} \leftarrow \mathcal{L}_{H_1} \cup \{(i, \tau_{i1})\}$, and returns τ_{i1} to $\mathcal{A}$.

Oracle $\mathcal{O}_{\text{H}_2}(pk_i, \alpha_i)$: $\mathcal{A}$ can query the H_2 oracle with a public key pk_i and a randomness α_i at most q_{H_2} times. $\mathcal{S}$ randomly selects a number $\tau_{i2} \leftarrow \mathbb{Z}_q^*$, updates the list $\mathcal{L}_{H_2} \leftarrow \mathcal{L}_{H_2} \cup \{(pk_i, \alpha_i, \tau_{i2})\}$, and returns τ_{i2} to $\mathcal{A}$.

Oracle $\mathcal{O}_{\text{HMAC}}(msk, i)$: $\mathcal{A}$ can query the $HMAC$ oracle with a randomness $i \in \mathbb{Z}_q^*$ at most q_{HMAC} times. $\mathcal{S}$ randomly selects a random number $j \leftarrow \mathbb{Z}_q^*$, updates the list $\mathcal{L}_{HMAC} \leftarrow \mathcal{L}_{HMAC} \cup \{(i, j)\}$, and returns j to $\mathcal{A}$.

Oracle $\mathcal{O}_{\text{Setup}_\text{v}}(ID_i)$: $\mathcal{A}$ can query the oracle with an identity ID_i at most $q_{\text{setup}_\text{v}}$ times. $\mathcal{S}$ generates a key pair $(sk_i, pk_i) \leftarrow \mathcal{UPKE}$.KeyGen$(pp)$, updates the list $\mathcal{L}_{user} \leftarrow \mathcal{L}_{user} \cup \{(ID_i, sk_i, pk_i)\}$ and returns pk_i to $\mathcal{A}$.

Oracle $\mathcal{O}_{\text{Corrupt}}(ID_i, pk_i)$: $\mathcal{A}$ can query the user corrupt oracle with an identity ID_i at most q_{corrupt} times. $\mathcal{S}$ updates the list $\mathcal{L}_{corr} \leftarrow \mathcal{L}_{corr} \cup \{(ID_i, pk_i)\}$ and returns sk_i to $\mathcal{A}$.

Oracle $\mathcal{O}_{\text{TKGen}}(ID_i, pk_i)$: $\mathcal{A}$ can query the token generation oracle with an identity ID_i and a public key pk_i at most q_{TKGen} times. $\mathcal{A}$ selects random numbers $\alpha_0, \alpha_1, \tau_{i1}, m_0, m_1 \in \mathbb{Z}_q^*$, computes $r_0 = ID_i \oplus \tau_{i1}$, $h \leftarrow \mathcal{CH}$.Hash$(mpk, m_0, r_0)$ $pk_1 \leftarrow pk_0 * g^{r_0}$, $r_1 \leftarrow \mathcal{CH}$.Adapt$(msk, m_0, r_0, m_1)$, $c_{tk} \leftarrow \mathcal{UPKE}$.Enc$(pk_i, tk_0 = \{\alpha_0, r_0, r_1\})$. Then, $\mathcal{A}$ in-

serts $(\alpha_0, \alpha_1), (pk_0{}^{msk}, \tau_{i1}), (pk_0, \alpha_0, m_0), (pk_1, \alpha_1, m_1)$ into $\mathcal{L}_{HMAC}, \mathcal{L}_{H_1}, \mathcal{L}_{H_2}$, and $\mathcal{L}_{H_2}$ respectively and returns c_{tk} to $\mathcal{A}$.

3) Output: After limited queries, $\mathcal{A}$ outputs $(pk_i^*, \alpha_i^*, r_i^*, r_{i+1}^*)$ as a tuple of results. $\mathcal{S}$ returns 1 only when the following conditions are satisfied simultaneously:

a) $\mathcal{S}$ extracts (pk_i, α_i, m_i) from $\mathcal{L}_{H_2}$, $(r_i, \delta_i), (\alpha_i, \alpha_{i+1})$ from $\mathcal{L}_{HMAC}$, computes $pk_{i+1} = pk_i g^{\delta_i}$, extracts $(pk_{i+1}, \alpha_{i+1}, m_{i+1})$ from $\mathcal{L}_{H_2}$, and the equation $\mathcal{CH}.\text{Hash}(mpk, m_i, r_i) = \mathcal{CH}.\text{Hash}(mpk, m_{i+1}, r_{i+1})$ holds.

b) $pk_i^* \notin \mathcal{L}_{corr}$.

Two cases may occur as follows:

a) $\mathcal{A}$ first queries **Oracle** $\mathcal{O}_{\text{TKGen}}$ with (ID_i^*, pk_i^*) and obtains c_{tk}. Then, $\mathcal{A}$ decrypts the ciphertext and obtains $tk_i = \{\alpha_i^*, r_i^*, r_{i+1}^*\}$ without sk_i^*. In this case, $\mathcal{S}$ can output $tk_i = \{\alpha_i^*, r_i^*, r_{i+1}^*\}$ to $Exp_{\mathcal{A},\mathcal{UPKE}}^{IND-CPA}$ to break the IND-CPA security of $\mathcal{UPKE}$.

b) There exist i and j that satisfy $\mathcal{CH}.\text{Hash}(mpk, m_i^*, r_i^*) = \mathcal{CH}.\text{Hash}(mpk, m_{i+1}^*, r_{i+1}^*) = \mathcal{CH}.\text{Hash}(mpk, m_j^*, r_j^*)$. In this case, $\mathcal{S}$ can output m_{i+1}^* and m_j^* to break the collision resistance of $\mathcal{HMAC}$ since the calculation of both m_{i+1}^* and m_j^* is deterministic according to the definition of HMAC. Meanwhile, $\mathcal{S}$ can also output (m_{i+1}^*, r_{i+1}^*) and (m_j^*, r_j^*) to break the collision resistance of $\mathcal{CH}$.

Summarizing the above analysis, we can conclude that $\mathcal{S}$ can break the IND-CPA security of UPKE or the collision resistance of HMAC and CH with a non-negligible advantage. The advantage $Adv_{\mathcal{A},AuthUp}^{EUF-CMA}(1^\lambda) = \epsilon_{\mathcal{UPKE}} + \epsilon_{\mathcal{HMAC}} * \epsilon_{\mathcal{CH}}$, where $\epsilon_{\mathcal{HMAC}}$, $\epsilon_{\mathcal{UPKE}}$, and $\epsilon_{\mathcal{CH}}$ are the advantages of breaking the collision resistance of $\mathcal{HMAC}$, the IND-CPA security of $\mathcal{UPKE}$ and the collision resistance of $\mathcal{CH}$. □

B. Security requirement discussion

1) Message integrity and authenticity: Theorem 1 indicates that AuthUp satisfies EUF-CMA security. An obvious corollary of the theorem is that AuthUp satisfies both message integrity and authenticity. In other words, no adversary can forge a legitimate request that can pass the authentication of TA.

2) Conditional privacy-preserving:
   - Anonymity: Identities of vehicles are hidden in the token r_i to protect private information, where $r_i = (ID_v \parallel n) \oplus H_2(pk_i{}^{msk})$. No entities accept TA and the vehicle itself can compute $(ID_v \parallel n) = r_i \oplus H_2(pk_i{}^{msk}) = r_i \oplus H_2(mpk^{sk_i})$ due to the discrete logarithm (DL) assumption.
   - Unlinkability: According to AuthUp, a vehicle submits a request typed $\{pk_i, \alpha_i, t_i, r_i, r_{i+1}, \sigma_i\}$ to access vehicular services. Among the requests, $\{pk_i, \alpha_i, r_i, r_{i+1}\}$ are one-time tokens, and t_i is a real-time timestamp; these lead to the randomness of σ_i. Besides, TA may return a ciphertext c to update the token of the vehicle. However, adversaries cannot obtain anything from the ciphertext since the UPKE scheme can protect the confidentiality of the message. Thus, no adversaries can distinguish whether two requests or feedback are from the same vehicle.
   - Traceability: Similarly, since TA computes $r_i = (ID_v \parallel n) \oplus H_2(pk_i{}^{msk})$ as an access token, only TA can reveal the vehicle's identity as $(ID_v \parallel n) = r_i \oplus H_2(pk_i{}^{msk})$ according to the DL assumption.

3) Resist different attacks: The authentication scheme should have the ability to effectively resist various attacks, i.e., Sybil attacks, side-channel attacks, modification, and replay attacks.
   - Resistance to Sybil attacks: AuthUp utilizes limited authentication to resist Sybil attacks. Every time a vehicle accesses vehicular services, the TA will authorize a new token, including $r_i = (ID_v \parallel n) \oplus H_2(pk_i{}^{msk})$ for the next service access. Additionally, the TA maintains a user list that is continuously updated to authenticate the legitimacy of tokens. Only the most recently updated token can successfully pass authentication, and each token expires once utilized. Consequently, AuthUp allows each legitimate vehicle to hold only one valid token at a given time, effectively thwarting Sybil attacks.
   - Resistance to side-channel attacks: In AuthUp, vehicles rotate key pairs and tokens stored in TPD each time they successfully access services. This means that secret key pair and access token will be used only once, and no long-term security parameters exist in OBUs. In other words, adversaries can only collect and analyze one-time parameters, achieving effectiveness similar to that of single-shot attacks, which are constrained by various factors. Therefore, AuthUp effectively prevents the theft of sensitive information through side-channel attacks.
   - Resistance to modification attacks: The integrity of a request from a vehicle is guaranteed by the HMAC scheme. An ephemeral secret key between the vehicle and TA is established as the trapdoor for HMAC. Due to the DL assumption, no one can forge such a request, and any modification will be detected. Consequently, AuthUp is capable of defending against modification attacks.
   - Resistance to replay attacks: To mitigate such attacks, every request includes a real-time timestamp protected by HMAC. Whenever TA receives a request, it will examine the freshness of the timestamp. Requests containing a stale timestamp will be classified as a replay attack and subsequently discarded. Therefore, AuthUp effectively neutralizes the threat posed by replay attacks.

It is worth noting that AuthUp addresses the key exposure problem, which is an important security property in CH. Specifically, the original chameleon signature scheme [34] utilizes Chaum-Pedersen trapdoor commitments. Given two pairs (m, r) and (m', r') such that $\mathcal{CH}.\text{Hash}(pk, m, r) = \mathcal{CH}.\text{Hash}(pk, m', r') = h$, the se-

TABLE II: Security comparison among the proposed scheme and related works

Requirements	[12]	[13]	[14]	[18]	[19]	[20]	[21]	AuthUp
Message integrity and authenticity	✓	✓	✓	✓	×	✓	✓	✓
Anonymity	✓	✓	✓	✓	×	✓	✓	✓
Unlinkability	×	✓	×	×	×	×	×	✓
Traceability	✓	✓	✓	✓	×	✓	×	✓
Sybil attack resistance	✓	✓	✓	✓	✓	✓	✓	✓
Side-channel attack resistance	×	×	×	×	✓	×	×	✓
Modification attack resistance	✓	✓	✓	✓	×	✓	✓	✓
Replay attack resistance	✓	×	✓	×	×	✓	✓	✓

✓: The requirement is satisfied ×: The requirement is not satisfied

cret key x can be reconstructed using the formula $x = (H(m')-H(m))/(r-r')$. In AuthUp, the legitimacy of access tokens is verified by the equation $\mathcal{CH}.\text{Hash}(mpk, m_i, r_i) = \mathcal{CH}.\text{Hash}(mpk, m_{i+1}, r_{i+1})$, where m_{i+1} is only known to the TA. Since $\alpha_{i+1} = HMAC_{msk}(\alpha_i)$ and $m_{i+1} = H_2(pk_{i+1}, \alpha_{i+1})$, it is infeasible to compute m_{i+1} without the master secret key msk. Thus, the security of AuthUp is ensured by the collision resistance and one-wayness properties of the hash functions.

A security comparison of various Sybil attack-resistant schemes for VANET is presented in Table II. It is evident that only [19] and AuthUp can withstand side-channel attacks. This is because OBUs in [19] use only real-time coordinates to generate trajectories, which helps resist Sybil attacks. In contrast, other schemes rely on long-term keys to generate signatures or certificates for vehicle authentication. The key rotation and identity authentication mechanisms enable AuthUp to satisfy all security and privacy-preserving requirements.

VII. Performance evaluation

After a comprehensive consideration of security and efficiency, the cornerstones of AuthUp, which involve UMAC [30], UPKE [32], and CH [34], are leveraged to instantiate our proposed AuthUp. We refer the reader to Appendices A, B, and C for a detailed description of the protocols. The experimental results of our implementation were obtained on a virtual machine with Intel(R) Core(TM) i7-12700 CPU @2.10GHz, 32 GB RAM, and an operating system running 64-bit Ubuntu 22.04.4 LTS. The Python cryptography library Charm-crypto [38] is adopted to simulate cryptographic operations. The parameters were chosen to ensure that the schemes can achieve the target 80-bit level of security. In particular, we leverage a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are groups of order q generated by a point from the curve $SS512$. For ECC-based scheme, we use an additive group $\mathbb{G}$ generated by a point P with the order q from curve $secp160r1$. Two state-of-the-art solutions (i.e., a cryptography-based authentication approach [13] and a trajectory-based approach [21]) are also analyzed and compared with AuthUp to better evaluate the proposed scheme.

A. Computational cost

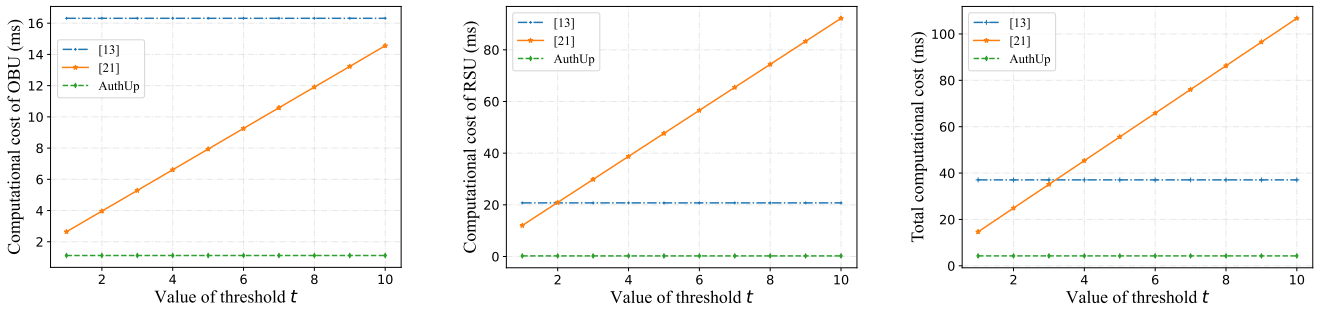
According to the setting, some definitions and notations are presented in Table III. The benchmarks for common cryptographic operations in AuthUp and related works are presented in Table IV, which involve 10^3 trials in our experiment. A comparative analysis of the two existing approaches [13], [21] and the proposed scheme is conducted. The BLS signature scheme [39] is adopted to instantiate the solution of Baza et al. [21]. The computational cost of OBU, RSU, and TA in both schemes is given in Table V. The performance of Baza et al.'s scheme [21] varies with the threshold t. Specifically, it requires $(t+1)T_{sm-bp} + tT_h$ for a vehicle to generate a request and $(4t+2)T_{bp}+(2t)T_{sm-bp}+(t-1)T_{pa-bp}+(2t+1)T_h+(2t)T_{mtp}$ for t RSUs to generate and verify a signature to avoid Sybil attacks. Note that the communication cost for PoW is not counted due to the absence of a setting. In Lin's scheme [13], it requires $2T_{bp} + 10T_{sm-bp} + 5T_{pa-bp} + 2T_h$ for a vehicle to generate an event, and $2T_{bp} + 11T_{sm-bp} + 7T_{pa-bp} + 2T_h$ for an RSU to verify an event report. Besides, $2T_{bp} + T_{pa-bp}$ is needed to compute an exclusive value for Sybil message detection. In the proposed AuthUp, it requires $5T_{sm-ecc} + T_{pa-ecc} + 4T_h + T_{mtp}$ for each OBU to generate a request and update its token, $T_{sm-ecc} + T_{mtp}$ for an RSU to verify a request, and $13T_{sm-ecc} + 4T_{pa-ecc} + 13T_h + T_{mtp} + 5T_{hmac}$ for TA to verify the request and generate a fresh token, respectively.

Fig. 4 vividly shows the computational costs of different schemes. From Fig. 4a and Fig. 4b, the computational costs of OBU and RSU in AuthUp are much less than those in Baza et al.'s scheme [21] and Lin's scheme [13]. In particular, the proposed AuthUp is more efficient than Baza et al.'s scheme [21] even when the threshold $t = 1$. Fig. 4c demonstrates that though TA is involved in the verification process of AuthUp, the total computational cost of AuthUp is still superior to the other two schemes. Compared to LSR [13] and Baza et al.'s scheme [21] when $t = 1$, AuthUp significantly reduces the computational cost, achieving an improvement of approximately 88.5% and 70.8%, respectively, demonstrating its superior efficiency. Considering that OBU is resource-limited, AuthUp is more efficient and applicable to VANET.

B. Communication overhead

According to the 80-bit security level setting, we assume that elements in $\mathbb{G}_1$, $\mathbb{G}$, and $\mathbb{Z}_q^*$ occupy 1024 bits, 320 bits, and 160 bits, respectively, and are denoted by $|\mathbb{G}|$ and $|\mathbb{Z}_q^*|$. The inputs of general hash functions are mapped to $\mathbb{Z}_q^*$. Besides, we utilize $|T|$ to denote the size of a timestamp and occupy 32 bits.

In Lin's scheme [13], an event report is represented as $ER_j = event_0 \parallel T_u \parallel T_v \parallel c \parallel s_1 \parallel s_2 \parallel s_3$, where $event_0 = \{t_0 \parallel l_0 \parallel e_0\}$, $\{T_u, T_v\} \in \mathbb{G}_1$,



(a) (b) (c)

Fig. 4: Computational cost comparison.

TABLE III: Partial notations used in the proposed scheme.

Notations	Description
T_{bp}	Execution time for one bilinear pairing operation
T_{sm-bp}	Execution time for one scale multiplication operation in $\mathbb{G}_1$
T_{pa-bp}	Execution time for one point addition operation in $\mathbb{G}_1$
T_{sm-ecc}	Execution time for one scale multiplication operation in $\mathbb{G}$
T_{pa-ecc}	Execution time for one point addition operation in $\mathbb{G}$
T_h	Execution time for one general hash operation
T_{mpt}	Execution time for one map-to-point hash operation
T_{hmac}	Execution time for one hmac operation

TABLE IV: Execution time of different cryptographic operations.

Notations	Execution time (ms)	Notations	Execution time (ms)
T_{bp}	1.5540	T_{sm-bp}	1.3159
T_{pa-bp}	0.0059	T_{sm-ecc}	0.2136
T_{pa-ecc}	0.0007	T_h	0.0008
T_{mpt}	0.0191	T_{hmac}	0.0038

$\{l_0, e_0, c, s_1, s_2, s_3\} \in \mathbb{Z}_q^*$, and t_0 is a timestamp. The RSU will transmit the reports to TA for verification and return the updated revocation list. Therefore, the communication cost of LSR is at least $4|\mathbb{G}_1| + 12|\mathbb{Z}_q^*| + 2|T|$ =6080 bits to realize Sybil attack detection with traceability. In Baza et al.'s scheme [21], the communication cost depends on the maximum number of RSUs a vehicle traverses within a given period. We assume the number as the threshold t to quantify its performance. A vehicle then sends t requests, each requiring $(t+4)|\mathbb{G}_1|+(2t+1)|\mathbb{Z}_q^*|+(2t)|T|$, while the feedback occupies $(t+1)|\mathbb{G}_1|+t|\mathbb{Z}_q^*|+t|T|$. Subsequently, a message including a trajectory with $2|\mathbb{G}_1|+t|\mathbb{Z}_q^*|+t|T|$ will be reported. Thus, the total communication overhead is $(2t+7)|\mathbb{G}_1|+(4t+1)|\mathbb{Z}_q^*|+(4t)|T|$. When $t = 1$, its communication overhead requires $9|\mathbb{G}_1| + 5|\mathbb{Z}_q^*| + 4|T|$ =10144 bits. In AuthUp, a vehicle generates a request represented as $\{pk_i, \alpha_i, t_i, r_i, r_{i+1}, \sigma_i\}$, where $\{pk_i, \sigma_i\} \in \mathbb{G}$, $\{\alpha_i, r_i, r_{i+1}\} \in \mathbb{Z}_q^*$, and t_i is a timestamp. The RSU relays the request to TA to verify the legitimacy of the request. Then, TA generates feedback represented as (σ_r, c), where $\sigma_r \in \mathbb{G}$ and c is the ciphertext of three elements in $\mathbb{Z}_q^*$. After verifying the feedback, RSU returns the ciphertext c for OBU to update its key pair and token. Therefore, the total communication cost of AuthUp is $7|\mathbb{G}|+12|\mathbb{Z}_q^*|+2|T|$ =4224 bits.

In summary, Baza et al.'s scheme [21] requires a higher communication overhead than Lin's scheme [13] and AuthUp. Specifically, when comparing AuthUp to Lin et al.'s scheme and Baza et al.'s scheme at $t = 1$, AuthUp achieves a significant reduction in communication overhead of approximately 30.5% and 58.4%, respectively. Moreover, detecting Sybil nodes by the TA is more significant than local Sybil node detection. This is due to the TA's authority to revoke access permissions for malicious nodes, thereby preventing real-time attacks. Thus, the proposed AuthUp facilitates dynamic and real-time V2I communication with less communication overhead, which is much practical to provide efficient and robust vehicular services.

C. Network simulation

To further assess the performance of AuthUp in real traffic environments, we utilize the widely used network simulator OMNeT++ 5.6.1[1], integrated with the bidirectional coupling tool VEINS 5.2[2] and traffic simulation tool SUMO 1.2.0[3]. The simulation parameters are detailed in Table VI. We evaluate the average message delay and the average message loss rate in V2I communications, considering various maximum speeds and densities, as OBUs upload requests to RSUs and download feedback from them. Specifically, we control the density of OBUs by adjusting different departure periods.

The simulation results for the two metrics are illustrated in Fig. 5, where the departure period is set as 10 s in Figs. 5a and 5b, and the speed is set to 15 m/s in Figs. 5c and 5d. As illustrated in Fig. 5, the average message delay increases as the speed of OBUs rises. The speed of OBUs also has a more pronounced effect on the average packet loss rate during the V2I upload phase. This is likely due to the increased number of requests received by the RSU, which has led to channel congestion. Meanwhile, both speed and density of OBU have little effect on average communication delay, which remains stable at 0.490 ms and 0.178 ms during the V2I upload and download phases, respectively. It is evident that the packet

[1]https://omnetpp.org/
[2]https://veins.car2x.org/
[3]https://eclipse.dev/sumo/

TABLE V: Comparative comparison of computational cost.

Entity		OBU (ms)	RSU (ms)	TA (ms)
Schemes	[13]	$2T_{bp} + 10T_{sm-bp} + 5T_{pa-bp} + 2T_h \approx 16.313$	$4T_{bp} + 11T_{sm-bp} + 8T_{pa-bp} + 2T_h \approx 20.754$	-
	[21]	$(t+1)T_{sm-bp} + tT_h \approx 1.3167 * t + 1.3159$	$(4t+2)T_{bp} + (2t)T_{sm-bp} + (t-1)T_{pa-bp} + (2t+1)T_h + (2t)T_{mtp} \approx 8.9079t + 3.1101$	-
	AuthUp	$5T_{sm-ecc} + T_{pa-ecc} + 4T_h + T_{mtp} \approx 1.120$	$T_{sm-ecc} + T_{mtp} \approx 0.233$	$13T_{sm-ecc} + 4T_{pa-ecc} + 13T_h + T_{mtp} + 5T_{hmac} \approx 2.922$

t represents the threshold for RSUs to generate a trajectory. $-$ represents TA does not participate in calculations.

TABLE VI: Simulation settings

Parameters	Value
Simulation area	2500×2500 m^2
Max interface distance	500 m
Data transmission rate	6 Mbps
Transmission power	20 mW
Sensitivity	-98 dBm
Thermal noise	-110 dBm
Simulation time	2000 s
Network protocol	IEEE 802.11p

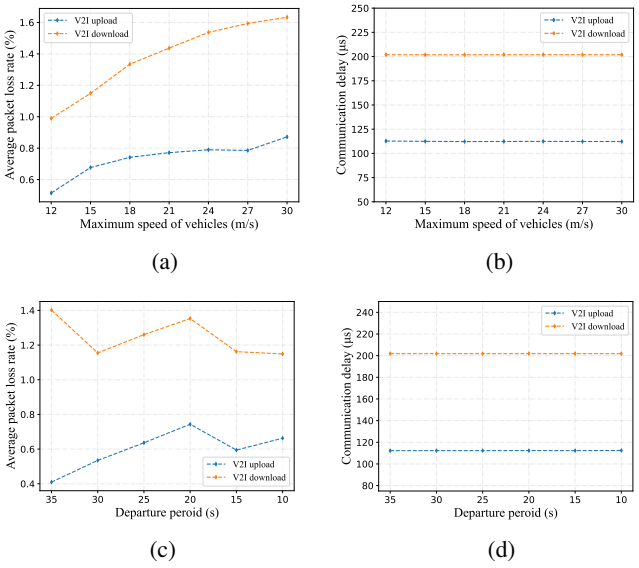


Fig. 5: Simulation results in terms of speeds and densities.

loss rate and communication delay in the upload phase are significantly higher than those during the download phase. This is because OBUs upload messages that are larger in size compared to those in the download phase. In other words, the size of the beacons significantly affects the performance of vehicular communications. Since V2I communications in AuthUp account for nearly half of the total communication overhead, the scheme is practical for deployment in VANETs.

D. TA simulation

A critical factor influencing the practicality and scalability of AuthUp is the performance of the TA. We evaluate its performance on a server equipped with a 64-bit Ubuntu 22.04.4 LTS OS, powered by an Intel Xeon Gold 6326 CPU @ 2.90GHz, supported by 251 GB of RAM and 3 NVIDIA GeForce RTX 3090 GPUs. Specifically, we evaluate the storage requirements and parallel computing capabilities.

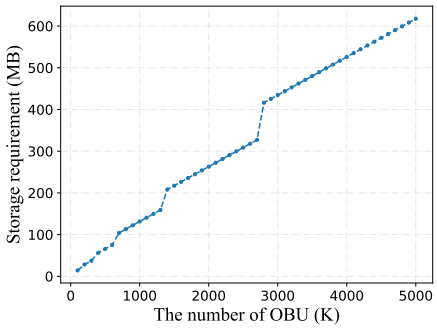


Fig. 6: The storage requirement of TA.

In AuthUp, the server maintains a list typed (ID_v, n), where ID_v is a random 160-bit number and n is a 32-bit number. The storage requirement of TA as the number of registered vehicles increases is illustrated in Fig 6. It is evident that the storage requirements remain quite low. Specifically, TA requires 617.61 MB to store identity messages of 5 million OBUs, which is relatively lightweight for any server.

We also evaluate the throughput of the server when executing the Feedback Algorithm in AuthUp. The server is subjected to continuous and concurrent processing of a large number of requests to assess its performance under maximum load. The CPU usage over time is depicted in Fig. 7. Specifically, the Feedback Algorithm one million times within 233 s. During this period, the CPU usage typically remained around 76%. From the figure, we conclude that the server can achieve more than 5000 transactions per second (TPS) when operating at full capacity. In addition to CPU performance, we also evaluate the input and output bandwidth. Compared to computational demands, AuthUp has significantly lower requirements for the server's I/O capabilities. Specifically, the server requires approximately 1.87 MB of input bandwidth and 1.72 MB of output bandwidth to sustain 5000 TPS. The feasibility of achieving 5000 TPS with the server is well supported by these relatively modest bandwidth requirements.

Generally, compared to storage requirements and I/O capabilities, the primary bottleneck of AuthUp is its computational capability. In extreme cases, a single TA might be vulnerable to distributed denial-of-service (DDoS) attacks. However, with advancements in cloud computing technology [40], the TA can possess significantly more computing power, allowing

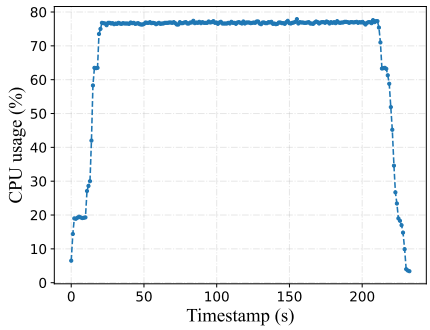


Fig. 7: CPU usage over time.

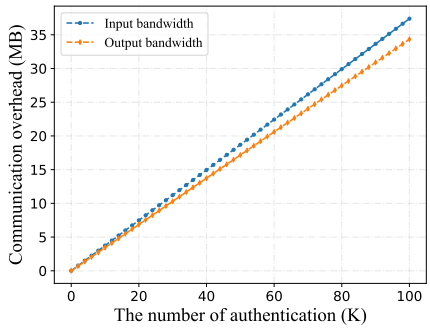


Fig. 8: Simulation result of the server's input and output bandwidth.

it to handle the desired transaction rate efficiently without substantial strain on its resources. Alternatively, other distributed technologies can be employed to offload the single TA [40]. Nonetheless, employing such technologies introduces the challenge of cross-domain authentication [41], which we identify as future work.

VIII. Conclusion

In this paper, a privacy-preserving authentication scheme for V2I communications entitled AuthUp is proposed for VANET to thwart Sybil attacks. TA generates an access token for vehicles either during registration or when accessing vehicular services with the technologies of UMAC and CH. These tokens are then distributed to vehicles using a UPKE scheme to provide resistance against passive and active attacks. Notably, the one-time token can pass authentication by TA only once, and any abuse of tokens will be promptly rejected as a Sybil attack. A comprehensive security proof under the ROM is conducted to demonstrate the EUF-CMA security of AuthUp. Extensive experimental results also show the efficiency and practicability of AuthUp.

Appendix A

UMAC proposed by Cini et al. [30] is used to instantiate AuthUp. The construction is as follows:

1) $\mathcal{UMAC}.\text{Setup}(1^\lambda, n) \rightarrow (pp, k)$: Run $(\mathbb{G}, q, g) \leftarrow GGen(1^\lambda)$, select a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$. Select a random number $k_0 \in \mathbb{Z}_q^*$ and return $pp = \{\mathbb{G}, q, g, H\}$ and k_0.
2) $\mathcal{UMAC}.\text{Next}(k_e) \rightarrow (k_{e+1}, \delta_{e+1})$: Select $\delta_{e+1} \in \mathbb{Z}_q^*$ and return $k_{e+1} = k_e * \delta_{e+1}$.
3) $\mathcal{UMAC}.\text{Sig}(k_e, M) \rightarrow \sigma_e$: Compute $\sigma_e = H(M)^{k_e}$ and return (M, σ_e).
4) $\mathcal{UMAC}.\text{Update}(\delta_{e+1}, \sigma_e) \rightarrow \sigma_{e+1}$: Compute $\sigma_{e+1} = \sigma_e^{\delta_{e+1}}$ and return σ_{e+1}.
5) $\mathcal{UMAC}.\text{Ver}(k_e, M, \sigma_e) \rightarrow b$: Return 1 if $H(M)^{k_e} = \sigma_e$, otherwise return 0.

Appendix B

UPKE proposed by Alwen et al. [32] is used to instantiate AuthUp. The construction is as follows:

1) $\mathcal{UPKE}.\text{Setup}(1^\lambda) \rightarrow pp$: Let $(\mathbb{G}, q, g)$ be an elliptic curve group with prime order q, and H be a collision-resistant hash function $\{0,1\}^* \rightarrow \{0,1\}^l$. The public parameters $pp = \{\mathbb{G}, q, g, H\}$.
2) $\mathcal{UPKE}.\text{KeyGen}(pp) \rightarrow (sk_0, pk_0)$: Select a random number $sk_0 \in Z_q^*$, and compute $pk_0 = g^{sk_0}$. Return the key pair (sk_0, pk_0).
3) $\mathcal{UPKE}.\text{Enc}(pk_e, m) \rightarrow c$: $(r, \delta) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$, compute $c \leftarrow (g^r, H(pk_e^r) \oplus (m \parallel \delta))$, and return c.
4) $\mathcal{UPKE}.\text{Dec}(sk_e, c) \rightarrow m \ / \ \perp$: Compute $m \parallel \delta \leftarrow H(c_1^{sk_e}) \oplus c_2$, and return m.
5) $\mathcal{UPKE}.\text{Upd-Pk}(pk_e, \delta) \rightarrow pk_{e+1}$: Compute $pk_{e+1} = pk_e g^\delta$ and return pk_{e+1}.
6) $\mathcal{UPKE}.\text{Upd-Sk}(sk_e, \delta) \rightarrow sk_{e+1}$: Compute $sk_{e+1} = sk_e + \delta$ and return sk_{e+1}.

Appendix C

CH proposed by Krawczyk and Rabin [34] is used to instantiate AuthUp. The construction is as follows:

1) $\mathcal{CH}.\text{Setup}(1^\lambda) \rightarrow pp$: Let $(\mathbb{G}, q, g) \leftarrow GGen(\lambda)$ be an elliptic curve group with prime order q, and H be a collision-resistant hash function $\{0,1\}^* \rightarrow \mathbb{Z}_q$. Return $pp = (\mathbb{G}, q, g, H)$.
2) $\mathcal{CH}.\text{KeyGen}(pp) \rightarrow (sk, pk)$: Choose a random number $x \in \mathbb{Z}_q^*$ and return $(sk, pk) = (x, g^x)$.
3) $\mathcal{CH}.\text{Hash}(pk, m, r) \rightarrow h$: Return $h = g^{H(m)} pk^r$.
4) $\mathcal{CH}.\text{Verify}(pk, m, h, r) \rightarrow b$: Set $h' = g^{H(m)} pk^r$. If $h' = h$, return 1 and return 0, otherwise.
5) $\mathcal{CH}.\text{Adapt}(sk, m, r, m') \rightarrow r'$: Return $r' = (H(m) + xr - H(m'))/x$.

References

[1] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of Network and Computer Applications*, vol. 37, pp. 380–392, 2014.
[2] R. Hussain, J. Lee, and S. Zeadally, "Trust in VANET: A survey of current solutions and future research opportunities," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2553–2571, 2021.
[3] X. Lin, R. Lu, C. Zhang, H. Zhu, P.-h. Ho, and X. Shen, "Security in vehicular ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 88–95, 2008.
[4] T. Limbasiya and D. Das, "IoVCom: Reliable comprehensive communication system for internet of vehicles," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2752–2766, 2021.

[5] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[6] L. Wei, J. Cui, H. Zhong, Y. Xu, and L. Liu, "Proven secure tree-based authenticated key agreement for securing V2V and V2I communications in VANETs," *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3280–3297, 2022.

[7] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 251–260.

[8] X. Dong, Y. Zhang, Y. Guo, Y. Gong, Y. Shen, and J. Ma, "PRAM: A practical sybil-proof auction mechanism for dynamic spectrum access with untruthful attackers," *IEEE Transactions on Mobile Computing*, vol. 22, no. 2, pp. 1143–1156, 2023.

[9] M. Luo, A. C. Myers, and G. E. Suh, "Stealthy tracking of autonomous vehicles with cache side channels," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020, pp. 859–876.

[10] C. Liu, A. Chakraborty, N. Chawla, and N. Roggel, "Frequency throttling side-channel attack," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1977–1991.

[11] P. Horváth, D. Lauret, Z. Liu, and L. Batina, "Sok: Neural network extraction through physical side channels," in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/horvath

[12] Q. Wu, J. Domingo-Ferrer, and U. Gonzalez-Nicolas, "Balanced trustworthiness, safety, and privacy in vehicle-to-vehicle communications," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, pp. 559–573, 2010.

[13] X. Lin, "LSR: Mitigating zero-day sybil vulnerability in privacy-preserving vehicular peer-to-peer networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 237–246, 2013.

[14] T. Zhou, R. R. Choudhury, P. Ning, and K. Chakrabarty, "P2DAP — Sybil attacks detection in vehicular ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 582–594, 2011.

[15] Y. Chen, J. Yang, W. Trappe, and R. P. Martin, "Detecting and localizing identity-based attacks in wireless and sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2418–2434, 2010.

[16] Y. Yao, B. Xiao, G. Wu, X. Liu, Z. Yu, K. Zhang, and X. Zhou, "Multi-channel based sybil attack detection in vehicular ad hoc networks using rssi," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 362–375, 2019.

[17] Y. Yao, B. Xiao, G. Yang, Y. Hu, L. Wang, and X. Zhou, "Power control identification: A novel sybil attack detection scheme in VANETs using RSSI," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2588–2602, 2019.

[18] S. Benadla, O. R. Merad-Boudia, S. M. Senouci, and M. Lehsaini, "Detecting sybil attacks in vehicular fog networks using RSSI and blockchain," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 3919–3935, 2022.

[19] V.-L. Nguyen, P.-C. Lin, and R.-H. Hwang, "Enhancing misbehavior detection in 5G vehicle-to-vehicle communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9417–9430, 2020.

[20] S. Chang, Y. Qi, H. Zhu, J. Zhao, and X. Shen, "Footprint: Detecting sybil attacks in urban vehicular networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 1103–1114, 2012.

[21] M. Baza, M. Nabil, M. M. E. A. Mahmoud, N. Bewermeier, K. Fidan, W. Alasmary, and M. Abdallah, "Detecting sybil attacks using proofs of work and location in VANETs," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 39–53, 2022.

[22] A. Purnal, M. Bognar, F. Piessens, and I. Verbauwhede, "Showtime: Amplifying arbitrary CPU timing side channels," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security, ASIA CCS 2023, Melbourne, VIC, Australia, July 10-14, 2023*, J. K. Liu, Y. Xiang, S. Nepal, and G. Tsudik, Eds. ACM, 2023, pp. 205–217. [Online]. Available: https://doi.org/10.1145/3579856.3590332

[23] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power Side-Channel attacks into remote timing attacks on x86," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 679–697. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/wang-yingchen

[24] D. Chatterjee, K. Pratihar, A. Hazra, U. Rührmair, and D. Mukhopadhyay, "Systematically quantifying cryptanalytic nonlinearities in strong pufs," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 1126–1141, 2024.

[25] Q. Xie, Z. Ding, W. Tang, D. He, and X. Tan, "Provable secure and lightweight blockchain-based v2i handover authentication and v2v broadcast protocol for vanets," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 15 200–15 212, 2023.

[26] Q. Xie, Z. Ding, and P. Zheng, "Provably secure and anonymous v2i and v2v authentication protocol for vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7318–7327, 2023.

[27] Y. Wang, W. Zhang, X. Wang, M. K. Khan, and P. Fan, "Efficient privacy-preserving authentication scheme with fine-grained error location for cloud-based vanet," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 436–10 449, 2021.

[28] J. Zhang, J. Cui, H. Zhong, Z. Chen, and L. Liu, "PA-CRT: chinese remainder theorem based conditional privacy-preserving authentication scheme in vehicular ad-hoc networks," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 2, pp. 722–735, 2021.

[29] H. Sikarwar and D. Das, "A novel mac-based authentication scheme (nomas) for internet of vehicles (iov)," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 4904–4916, 2023.

[30] V. Cini, S. Ramacher, D. Slamanig, C. Striecks, and E. Tairi, "Updatable signatures and message authentication codes," in *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 12710. Springer, 2021, pp. 691–723.

[31] D. Jost, U. Maurer, and M. Mularczyk, "Efficient ratcheting: Almost-optimal guarantees for secure messaging," in *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 11476. Springer, 2019, pp. 159–188.

[32] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, "Security analysis and improvements for the IETF MLS standard for group messaging," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 12170. Springer, 2020, pp. 248–277.

[33] Y. Dodis, H. Karthikeyan, and D. Wichs, "Updatable public key encryption in the standard model," in *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, ser. Lecture Notes in Computer Science, vol. 13044. Springer, 2021, pp. 254–285.

[34] H. Krawczyk and T. Rabin, "Chameleon hashing and signatures," Cryptology ePrint Archive, Paper 1998/010, 1998.

[35] G. Brassard, D. Chaum, and C. Crépeau, "Minimum disclosure proofs of knowledge," *Journal of Computer and System Sciences*, vol. 37, no. 2, pp. 156–189, 1988.

[36] J. Camenisch, D. Derler, S. Krenn, H. C. Pöhls, K. Samelin, and D. Slamanig, "Chameleon-hashes with ephemeral trapdoors," in *Public-Key Cryptography – PKC 2017*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 152–182.

[37] S. Xu, J. Ning, J. Ma, G. Xu, J. Yuan, and R. H. Deng, "Revocable policy-based chameleon hash," in *Computer Security – ESORICS 2021*. Cham: Springer International Publishing, 2021, pp. 327–347.

[38] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, pp. 111–128, 2013.

[39] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 514–532.

[40] N. Agrawal and S. Tapaswi, "Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3769–3795, 2019.

[41] J. Sun, G. Xu, T. Zhang, X. Cheng, X. Han, and M. Tang, "Secure data sharing with flexible cross-domain authorization in autonomous vehicle systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 7527–7540, 2023.

Xincheng Li received the bachelor's degree from Jiangsu Normal University, Xuzhou, China, in 2017. He is currently pursuing the Ph.D. degree in software engineering with the College of Information Engineering, Yangzhou University, Yangzhou, China. His research interests include the Internet of Things security and applied cryptography.

Xinchun Yin received the Ph.D. degree from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003. He is currently a Professor with the College of Information Engineering, Yangzhou University, Yangzhou, China. His research interests include cryptography, high-performance computing, and software quality assurance.

Jianting Ning (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, in 2016. He is currently a professor with the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Computer Science, Fujian Normal University, China. Previously, he was a research scientist with the School of Information Systems, Singapore Management University and a research fellow with the Department of Computer Science, National University of Singapore. His research interests include applied cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, NDSS, ESORICS, ACSAC, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, etc.

Tianwei Zhang (Member, IEEE) received the BSc degree from Peking University in 2011 and the Ph.D. degree from Princeton University in 2017. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanyang Technological University. His research focuses on computer system security, security threats and defenses in machine learning systems, autonomous systems, computer architecture, and distributed systems. He is serving on the editorial boards for IEEE Transactions on Circuits and Systems for Video Technology.