Published in IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 21, NO. 4, JULY/AUGUST 2024, pp. 1621-1638

AnoPay: Anonymous Payment for Vehicle Parking With Updatable Credential

Yang Yang, *Member, IEEE*, Wenyi Xue, Yonghua Zhan, Minming Huang, Yingjiu Li, and Robert H. Deng, *Fellow, IEEE*

***Abstract*—Many existing anonymous parking payment schemes lack high efficiency and flexibility. For instance, the calculation and communication costs involved in payment may linearly increase with the payment amount. In this paper, we propose an anonymous payment system (dubbed AnoPay) for vehicle parking, which leverages updatable attribute-based anonymous credentials and efficient zero-knowledge proof (ZKP) to achieve user anonymity and constant overhead for parking fee payment. To further improve the efficiency, we design a secure parking fee aggregation protocol based on linear homomorphic encryption to aggregate parking transactions, where the amount of each parking transaction is hidden and the privacy of the parking lot in terms of its revenue is guaranteed. AnoPay achieves both unlinkability and accountability, malicious payments can be efficiently traced when it is necessary. We provide a security model and rigorous proof for each security property of AnoPay. Extensive experiments and comparisons demonstrate the efficiency and practicality of the system.**

***Index Terms*—Anonymous payment, updatable credential, zero-knowledge proof, vehicle parking.**

I. Introduction

With the development of Internet of things (IoT) and artificial intelligence (AI) technologies, parking management is becoming increasingly digitized, automated and intelligent. Smart parking [1] is an emerging parking paradigm involving sensors, wireless communication, GPS positioning and artificial intelligence. As estimated in [2], the market size of parking industry will grow at a compound annual growth rate of 14% and reach US$3.8 billion by 2023. The advanced parking services effectively integrate the allocation, navigation and management of parking spaces to maximize the utilization of parking resources, improve parking efficiency and increase parking lot profit.

The development of parking technology not only brings great convenience to urban life, but also introduces privacy and security concerns on individual vehicles' (and their drivers') locations and trajectories. Recently, millions of drivers in Alexandria, VA are at risk of personal information leakage after a cyber breach on the parking app "ParkMobile" [3]. Due to system vulnerabilities, channel eavesdropping and poor management, drivers' personal details (such as phone numbers, e-mails, license plate numbers, parking records and vehicle descriptions) are leaked and may be used for illegal purposes. Most parking lots rely on third-party payment platforms (such as PayPal, Square, etc.) to charge parking fees. Adversaries who compromise these platforms may collect drivers' personal information, link parking bills, and profile drivers' behaviors and trajectories. The drivers adopting these traditional payment approaches are at the risk of personal data theft and abuse. Introducing anonymity into vehicle parking systems is necessary for protecting drivers' privacy. Detaching parking and payment records from drivers' personal identifiers would not only strengthen drivers' privacy but also reduce the risks caused by data breaches. Anonymous payment plays a critical role for privacy-preserving parking systems.

Besides anonymity requirements, efficiency and flexibility in payments are also primary design objectives for privacy-preserving parking systems. Many existing research works [4], [5], [6], [7] adopt anonymous currencies as the payment method. These payments are made using blind signature-based anonymous coins. In this case, the generation and verification of each coin requires constant overhead, and the total overhead of a payment increases with the number of involved coins. Moreover, flexibility is an issue. For instance, if a coin is worth \$10, the parking lot is hard to perform promotions (e.g., a 20% discount) or appropriate price adjustments (e.g., cutting the parking fee by \$1), because each coin is atomic. Other parking payment schemes [8], [9], [10], [11] adopt blockchain-based crypto-currencies. Schemes [8], [9], [10] are constructed on an Ethereum blockchain and adopt Ether to pay parking fees.

Manuscript received 11 July 2022; revised 12 April 2023; accepted 13 June 2023. Date of publication 19 June 2023; date of current version 11 July 2024. The work of Yang Yang and Robert Deng was supported in part by the National Natural Science Foundation of China under Grant 61872091, in part by the Fujian Provincial Natural Science Foundation under Grant 2023J01410193, in part by Singapore National Research Foundation under the National Satellite of Excellence in Mobile Systems Security and Cloud Security under Grant NRF2018NCR-NSOE004-0001, and in part by AXA Research Fund. The work of Yingjiu Li was supported by Ripple University Blockchain Research Initiative. *(Corresponding author: Yang Yang.)*

Yang Yang is with the College of Computer Science and Big Data, Fuzhou University, Fuzhou 350116, China, and also with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: yang.yang.research@gmail.com).

Wenyi Xue and Yonghua Zhan are with the College of Computer Science and Big Data, Fuzhou University, Fuzhou 350116, China (e-mail: wenyixue.research@gmail.com; Zhanyonghua@126.com).

Minming Huang and Robert H. Deng are with the School of Computing and Information Systems, Singapore Management University, Singapore 188065 (e-mail: mmhuang625@gmail.com; robertdeng@smu.edu.sg).

Yingjiu Li is with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403 USA (e-mail: yingjiul@uoregon.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TDSC.2023.3287228, provided by the authors.

Digital Object Identifier 10.1109/TDSC.2023.3287228

While such payment schemes are efficient, the addresses of payers are recorded in the Ethereum and thus different payments can be simply linked by comparing the payers' addresses. The scheme [11] is a Monero-type payment method [12] for unlinkable parking payment. However, the computation and communication overheads in [11] grow with the number of input accounts and account groups, and thus its efficiency is not fully satisfactory.

In addition to privacy-preserving parking payment, the privacy of parking lots' revenue is also indispensable. To be specific, each parking fee charged by parking lots should be kept confidential. By analyzing a parking lot's real-time income, business competitors may infer parking strategies and then take targeted competitive measures. In traditional digital payment applications, the specific parking income is public to the third-party payment platforms. If the data stored in these platforms are leaked, the privacy of parking lots' revenue will not be guaranteed. Schemes in [8], [9] also do not support this feature. The specific amount of each payment in [8], [9] is recorded on the blockchain to prevent double-spending and overdraft. Therefore, a secure aggregation method is necessary to protect the privacy of parking lots' revenue.

A. Contributions

In this paper, we propose an efficient anonymous parking payment system (AnoPay). Specifically, we design an efficient anonymous payment method by utilizing updatable anonymous credentials [13]. The payment process in AnoPay has a constant overhead and is independent to the amount of each payment. AnoPay implements anonymous payment by updating credential attributes, which is more efficient than the existing schemes while ensuring anonymity and unlinkability. To protect the privacy of parking lots, the linear homomorphic encryption scheme [14] is adopted in AnoPay to achieve secure aggregation of parking fees. Auditable anonymity is also achieved in AnoPay, in which anonymity of malicious drivers can be efficiently revoked by an arbitrator. To be specific, the contributions are described below.

- *Efficient anonymous and unlinkable parking payment:* We propose an efficient anonymous parking payment mechanism. AnoPay is designed with attribute-based credentials and its efficiency outperforms the existing coin-based schemes. We realize anonymous payment by updating credential attributes, thereby avoiding the overhead growth with parking fee increase. Our scheme is more efficient than existing coin-based parking payment schemes, since the top-up, payment and aggregation processes in our system can be completed within constant time.
- *Secure payment aggregation:* AnoPay supports secure aggregation of parking fee by combining linear homomorphic encryption and zero-knowledge proof. The parking lot can securely aggregate multiple parking fees and cash them from the credential issuer. The linear homomorphic encryption scheme ensures the privacy of each transaction, while the zero-knowledge proof ensures the correctness of aggregation.
- *Auditable anonymity:* AnoPay supports auditable anonymity. When an anonymous driver commits an illegal act, the credential issuer can disclose their identity by interacting with the arbitrator.
- *Double-spending Resistance:* Anopay supports efficient double-spending detection. A double-spending identifier is set as an attribute in drivers' credentials. The double-spending identifier must be disclosed for parking payment. Drivers who refuse to pay parking fees will be quickly detected and tracked by an arbitrator.

B. Related Work

Privacy-Preserving Vehicle Parking: Existing studies for privacy-preserving vehicle parking mainly focuses on vehicle authentication [15], [16], [17], [18], [19], [20], parking lot navigation [5], [17], [18], [21], [22], parking spaces sharing [5], [11], [23] and parking payment [4], [5], [6], [7], [8], [9], [10], [11]. Lu et al. [21] proposed a privacy-preserving parking management scheme, which provides real-time parking navigation, vehicle anti-theft and secure parking information releasing for drivers. To improve authentication efficiency, Liu et al. [15] proposed a novel vehicle authentication scheme based on aggregated-proofs, in which multiple parking vehicles can be authenticated simultaneously. Chim et al. [22] designed a real-time navigation scheme. The privacy of navigation results was protected by the combining proxy re-encryption technique. However, road-side units (RSUs) in [22] are not authenticated, which may be exploited by malicious attackers. To alleviate the situation of insufficient public parking spaces, Timpner et al. [23] suggested a dynamic and distributed method to construct decentralized parking communities. Vehicles in the same community help each other to obtain parking information securely. P-SPAN [17] is another secure and intelligent parking scheme with efficient parking navigation. By adopting anonymous credentials, P-SPAN [17] solves the problem of untrusted RSUs in [22] and provides accountability to unqualified drivers. Except protecting drivers' identity privacy and location privacy, Huang et al. [16] prevented a scheme for double-reservation of parking spaces with zero-knowledge proofs and proxy re-signatures. Ni et al. [18] presented a two-factor authentication protocol for automated valet parking. The proposed authentication protocol is based on one-time password and secure mobile devices so that the risk of vehicle theft is significantly reduced. Azees et al. [19] proposed an anonymous authentication protocol for vehicle ad hoc networks, where vehicles and RSUs are allowed to generate their credentials on their own, and the computational costs and message loss ratio are significantly reduced. To further optimize authentication efficiency, system in [20] achieves fast re-authentication by transferring anonymous authentication codes between consecutive RSUs and verifying authentication records with the Merkle hash tree. Chen et al. [41] proposed a novel and efficient offloading strategy for smart vehicle IoT Systems in Cloud-Edge Environments, which can efficiently make offloading decisions with module partition operations.

To support anonymous parking payment, Garra et al. [4] proposed a pay-by-phone parking system using RSA blind signatures in [24]. ASAP [5] is also an anonymous parking and payment system, which combines anonymous payment with secure parking navigation and parking lot sharing. By adopting

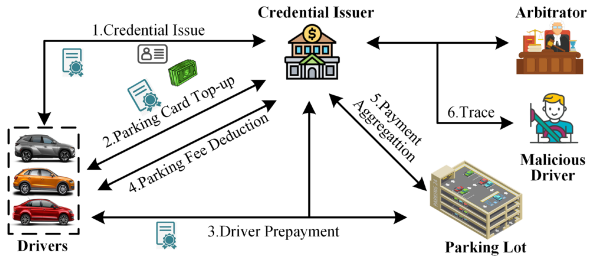


Fig. 1. System architecture.

the cut-and-choose technique, the scheme in [6] enhanced the robustness of parking at a price of efficiency. To solve the efficiency problem, Borges et al. [7] presented a novel pay-by-phone system, which supports drivers to revoke unused parking fees. Unfortunately, these schemes [4], [5], [6], [7] face the same problem that system overhead grows with parking charges. Another line of work [8], [9], [10] implemented anonymous parking payment by adopting Ethereum blockchain. However, payments through Ethereum are linkable, which is a serious privacy threat. Recently, Wang et al. [11] proposed a decentralized parking spot sharing scheme based on distributed anonymous credentials and Monero blockchain. The designed payment scheme hides transaction amounts and ensures unlinkability, but suffers a linear growth overhead as well.

Anonymous Payment: The first anonymous payment system (also called e-cash) was proposed by Chaum et al. [24] in 1983. The original study was fully untraceable and was extended for supporting accountability in [25]. Aiming at reducing the heavy storage overhead of coins, Camenisch et al. [26] proposed a compact e-cash system, in which 2^l coins can be stored using $O(l)$ storage space. Later, Vo et al. [27] designed a fair and transferable anonymous payment scheme. Transferability allows a coin to circulate among multiple users before being cashed in the bank. Unfortunately, the transfer process requires participation of a bank, which imposes a great burden on the bank when the amount of transfer coins is large. Wei et al. [28] distributed the transfer load to peers and utilized distributed hash tables (DHT) for distributed real-time double-spending detection. Canard et al. [29] proposed a divisible e-cash scheme, in which one coin can be divided and spent for consumption. Lian et al. [30] realized efficient tracing for compact e-cashes based on signatures of knowledge. However, the computation overhead in the above systems still grows with payment amount.

II. System and Threat Model

In this section, we present the system model, threat model, design goals and security model of our system. The notations used in our system are listed in Table I.

A. System Model

As shown in Fig. 1, the system consists of the following four entities: credential issuer ($\mathcal{CI}$), parking lot ($\mathcal{PL}$), driver ($\mathcal{DV}$) and arbitrator ($\mathcal{AR}$).

• *Credential Issuer ($\mathcal{CI}$)* is responsible to issue anonymous credentials for registered drivers (step ①), top-up $\mathcal{DV}$'s parking card (step ②), deduct parking fee from the parking card (step ④), and pay aggregated parking fees to $\mathcal{PL}$ (step ⑤). When any malicious behavior is discovered, $\mathcal{CI}$ discloses malicious driver's real identity under the supervision of $\mathcal{AR}$ (step ⑥).

• *Parking Lot ($\mathcal{PL}$)* provides parking services for anonymous drivers (step ③), in which $\mathcal{PL}$ verifies $\mathcal{DV}$'s anonymous credential, ensures that $\mathcal{DV}$'s parking card has sufficient balance and obtains a pre-payment receipt. $\mathcal{PL}$ interacts with $\mathcal{CI}$ to aggregate receipts and charge aggregated parking fees (step ⑤).

• *Driver ($\mathcal{DV}$)* registers to $\mathcal{CI}$ to obtain anonymous credential (i.e., parking card, step ①). Then, $\mathcal{DV}$ interacts with $\mathcal{CI}$ to top up the balance in his parking card (step ②). During the parking phase, $\mathcal{DV}$ interacts with $\mathcal{PL}$ to make a prepayment for the parking fee (step ③). Then, $\mathcal{CI}$ runs an interactive protocol with $\mathcal{DV}$ to deduct the parking fee from his card (step ④).

• *Arbitrator ($\mathcal{AR}$)* is a trusted entity, who helps $\mathcal{CI}$ to trace malicious drivers (step ⑥).

TABLE I
Summary of Notations

Notation	Description
$\mathcal{CI}/\mathcal{PL}$	Credential Issuer/Parking Lot
$\mathcal{DV}/\mathcal{AR}$	Driver/Arbitrator
$\mathsf{SK}_{ci}/\mathsf{PK}_{ci}$	private/public key of $\mathcal{CI}$
$\mathsf{SK}_{pl}/\mathsf{PK}_{pl}$	private/public key of $\mathcal{PL}$
$\mathsf{SK}_{dv}/\mathsf{PK}_{dv}$	private/public key of $\mathcal{DV}$
$\mathsf{SK}_{ar}/\mathsf{PK}_{ar}$	private/public key of $\mathcal{AR}$
cred_{dv}	anonymous credential of $\mathcal{DV}$
cred'_{dv}	new credential of cred_{dv} after randomization
$\mathsf{Cm}_{dv,1}, \mathsf{Cm}_{dv,2}$	commitments of $\mathcal{DV}$
$(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$	PS signature for generating cred_{dv}
$\Pi^1_{dv}/\Pi^2_{dv}/\Pi^3_{dv}/\Pi^4_{dv}$	ZKP generated by $\mathcal{DV}$
$(\Pi^1_{ci}, \Pi^2_{ci})/\Pi^1_{pl}/\Pi^1_{ar}$	ZKP generated by $\mathcal{CI}/\mathcal{PL}/\mathcal{AR}$
ID_{dv}	identity identifier of $\mathcal{DV}$
$dsid$	double-spending identifier
$dsid_{ci}/dsid_{dv}$	random components of $dsid$
val/inc	$\mathcal{DV}$'s account balance/top-up value
$chrg$	$\mathcal{DV}$'s parking charge
$Chrg$	aggregated parking charge
$\mathsf{cred}^*_{dv}/dsid^*/val^*$	updated $\mathsf{cred}_{dv}/dsid/val$
T_{dv}	transaction identifier
$x \xleftarrow{\$} \mathbb{Z}^*_p$	randomly choose an element x from $\mathbb{Z}^*_p$
$PoK\{\}$	a proof of knowledge protocol

B. Formal Definition

The AnoPay system consists of following algorithms.

• $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: This algorithm is performed by $\mathcal{CI}$. Taking a security parameter 1^λ as input, $\mathcal{CI}$ generates the system public parameters pp.

• $\mathsf{CI.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$: This algorithm is performed by $\mathcal{CI}$. It takes pp as input, and outputs a secret-public key pair $(\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$ for $\mathcal{CI}$.

• $\mathsf{PL.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{pl}, \mathsf{PK}_{pl})$: This algorithm is performed by $\mathcal{PL}$. It takes pp as input, and outputs a secret-public key pair $(\mathsf{SK}_{pl}, \mathsf{PK}_{pl})$ for $\mathcal{PL}$.

• $\mathsf{AR.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{ar}, \mathsf{PK}_{ar})$: This algorithm is performed by $\mathcal{AR}$. It takes pp as input, and outputs a secret-public key pair $(\mathsf{SK}_{ar}, \mathsf{PK}_{ar})$ for $\mathcal{AR}$.

• $\mathsf{DV.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{dv}, \mathsf{PK}_{dv})$: This algorithm is performed by $\mathcal{DV}$. It takes pp as input, and outputs a secret-public key pair $(\mathsf{SK}_{dv}, \mathsf{PK}_{dv})$ for $\mathcal{DV}$.

• $\mathsf{Issue}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{PK}_{dv}, ID_{dv}, val, \mathsf{pp})) \rightarrow (\mathsf{cred}_{dv}, dsid)$: $\mathcal{DV}$ interacts with $\mathcal{CI}$ to run this

algorithm. $\mathcal{DV}$ takes his secret-public key pair $(\mathsf{SK}_{dv}, \mathsf{PK}_{dv})$, his identity ID_{dv} and prepaid initial balance val as input. $\mathcal{CI}$ takes the secret-public key pair $(\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$ as input. The interaction returns an anonymous credential cred_{dv} and the corresponding double-spending identifier $dsid$ to $\mathcal{DV}$.

• $\mathsf{TopUp}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{cred}_{dv}, dsid, val, inc, \mathsf{pp})) \rightarrow (\mathsf{cred}^*_{dv}, val^*)$: $\mathcal{DV}$ interacts with $\mathcal{CI}$ to run this algorithm. $\mathcal{DV}$ takes his secret key SK_{dv}, the anonymous credential cred_{dv}, the double-spending identifier $dsid$, the parking card balance val and a top-up value inc as input. $\mathcal{CI}$ takes the secret-public key pair $(\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$ as input. The interaction returns an updated anonymous credential cred^*_{dv} with an updated parking card balance val^* to $\mathcal{DV}$.

• $\mathsf{Pre\text{-}Payment}(\mathcal{PL}(\mathsf{PK}_{pl}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{PK}_{ci}, \mathsf{PK}_{ar}, \mathsf{cred}_{dv}, dsid, val, chrg, \mathsf{pp})) \rightarrow (T_{dv}, E_{dv}, C_{dv})$: $\mathcal{DV}$ interacts with $\mathcal{PL}$ to run this algorithm. $\mathcal{DV}$ takes his secret key SK_{dv}, the anonymous credential cred_{dv}, the double-spending identifier $dsid$, the parking card balance val and parking charge $chrg$ as input. $\mathcal{PL}$ takes the secret-public key pair $(\mathsf{SK}_{pl}, \mathsf{PK}_{pl})$ as input. The interaction outputs a tuple (T_{dv}, E_{dv}, C_{dv}) to $\mathcal{PL}$ as the pre-payment receipt.

• $\mathsf{FeeDED}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{cred}_{dv}, dsid, val, chrg, T_{dv}, \mathsf{pp})) \rightarrow (\mathsf{cred}^*_{dv}, dsid^*, val^*)$: $\mathcal{DV}$ interacts with $\mathcal{CI}$ to run this algorithm. $\mathcal{DV}$ takes his secret key SK_{dv}, the anonymous credential cred_{dv}, the double-spending identifier $dsid$, the parking card balance val, parking charge $chrg$ and T_{dv} as input. $\mathcal{CI}$ takes the secret-public key pair $(\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$ as input. The interaction returns an updated anonymous credential cred^*_{dv}, an updated double-spending identifier $dsid$ and a deducted parking card balance val^* to $\mathcal{DV}$.

• $\mathsf{Aggregate}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{PL}(\mathsf{SK}_{pl}, \mathsf{PK}_{pl}, \{(T_{dv_i}, C_{dv_i}, E_{dv_i}), chrg_i\}_{i\in I}, \mathsf{pp})) \rightarrow Chrg$: $\mathcal{PL}$ interacts with $\mathcal{CI}$ to run this algorithm. $\mathcal{PL}$ collects the identifiers of pre-payment receipts as I. $\mathcal{PL}$ takes the secret-public key pair $(\mathsf{SK}_{pl}, \mathsf{PK}_{pl})$, the collected pre-payment receipts $\{(T_{dv_i}, C_{dv_i}, E_{dv_i})\}_{i\in I}$ and the corresponding parking charge $\{chrg_i\}_{i\in I}$ as input. $\mathcal{CI}$ takes the secret-public key pair $(\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$ as input. After the interactive computation, $\mathcal{CI}$ pays the aggregated parking charge $Chrg = \sum_{i\in I} chrg_i$ to $\mathcal{PL}$.

• $\mathsf{Trace}(\mathcal{AR}(\mathsf{SK}_{ar}, \mathsf{pp}) \leftrightarrow \mathcal{CI}(\mathsf{SK}_{ci}, T_{dv}, E_{dv}, C_{dv}, \mathsf{pp})) \rightarrow ID_{dv}$: $\mathcal{CI}$ interacts with $\mathcal{AR}$ to run this algorithm. $\mathcal{CI}$ takes the secret key SK_{ci} and a pre-payment receipt (T_{dv}, E_{dv}, C_{dv}) as input. $\mathcal{AR}$ takes the secret key SK_{ar} as input. The interaction outputs the real identity ID_{dv} of a malicious $\mathcal{DV}$, which is responsible for having generated the tuple (T_{dv}, E_{dv}, C_{dv}).

C. Threat and Security Model

In AnoPay, $\mathcal{CI}$ and $\mathcal{PL}$ are untrustworthy parties, who are interest in $\mathcal{DV}$'s privacy and may modify credentials and pre-payment receipts to gain higher profits. $\mathcal{DV}$ and adversary may modify or forge anonymous credentials to reduce or escape from his parking fee payment. The specific threat models are described as follows.

• *Credential Issuer ($\mathcal{CI}$)* is interested in $\mathcal{DV}$'s real identity and consumption status (e.g., the concrete parking expenses, the parking card balance). $\mathcal{CI}$ also attempts to link two parking fee deduction records to determine whether they are generated by the same driver. In issuing or updating credentials, $\mathcal{CI}$ may return $\mathcal{DV}$ a credential with a balance less than val. In Aggregate, $\mathcal{CI}$ may frame $\mathcal{PL}$ of requesting an incorrect aggregated parking charge and refuse to pay.

• *Parking Lot ($\mathcal{PL}$)* is interested in the real identity of anonymous drivers. $\mathcal{PL}$ may attempt to link two parking pre-payment receipts and obtain more profits from $\mathcal{CI}$ than it deserves. To get higher profits, $\mathcal{PL}$ may modify the pre-payment receipts or request incorrect aggregated parking charges.

• *Driver ($\mathcal{DV}$)* may modify the issued anonymous credential to increase the balance in his parking card. In the parking and pre-payment phase, he may submit a forged double-spending identifier to perform double-spending. He motivates to evade tracing if he is involved in a criminal case.

• *Adversary* eavesdrops on system communication channels and attempts to link eavesdropping data. Adversary may forge valid anonymous credentials without registering to $\mathcal{CI}$, use forged credentials for parking pre-payment and fee deduction without being traced.

The security models of AnoPay system are formally defined in terms of: unforgeability of anonymous credentials, anonymity and accountability. The concrete definitions of these security properties are presented in Supplemental Material A, available online.

D. Design Goals

AnoPay aims to achieve the following design goals that are crucial for a *privacy-preserving parking* system.

• *Authentication:* Unregistered drivers cannot forge valid anonymous credentials. Only approved drivers are entitled to anonymous parking payment services.

• *Auditable Anonymity:* $\mathcal{DV}$ remains to be anonymous during the top-up, pre-payment and deduction phases. The anonymity of malicious drivers can be removed under the supervision of $\mathcal{AR}$.

• *Unlinkability:* No parking payments performed by the same $\mathcal{DV}$ can be linked. To be specific, neither $\mathcal{PL}$ nor $\mathcal{CI}$ can judge whether any two pre-payment receipts or fee deduction records are generated by the same $\mathcal{DV}$.

• *Efficient Payment:* The execution of anonymous payment should be efficient. Time and communication overheads should be constant, which do not grow with the parking fee.

• *Double spending Resistance:* Double spending behavior can be quickly detected. The double-spending identifier in $\mathcal{DV}$'s anonymous credential cannot be modified before fee deduction.

• *Secure Aggregation:* Parking fees can be securely aggregated. $\mathcal{CI}$ cannot discover the specific parking fee in each pre-payment receipt. The aggregated result should be correct and kept confidential.

III. Preliminaries

A. Bilinear Pairing and Hardness Assumptions

Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map/pairing if the following properties hold: (1) *bilinearity:* for all $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$

and $a, b \in \mathbb{Z}_p^*, e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$. (2) *non-degeneracy:* for all $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$. (3) *computability:* for all $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$, $e(g, \widetilde{g})$ can be efficiently computed.

Assumption 1(LRSW assumption [31]) : Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear group of Type-III, and g, $\widetilde{g}$ are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. For $(X = g^x, Y = g^y, \widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$ where $x, y \in_R \mathbb{Z}_p^*$, we define an oracle $\mathcal{O}(m)$ on input $m \in \mathbb{Z}_p^*$ which chooses a random $h \in \mathbb{G}_1$ and outputs $T = (h, h^y, h^{x+my})$. We say that LRSW assumption holds if no adversary $\mathcal{A}$ with unlimited access to $\mathcal{O}$ can efficiently generate such a tuple for a new scalar $m^* \in \mathbb{Z}_p^*$ without querying m^* in $\mathcal{O}$.

Assumption 2(divisible decision Diffie-Hellman (DDDH) assumption [32]) : Given (g, g^a, g^b, r), where $g \in \mathbb{G}$ and $a, b, r \in_R \mathbb{Z}_p^*$, we define the advantage function of adversary $\mathcal{A}$ as: $\mathsf{Adv}_{\mathcal{A}}(\lambda)^{DDDH} = |\mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^{a/b}) = 1]| - |\mathsf{Pr}[\mathcal{A}(g, g^a, g^b, g^r) = 1]|$, where λ is the security parameter. We say that DDDH assumption holds if $\mathsf{Adv}_{\mathcal{A}}(\lambda)^{DDDH}$ is negligible.

Assumption 3(discrete logarithm (DL) assumption [33]) : Given a tuple (g, g^a), where $g \in \mathbb{G}$ and $a \in_R \mathbb{Z}_p^*$. DL assumption requires that the advantage for the adversary $\mathcal{A}$ to output the discrete logarithm a is negligible, i.e., $\mathsf{Adv}_{\mathcal{A}}(\lambda)^{DL} = \mathsf{Pr}[\mathcal{A}(g, g^a) = a] \leq negl(\lambda)$.

B. Zero-Knowledge Proof (ZKP)

Zero-knowledge proof is a cryptography method by which a prover is able to prove to a verifier that a statement is true without revealing any redundant information. Specifically, a proof of knowledge (PoK) protocol for a language $\mathcal{L}$ is represented as $PoK\{(w) : (w, x) \in R, x \in \mathcal{L}\}$, where the prover P keeps a witness w secretly and aims to convince the verifier V that the secret witness w and the public statement x meets a relation R. We say a proof is a zero-knowledge proof [34], [35] if the following properties hold:

• *Completeness*. If P knows a w with $(w, x) \in R$, V accepts the proof with probability at least $1 - \epsilon(x)$, where $\epsilon(x)$ is negligible.

• *Soundness*. If P is a cheating prover knowing nothing about w with $(w, x) \in R$, V will reject the proof with probability at least $1 - \epsilon(x)$, where $\epsilon(x)$ is negligible.

• *Auxiliary-input Zero-Knowledge*. For all probabilistic polynomial time (PPT) verifier V, there exists a PPT simulator M_V such that the distribution ensembles $\{\mathsf{P}(x), \mathsf{V}(x, y)\}_{x \in \mathcal{L}}$ and $\{\mathsf{M}_{\mathsf{V}}(x)\}_{x \in \mathcal{L}}$ are polynomially indistinguishable, where y is the auxiliary input for V [34].

C. PS Signature

PS-signature is a short randomizable signature proposed by Pointcheval and Sanders [31]. This signature has the same features as CL-signatures [36], [37]. However, the former has shorter signature size with only two group elements and higher computational efficiency in signing and verification. In this paper, PS-signature is introduced as a building block of our anonymous credentials. A multi-message blind signature scheme consists of the following polynomial-time algorithms:

• $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: The algorithm takes the security parameter λ as input, generates three Type-III bilinear groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, set $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$.

• $\mathsf{Keygen}(\mathsf{pp}, 1^n) \rightarrow (sk, pk)$: A signer randomly selects $g \xleftarrow{\$} \mathbb{G}_1$ and $\widetilde{g} \xleftarrow{\$} \mathbb{G}_2$, where $g \neq 1_{\mathbb{G}_1}$, $\widetilde{g} \neq 1_{\mathbb{G}_2}$ holds. He then chooses $(x, y_1, \ldots, y_n) \xleftarrow{\$} \mathbb{Z}_p^{n+1}$, computes $(X, Y_1, \ldots, Y_n) \leftarrow (g^x, g^{y_1}, \ldots, g^{y_n})$ and $(\widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_n) \leftarrow (\widetilde{g}^x, \widetilde{g}^{y_1}, \ldots, \widetilde{g}^{y_n})$. This algorithm outputs $sk = X$ and $pk = (g, Y_1, \ldots, Y_n, \widetilde{g}, \widetilde{X}, \widetilde{Y}_1, \ldots, \widetilde{Y}_n)$.

• $\mathsf{Commit}(\mathsf{pp}, pk, \{m_i\}_{i \in [n]}) \rightarrow cmt$: A user selects $t \xleftarrow{\$} \mathbb{Z}_p^*$ and computes a commitment $cmt = g^t \prod_{i=1}^n Y_i^{m_i}$. He outputs cmt and sends it to the signer.

• $\mathsf{Sign}(\mathsf{pp}, pk, sk, cmt) \rightarrow \sigma'$: Given a commitment cmt, a signer selects $u \xleftarrow{\$} \mathbb{Z}_p^*$ and signs the commitment by computing $\sigma' = (g^u, (X \cdot cmt)^u)$

• $\mathsf{Unblind}(\mathsf{pp}, pk, \sigma', \{m_i\}_{i \in [n]}, t) \rightarrow (\sigma / \perp)$: Given a signature σ', a user computes $\sigma = (\sigma_1, \sigma_2) = (\sigma'_1, \sigma'_2 (\sigma'_1)^{-t}) = (g^u, (X \prod_{i=1}^n Y_i^{m_i})^u)$ and checks $e(\sigma_1, \widetilde{X} \prod_{i=1}^n \widetilde{Y}_i^{m_i}) \stackrel{?}{=} e(\sigma_2, \widetilde{g})$. This algorithm outputs σ if the equation holds; otherwise, it aborts.

• $\mathsf{Verify}(\mathsf{pp}, pk, \sigma, \{m_i\}_{i \in [n]}) \rightarrow (1/0)$: A verifier checks the equation $e(\sigma_1, \widetilde{X} \prod_{i=1}^n \widetilde{Y}_i^{m_i}) \stackrel{?}{=} e(\sigma_2, \widetilde{g})$ and outputs 1 if it holds. Otherwise, he outputs 0.

D. Linear Homomorphic Encryption

Linear homomorphic encryption (LHE) [14] is an extension of Elgamal encryption [38]. The encryption scheme preserves the additive homomorphic property: $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$. A linear homomorphic encryption scheme consists of the following algorithms:

• $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: Given a security parameter λ, this algorithm generates a cyclic group G of prime order p. The public parameter is $pp = (G, p)$

• $\mathsf{Keygen}(\mathsf{pp}) \rightarrow (sk, pk)$: A user randomly chooses $g \xleftarrow{\$} G$ and $x, y \xleftarrow{\$} \mathbb{Z}_p^*$, computes $X = g^x, Y = g^y$. He sets his secret key $sk = (x, y)$ and public key $pk = (g, X, Y)$.

• $\mathsf{Enc}(\mathsf{pp}, pk, m) \rightarrow c$: To encrypt a message m, this algorithm selects $a, b \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $c = (c_1, c_2, c_3) = (X^a, Y^b, g^{a+b} m)$

• $\mathsf{Dec}(\mathsf{pp}, sk, c) \rightarrow m$: To decrypt a ciphertext c, this algorithm computes $m = c_3 \cdot (c_1^{\frac{1}{x}} \cdot c_2^{\frac{1}{y}})^{-1}$

E. Updatable Credential

Updatable anonymous credential systems (UACS) [13] is an attribute-based anonymous credential system (ABCS) with the property of privacy-preserving attribute renewing. In UACS, a user is able to interact with his issuer to update credential attributes without revealing the attributes to the issuer. To securely update a set of attributes, an update function ψ should be predefined. A user with attributes $\vec{A}$ and a hidden parameter α runs the update protocol with the issuer to obtain an updated credential with new attributes $\vec{A^*} = \psi(\vec{A}, \alpha)$. By performing a

ZKP protocol, the issuer assures that the update function ψ is executed properly, without learning any information about $\vec{A}$ and α. A UACS consists of the following algorithms:

• $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{cpp}$: The Setup algorithm generates the system public parameters cpp, which contains the public parameters of a blind signature scheme Π_{sig} and the arguments for a ZKP system.

• $\mathsf{IssuerKeyGen}(\mathsf{cpp}, 1^n) \rightarrow (sk, pk)$: In this algorithm, a issuer runs the KeyGen algorithm of Π_{sig} to obtain his secret-public key pair (sk, pk).

• $\mathsf{Issue}(\mathsf{cpp}, sk, \psi) \leftrightarrow \mathsf{Receive}(\mathsf{cpp}, pk, \psi, \alpha) \rightarrow cred$: In this algorithm, an issuer and a user engage in an interactive protocol to generate an anonymous credential $cred$. The user first generates an update parameter α where $\vec{A} = \psi(\perp, \alpha)$. He then performs a zero-knowledge proof protocol and a blind-signature protocol with the issuer to obtain a valid credential $cred$ associated with $\vec{A}$.

• $\mathsf{Update}(\mathsf{cpp}, sk, \psi) \leftrightarrow \mathsf{UpdRcv}(\mathsf{cpp}, pk, \psi, \alpha, cred) \rightarrow cred^*$: In this algorithm, a user interacts with an issuer to obtain an updated credential $cred^*$. The user first computes a update parameter α and a commitment cmt on $\vec{A}^*$. He then proves to the issuer that $\vec{A}^* = \psi(\vec{A}, \alpha)$ and $cmt = Commit(\vec{A}^*)$ by performing a ZKP protocol. If the proof is accepted, the issuer signs cmt to generate $cred^*$.

• $\mathsf{ShowPrv}(\mathsf{cpp}, \phi, \alpha, cred) \leftrightarrow \mathsf{ShowVrfy}(\mathsf{cpp}, pk, \phi) \rightarrow 1/0$: In this algorithm, a user and a verifier first agree on a predicate ϕ. Then, the user generates a show of $cred$ and sends it to the issuer along with a zero-knowledge proof. The issuer runs the Verify algorithm of Π_{sig} and checks the proof to ensure that $\phi(\vec{A}, \alpha) = 1$. The issuer outputs 1 if above verification passed, and outputs 0 otherwise.

IV. Proposed System

A. Overview of AnoPay

AnoPay consists of the following phases: system initialization phase, key generation phase, driver's credential issue phase, parking card top-up phase, parking and pre-payment phase, fee deduction phase, payment aggregation phase and driver tracing phase.

In system initialization phase, $\mathcal{CI}$ generates the system public parameters. Then, $\mathcal{CI}$, $\mathcal{PL}$, $\mathcal{AR}$ and $\mathcal{DV}$s generate their secret-public key pair respectively in key generation phase.

To be a legitimate user, $\mathcal{DV}$ sends his identity and public key to $\mathcal{CI}$ in driver's credential issue phase. $\mathcal{CI}$ verifies $\mathcal{DV}$'s identity and generates an anonymous credential for $\mathcal{DV}$. The obtained credential implicitly contains the parking card balance val and a double-spending identifier $dsid$. The subsequent parking card top-up phase is essentially the updating of the balance val in $\mathcal{DV}$'s credential.

The payment of parking fee involves two phases: pre-payment phase and deduction phase. When the car prepares to depart the parking lot and check out, $\mathcal{PL}$ calculates the car's parking fee according to the policies. Then, $\mathcal{DV}$ prepays the required fee to $\mathcal{PL}$ by generating a pre-payment receipt (with a double-spending identifier $dsid$), which proves that his balance val is sufficient for the payment. Next, $\mathcal{CI}$ runs an interactive protocol with $\mathcal{DV}$ to deduct parking fee from his parking card and update $dsid$ to $dsid^*$. $\mathcal{DV}$ cannot refuse fee deduction since $dsid$ in pre-payment receipt is recorded, and his misbehavior will be detected if $dsid$ is used twice.

In payment aggregation phase, $\mathcal{CI}$ pays the aggregated parking fee to $\mathcal{PL}$. To prevent $\mathcal{CI}$ from knowing the concrete parking charge in each parking transaction, the linear homomorphic encryption technique is utilized. The pre-payment receipt contains the encrypted parking charge $chrg$, which can be securely aggregated and decrypted through the cooperation of $\mathcal{CI}$ and $\mathcal{PL}$. $\mathcal{DV}$'s public key PK_{dv} is also encrypted and stored in the pre-payment receipt. If $\mathcal{DV}$ is detected a double-spending behavior or involved in a criminal case, his identity will be disclosed by $\mathcal{CI}$ in the driver tracing phase. To prevent honest drivers from being maliciously tracing, the tracing process requires the consent and participation of $\mathcal{AR}$.

The instances of the zero-knowledge proofs used in AnoPay are given in Supplemental Material B, available online.

B. Concrete Construction

1) System Initialization: $\mathcal{CI}$ runs Setup algorithm to generate the public parameters for the system.

• $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$: Given a security parameter 1^λ, $\mathcal{CI}$ generates a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$, $\mathbb{G}_2$ are groups of prime order p. Select the elements g, w that are generators of $\mathbb{G}_1$, and $\widetilde{g}$ a generator of $\mathbb{G}_2$. Choose a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The public parameter is denoted as $\mathsf{pp} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, w, \widetilde{g}, H)$.

2) Key Generation: In key generation phase, $\mathcal{CI}$, $\mathcal{PL}$, $\mathcal{AR}$ and $\mathcal{DV}$ generate their secret-public key pairs.

• $\mathsf{CI.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{ci}, \mathsf{PK}_{ci})$: $\mathcal{CI}$ randomly selects $x, y_1, y_2, y_3, z \xleftarrow{\$} \mathbb{Z}_p^*$, computes $(X, Y_1, Y_2, Y_3) = (g^x, g^{y_1}, g^{y_2}, g^{y_3})$, $(\widetilde{X}, \widetilde{Y}_1, \widetilde{Y}_2, \widetilde{Y}_3) = (\widetilde{g}^x, \widetilde{g}^{y_1}, \widetilde{g}^{y_2}, \widetilde{g}^{y_3})$ and $Z = w^{\frac{1}{z}}$. The secret key of $\mathcal{CI}$ is $\mathsf{SK}_{ci} = (X, y_1, y_2, y_3, z)$ and public key is $\mathsf{PK}_{ci} = (Y_1, Y_2, Y_3, \widetilde{X}, \widetilde{Y}_1, \widetilde{Y}_2, \widetilde{Y}_3, Z)$.

• $\mathsf{PL.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{pl}, \mathsf{PK}_{pl})$: $\mathcal{PL}$ randomly chooses $x_{pl} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Y_{pl} = w^{\frac{1}{x_{pl}}}$. The secret-public key pair of $\mathcal{PL}$ is denoted as $(\mathsf{SK}_{pl}, \mathsf{PK}_{pl}) = (x_{pl}, Y_{pl})$.

• $\mathsf{AR.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{ar}, \mathsf{PK}_{ar})$: $\mathcal{AR}$ computes $Y_{ar} = w^{\frac{1}{x_{ar}}}$, where x_{ar} is randomly chosen over $\mathbb{Z}_p^*$. The secret-public key pair of $\mathcal{AR}$ is $(\mathsf{SK}_{ar}, \mathsf{PK}_{ar}) = (x_{ar}, Y_{ar})$.

• $\mathsf{DV.KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{SK}_{dv}, \mathsf{PK}_{dv})$: $\mathcal{DV}$ selects $x_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $Y_{dv} = w^{x_{dv}}$. The secret-public key pair of $\mathcal{AR}$ is denoted as $(\mathsf{SK}_{dv}, \mathsf{PK}_{dv}) = (x_{dv}, Y_{dv})$.

3) Driver's Credential Issue: A driver $\mathcal{DV}$ should register his identity ID_{dv} and public key PK_{dv} to $\mathcal{CI}$ to be a legitimate system user. In the credential issuing phase, $\mathcal{DV}$ pays initial parking fee val to be stored in the parking card.

• $\mathsf{Issue}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{PK}_{dv}, ID_{dv}, val, \mathsf{pp})) \rightarrow (\mathsf{cred}_{dv}, dsid)$: Fig. 2 shows driver's credential issue protocol, which is conducted by the interaction between $\mathcal{DV}$ and $\mathcal{CI}$. $\mathcal{DV}$ randomly selects $dsid_{dv}, k_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$, where $dsid_{dv}$ is a component of $\mathcal{DV}$'s double-spending

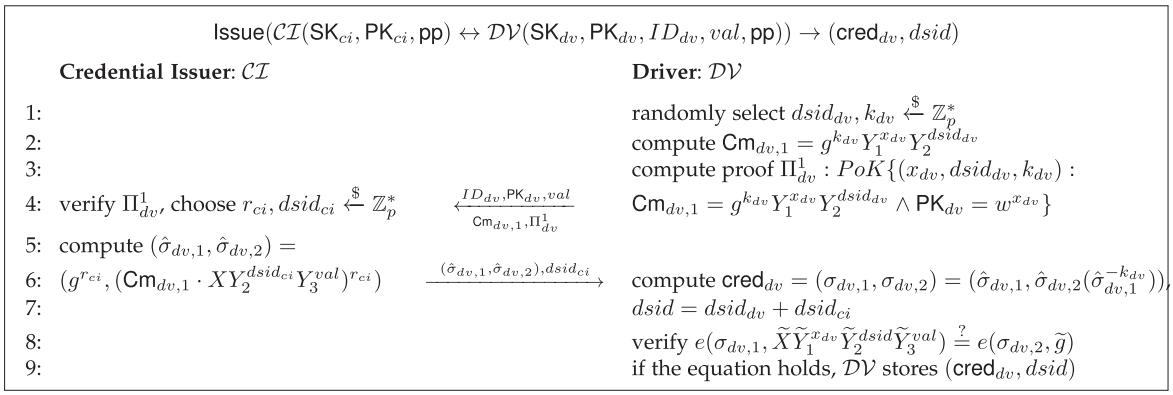


Fig. 2. Driver's credential issue protocol.

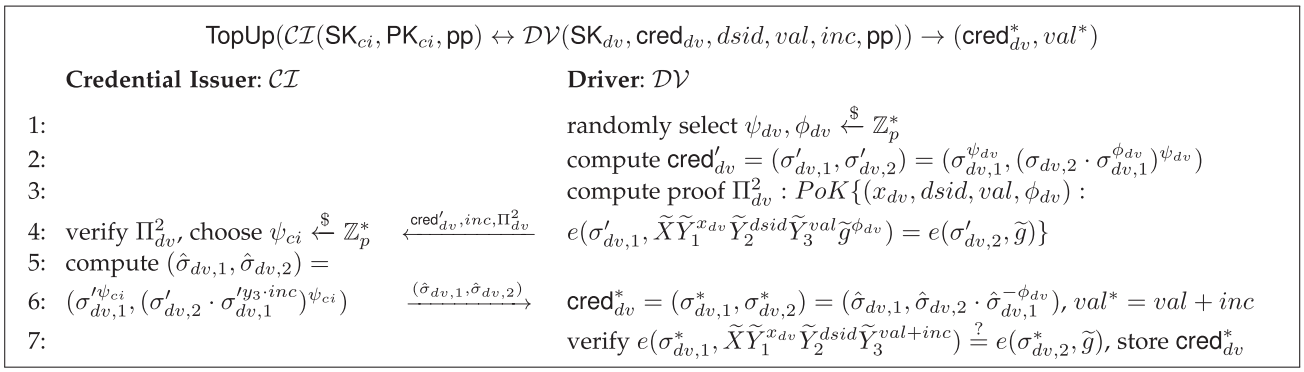


Fig. 3. Parking card top-up protocol.

identifier and k_{dv} is a nonce. Compute the commitment $\mathsf{Cm}_{dv,1} = g^{k_{dv}} Y_1^{x_{dv}} Y_2^{dsid_{dv}}$ and the corresponding zero knowledge proof Π_{dv}^1 on $(x_{dv}, dsid_{dv}, k_{dv})$. $\mathcal{DV}$ sends $(ID_{dv}, \mathsf{PK}_{dv}, val, \mathsf{Cm}_{dv,1}, \Pi_{dv}^1)$ to $\mathcal{CI}$. Receiving the credential issuing request from $\mathcal{DV}$, $\mathcal{CI}$ verifies the validity of Π_{dv}^1. If it is valid, $\mathcal{CI}$ generates a signature $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ on $\mathsf{Cm}_{dv,1}$: $\hat{\sigma}_{dv,1} = g^{r_{ci}}$, $\hat{\sigma}_{dv,2} = (\mathsf{Cm}_{dv,1} \cdot XY_2^{dsid_{ci}} Y_3^{val})^{r_{ci}}$, where $r_{ci}, dsid_{ci} \xleftarrow{\$} \mathbb{Z}_p^*$ and $dsid_{ci}$ is another component of $\mathcal{DV}$'s double-spending identifier. $\mathcal{CI}$ stores $(\mathsf{PK}_{dv}, ID_{dv})$ and sends $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}, dsid_{ci})$ to $\mathcal{DV}$. Receiving the tuple, $\mathcal{DV}$ derives the anonymous credential $\mathsf{cred}_{dv} = (\sigma_{dv,1}, \sigma_{dv,2})$: $\sigma_{dv,1} = \hat{\sigma}_{dv,1}$,

$$\begin{aligned}\sigma_{dv,2} &= \hat{\sigma}_{dv,2}(\hat{\sigma}_{dv,1}^{-k_{dv}}) = (\mathsf{Cm}_{dv,1} XY_2^{dsid_{ci}} Y_3^{val})^{r_{ci}} (\hat{\sigma}_{dv,1}^{-k_{dv}}) \\ &= (g^{k_{dv}} Y_1^{x_{dv}} \cdot XY_2^{dsid_{dv}+dsid_{ci}} Y_3^{val})^{r_{ci}} (\hat{\sigma}_{dv,1}^{-k_{dv}}).\end{aligned}$$

The double-spending identifier of $\mathcal{DV}$ is denoted as $dsid = dsid_{dv} + dsid_{ci}$. $\mathcal{DV}$ verifies the validity of cred_{dv} by checking $e(\sigma_{dv,1}, \widetilde{X}\widetilde{Y}_1^{x_{dv}} \widetilde{Y}_2^{dsid} \widetilde{Y}_3^{val}) \stackrel{?}{=} e(\sigma_{dv,2}, \widetilde{g})$, and stores $(\mathsf{cred}_{dv}, dsid)$ if this equation holds.

4) Parking Card Top-Up: When the parking card has insufficient balance, $\mathcal{DV}$ interacts with $\mathcal{CI}$ using TopUp protocol to top up his account, updating the balance in $\mathcal{DV}$'s anonymous credential cred_{dv}. To protect the privacy of driver, it is required that the parking card recharging is anonymous and unlinkable.

• $\mathsf{TopUp}(\mathcal{CI}(\mathsf{SK}_{ci},\mathsf{PK}_{ci},\mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv},\mathsf{cred}_{dv},dsid, val,inc,\mathsf{pp})) \rightarrow (\mathsf{cred}_{dv}^*, val^*)$: The TopUp protocol is shown in Fig. 3. If $\mathcal{DV}$ directly presents $\mathsf{cred}_{dv} = (\sigma_{dv,1}, \sigma_{dv,2})$ to $\mathcal{CI}$, the unlinkability cannot be guaranteed since $\mathcal{CI}$ is able to link the top-up activity with driver's anonymous credential. To ensure unlinkability, $\mathcal{DV}$ randomizes the original credential cred_{dv} with $\psi_{dv}, \phi_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$ to create a blinded credential $\mathsf{cred}'_{dv} = (\sigma'_{dv,1}, \sigma'_{dv,2}) = (\sigma_{dv,1}^{\psi_{dv}}, (\sigma_{dv,2} \cdot \sigma_{dv,1}^{\phi_{dv}})^{\psi_{dv}})$.
Then, $\mathcal{DV}$ generates the zero knowledge proof Π_{dv}^2 for $(x_{dv}, dsid, val, \phi_{dv})$. Suppose $\mathcal{DV}$ desires to top up value inc into the parking card. $\mathcal{DV}$ sends the tuple $(\mathsf{cred}'_{dv}, inc, \Pi_{dv}^2)$ as request to $\mathcal{CI}$.

After verifying the validity of Π_{dv}^2, $\mathcal{CI}$ randomly selects $\psi_{ci} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$: $\hat{\sigma}_{dv,1} = \sigma_{dv,1}'^{\psi_{ci}}$,

$$\begin{aligned}\hat{\sigma}_{dv,2} &= (\sigma'_{dv,2} \cdot \sigma_{dv,1}'^{y_3 \cdot inc})^{\psi_{ci}} \\ &= \hat{\sigma}_{dv,1}^{x + y_1 \cdot x_{dv} + y_2 \cdot dsid + y_3(val+inc) + \phi_{dv}}\end{aligned}$$

for $\mathcal{DV}$ with a new balance $val^* = val + inc$. $\mathcal{CI}$ sends $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ to $\mathcal{DV}$. Then, $\mathcal{DV}$ computes the updated credential $\mathsf{cred}_{dv}^* = (\sigma_{dv,1}^*, \sigma_{dv,2}^*)$, where $\sigma_{dv,1}^* = \hat{\sigma}_{dv,1}$, $\sigma_{dv,2}^* = \hat{\sigma}_{dv,2} \cdot \hat{\sigma}_{dv,1}^{-\phi_{dv}}$. If the equation $e(\sigma_{dv,1}^*, \widetilde{X}\widetilde{Y}_1^{x_{dv}} \widetilde{Y}_2^{dsid} \widetilde{Y}_3^{val+inc}) \stackrel{?}{=} e(\sigma_{dv,2}^*, \widetilde{g})$ holds, the updated credential cred_{dv}^* is valid and $\mathcal{DV}$ stores $(\mathsf{cred}_{dv}^*, val^*)$.

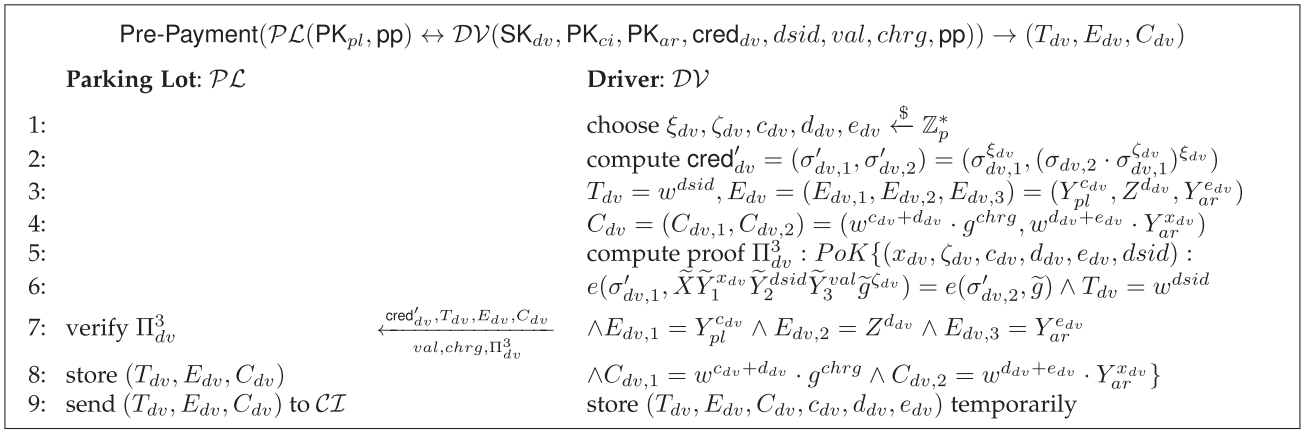


Fig. 4. Driver pre-payment protocol.

5) Driver Parking and Pre-Payment: When a vehicle departs a parking lot and checks out, $\mathcal{PL}$ calculates the parking fee $chrg$ for $\mathcal{DV}$ according to the charge policies. Then, $\mathcal{DV}$ generates a pre-payment receipt for $\mathcal{PL}$ using the Pre-Payment protocol. The fee deduction will be executed by $\mathcal{CI}$ using FeeDED protocol (described in next subsection).

• Pre-Payment$(\mathcal{PL}(\mathsf{PK}_{pl}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{PK}_{ci}, \mathsf{PK}_{ar}, \mathsf{cred}_{dv}, dsid, val, chrg, \mathsf{pp})) \rightarrow (T_{dv}, E_{dv}, C_{dv})$: $\mathcal{DV}$ interacts with $\mathcal{PL}$ to execute the Pre − Payment protocol (shown in Fig. 4). Specifically, $\mathcal{DV}$ chooses $\xi_{dv}, \zeta_{dv}, c_{dv}, d_{dv}, e_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$ and randomizes the original credential cred_{dv} to $\mathsf{cred}'_{dv} = (\sigma'_{dv,1}, \sigma'_{dv,2}) = (\sigma_{dv,1}^{\xi_{dv}}, (\sigma_{dv,2} \cdot \sigma_{dv,1}^{\zeta_{dv}})^{\xi_{dv}})$. Then, $\mathcal{DV}$ computes $T_{dv} = w^{dsid}$ to associate with the double-spending identifier $dsid$ in $\mathcal{DV}$'s credential. $\mathcal{DV}$ makes a commitment $C_{dv} = (C_{dv,1}, C_{dv,2})$ on the parking fee $chrg$: $C_{dv,1} = w^{c_{dv}+d_{dv}} \cdot g^{chrg}$ and $C_{dv,2} = w^{d_{dv}+e_{dv}} \cdot Y_{ar}^{x_{dv}}$, where x_{dv} is $\mathcal{DV}$'s secret key. $\mathcal{DV}$ generates the zero-knowledge proof Π_{dv}^3 for $(x_{dv}, val, \zeta_{dv}, c_{dv}, d_{dv}, e_{dv}, dsid)$ and sends the tuple $(\mathsf{cred}'_{dv}, T_{dv}, E_{dv}, C_{dv}, val, chrg, \Pi_{dv}^3)$ to $\mathcal{PL}$. Besides, the tuple $(T_{dv}, E_{dv}, C_{dv}, c_{dv}, d_{dv}, e_{dv})$ is temporarily stored by $\mathcal{DV}$ for parking fee deduction.

Upon receiving the data from $\mathcal{DV}$, $\mathcal{PL}$ checks the validity of Π_{dv}^3. It ensures that $\mathcal{DV}$ is a legitimate driver and affordable for the parking fee, i.e., the balance val in $\mathcal{DV}$'s anonymous credential cred_{dv} is no less than $chrg$. $\mathcal{PL}$ stores the received tuple if it passes the above verification. $\mathcal{PL}$ also sends the tuple (T_{dv}, E_{dv}, C_{dv}) to $\mathcal{CI}$ for parking fee deduction. Since $T_{dv} = w^{dsid}$ contains the double-spending identifier $dsid$, $\mathcal{CI}$ is able to detect the double-spending behavior by checking whether T_{dv} appears in existing transactions.

6) Parking Card Fee Deduction: After the pre-payment in a parking lot, $\mathcal{DV}$ should interact with $\mathcal{CI}$ to update the credential balance val and double-spending identifier $dsid$ in his anonymous credential cred_{dv}. Otherwise, double-spending behavior will be detected when $\mathcal{DV}$ parks next time. Note that the balance updating in credential indicates parking fee deduction.

• FeeDED$(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{cred}_{dv}, dsid, val, chrg, T_{dv}, \mathsf{pp})) \rightarrow (\mathsf{cred}_{dv}^*, dsid^*, val^*)$: The parking fee deduction FeeDED protocol is shown in Fig. 5, which is interactively executed between $\mathcal{DV}$ and $\mathcal{CI}$. $\mathcal{DV}$ chooses $\delta_{dv}, \epsilon_{dv}, \eta_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$ and calculates $\mathsf{cred}'_{dv} = (\sigma'_{dv,1}, \sigma'_{dv,2}) = (\sigma_{dv,1}^{\delta_{dv}}, (\sigma_{dv,2} \cdot \sigma_{dv,1}^{\epsilon_{dv}})^{\delta_{dv}})$. Besides, $\mathcal{DV}$ computes a commitment on updated double-spending identifier $dsid^* = dsid + e_{dv}$ and balance $val^* = val - chrg$ as $\mathsf{Cm}_{dv,2} = g^{\eta_{dv}} Y_1^{x_{dv}} Y_2^{dsid+e_{dv}} Y_3^{val-chrg}$, and generates the zero-knowledge proof Π_{dv}^4. Note that the elements c_{dv}, d_{dv}, e_{dv} used in $\mathsf{Cm}_{dv,2}$ and Π_{dv}^4 are random numbers generated in driver pre-payment protocol (in Fig. 4). $\mathcal{DV}$ sends $(\mathsf{cred}'_{dv}, T_{dv}, \mathsf{Cm}_{dv,2}, \Pi_{dv}^4)$ to $\mathcal{CI}$.

Remind that $\mathcal{PL}$ has sent the tuple (T_{dv}, E_{dv}, C_{dv}) to $\mathcal{CI}$ after the pre-payment of $\mathcal{DV}$. $\mathcal{CI}$ is able to link the two tuples sent by $\mathcal{DV}$ and $\mathcal{PL}$ using the element $T_{dv} = w^{dsid}$. If Π_{dv}^4 is valid, $\mathcal{CI}$ computes $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}) = (g^{\delta_{ci}}, (\mathsf{Cm}_{dv,2} X)^{\delta_{ci}})$, where δ_{ci} is a random number. $\mathcal{CI}$ returns $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ to $\mathcal{DV}$. $\mathcal{DV}$ derives the new credential $\mathsf{cred}_{dv}^* = (\sigma_{dv,1}, \sigma_{dv,2})$ as $\sigma_{dv,1} = \hat{\sigma}_{dv,1} = g^{\delta_{ci}}$,

$$\begin{aligned}\sigma_{dv,2} &= \hat{\sigma}_{dv,2} \cdot \hat{\sigma}_{dv,1}^{-\eta_{dv}} = (\mathsf{Cm}_{dv,2} X)^{\delta_{ci}} \cdot \hat{\sigma}_{dv,1}^{-\eta_{dv}} \\ &= (g^{\eta_{dv}} Y_1^{x_{dv}} Y_2^{dsid+e_{dv}} Y_3^{val-chrg} \cdot X)^{\delta_{ci}} \hat{\sigma}_{dv,1}^{-\eta_{dv}}.\end{aligned}$$

$\mathcal{DV}$ checks $e(\sigma_{dv,1}, \widetilde{X}\widetilde{Y}_1^{x_{dv}}\widetilde{Y}_2^{dsid^*}\widetilde{Y}_3^{val^*}) \stackrel{?}{=} e(\sigma_{dv,2}, \widetilde{g})$. If the equation holds, $\mathcal{DV}$ stores $(\mathsf{cred}_{dv}^*, dsid^*, val^*)$, where the updated $dsid^* = dsid + e_{dv}$ and $val^* = val - chrg$.

7) Payment Aggregation: $\mathcal{PL}$ periodically interacts with $\mathcal{CI}$ for secure aggregation of parking fee transactions. After the aggregation, $\mathcal{CI}$ should transfer the total parking fee to $\mathcal{PL}$. Leveraging a linear homomorphic encryption, AnoPay prevents $\mathcal{CI}$ from knowing the concrete parking charge in each parking transaction.

• Aggregate$(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{PL}(\mathsf{SK}_{pl}, \mathsf{PK}_{pl}, \{(T_{dv_i}, C_{dv_i}, E_{dv_i}), chrg_i\}_{i \in I}, \mathsf{pp})) \rightarrow Chrg$: The Aggregate protocol is shown in Fig. 6. $\mathcal{PL}$ collects the identifiers of the parking transactions and inserts them into a list $\mathcal{L} = \{T_{dv_i}\}_{i \in I}$. $\mathcal{PL}$ calculates $C_{pl} = (\prod_{i \in I} E_{dv_i,1})^{x_{pl}} = (\prod_{i \in I} Y_{pl}^{c_{dv_i}})^{x_{pl}} = \prod_{i \in I} w^{c_{dv_i}}$ and a zero-knowledge proof $\Pi_{pl}^1 : PoK\{(x_{pl}) : C_{pl} = (\prod_{i \in I} E_{dv_i,1})^{x_{pl}} \wedge w = Y_{pl}^{x_{pl}}\}$. $\mathcal{PL}$

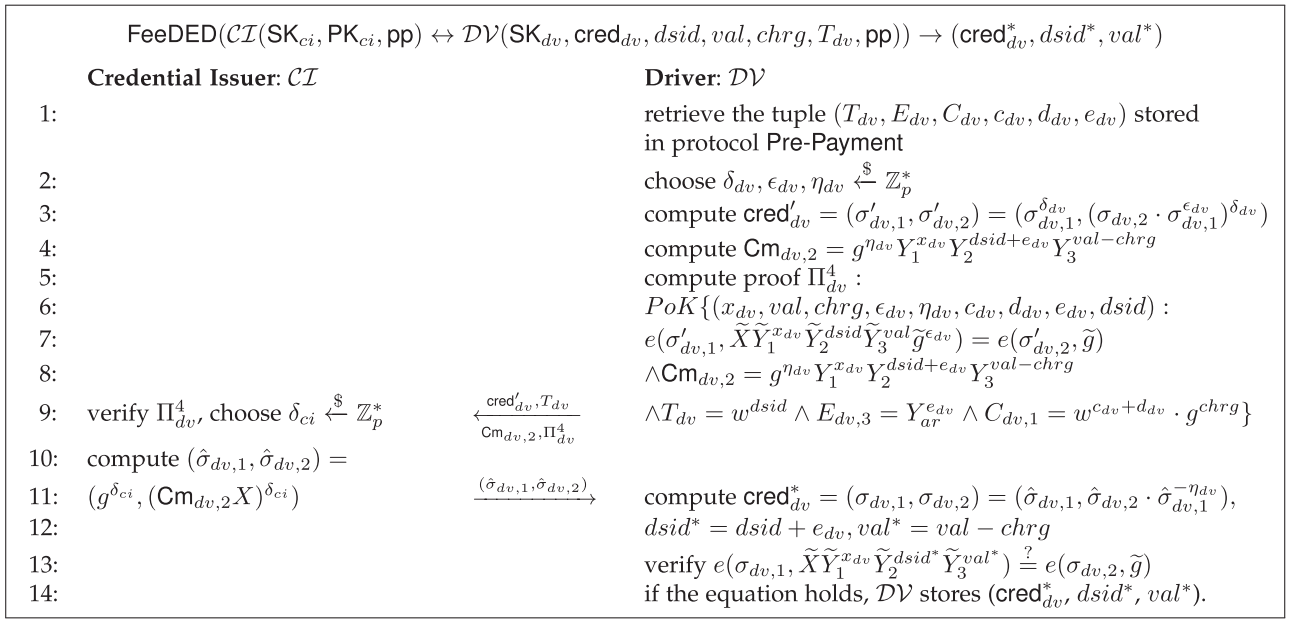


Fig. 5. Parking fee deduction protocol.

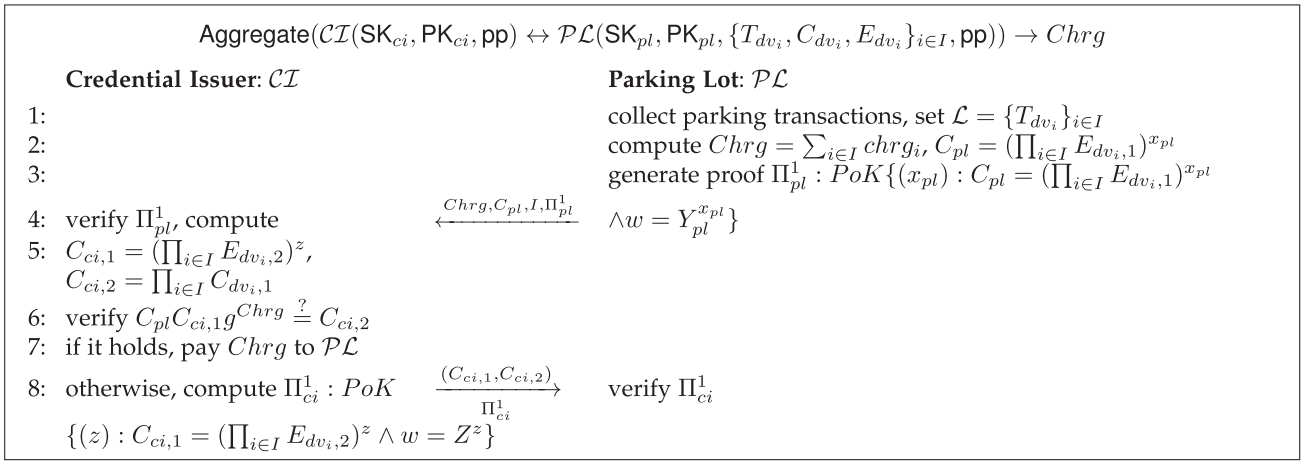


Fig. 6. Payment aggregation protocol.

sends the tuple $(I, Chrg, C_{pl}, \Pi^1_{pl})$ to $\mathcal{CI}$. After verifying the validity of Π^1_{pl}, $\mathcal{CI}$ computes $C_{ci,1} = (\prod_{i\in I} E_{dv_i,2})^z = (\prod_{i\in I} Z^{d_{dv_i}})^z = \prod_{i\in I} w^{d_{dv_i}}$, $C_{ci,2} = \prod_{i\in I} C_{dv_i,1} = \prod_{i\in I}(w^{c_{dv_i}+d_{dv_i}} \cdot g^{chrg_i}) = (\prod_{i\in I} w^{c_{dv_i}+d_{dv_i}}) g^{\sum_{i\in I} chrg_i}$. The total parking fee that $\mathcal{CI}$ should pay to $\mathcal{PL}$ is $Chrg = \sum_{i\in I} chrg_i$. Then, $\mathcal{CI}$ checks $C_{pl} C_{ci,1} g^{Chrg} \stackrel{?}{=} C_{ci,2}$. If above equation holds, $\mathcal{CI}$ pays the bill $Chrg$ to $\mathcal{PL}$. Otherwise, $\mathcal{CI}$ computes a proof $\Pi^1_{ci} : PoK\{(z) : C_{ci,1} = (\prod_{i\in I} E_{dv_i,2})^z \wedge w = Z^z\}$. $\mathcal{CI}$ refuses to pay $Chrg$ and returns $(C_{ci,1}, C_{ci,2}, \Pi^1_{ci})$ to $\mathcal{PL}$, where the proof Π^1_{ci} proves that $\mathcal{CI}$'s faithfully obey the Aggregate protocol and the refusal is justified.

8) Driver Tracing: When a malicious $\mathcal{DV}$ is detected a double-spending behavior or involved in a criminal case, it is vital to recover the driver's real identity from the parking pre-payment record. If the driver tracing algorithm is independently executed by $\mathcal{CI}$, the privacy of driver may be violated in normal situation. To avoid the abuse of accountability, the malicious $\mathcal{DV}$ tracing algorithm should be executed by the cooperation of $\mathcal{CI}$ and an arbitrator $\mathcal{AR}$.

• $\mathsf{Trace}(\mathcal{AR}(\mathsf{SK}_{ar}, \mathsf{pp}) \leftrightarrow \mathcal{CI}(\mathsf{SK}_{ci}, T_{dv}, E_{dv}, C_{dv}, \mathsf{pp})) \rightarrow (\mathsf{PK}_{dv}, ID_{dv})$: This algorithm takes the suspicious pre-payment record (T_{dv}, E_{dv}, C_{dv}) as input. $\mathcal{CI}$ computes $C_{ci,3} = (E_{dv,2})^z$, $\Pi^2_{ci} : PoK\{(z) : C_{ci,3} = (E_{dv,2})^z \wedge w = Z^z\}$, where z is an element of $\mathcal{CI}$'s secret key SK_{ci}. $\mathcal{CI}$ sends the suspicious pre-payment record (T_{dv}, E_{dv}, C_{dv}) and $(C_{ci,3}, \Pi^2_{ci})$ to $\mathcal{AR}$. If the tracing request is judged as reasonable and Π^2_{ci} passes the verification, $\mathcal{AR}$ computes $C_{ar} = (E_{dv,3})^{x_{ar}}$, $\mathsf{PK}_{dv} = (\frac{C_{dv,2}}{C_{ci,3} C_{ar}})^{x_{ar}}$ and $\Pi^1_{ar} : PoK\{(x_{ar}) : C_{ar} = (E_{dv,3})^{x_{ar}} \wedge \mathsf{PK}_{dv} = (\frac{C_{dv,2}}{C_{ci,3} C_{ar}})^{x_{ar}} \wedge w = Y_{ar}^{x_{ar}}\}$, which

are returned to $\mathcal{CI}$. Then, $\mathcal{CI}$ verifies Π^1_{ar} to ensure that PK_{dv} is derived from the suspicious pre-payment record (T_{dv}, E_{dv}, C_{dv}). As the tuple $(\mathsf{PK}_{dv}, ID_{dv})$ is stored by $\mathcal{CI}$ in $\mathcal{DV}$'s credential issue phase, it is convenient to reveal $\mathcal{DV}$'s real identity ID_{dv} from PK_{dv}.

V. Security Analysis

A. Unforgeability of Anonymous Credential

Theorem 1: The anonymous credential in AnoPay is secure under existential unforgeability under chosen message attack (EUF-CMA) if the LRSW assumption holds in Type-III bilinear group.

Proof: Assuming there exists a PPT adversary $\mathcal{A}$ that can break the security game *(Forgery of Anonymous Credential)* with non-negligible probability ϵ, we can construct a PPT challenger $\mathcal{C}$ to break the LRSW assumption with non-negligible probability $\epsilon' = \epsilon$.

• *Setup and Keygen Phase:* The challenger $\mathcal{C}$ is given an instance of LRSW: $(g, Y_1 = g^{y_1}, \widetilde{g}, \widetilde{X} = \widetilde{g}^{x}, \widetilde{Y}_1 = \widetilde{g}^{y_1})$ and an oracle $\mathcal{O}(m)$ which takes $m \in \mathbb{Z}^*_p$ as input and outputs a pair (h, h^{x+my_1}). $\mathcal{C}$ selects $y_2, y_3, z, x_{pl}, x_{ar} \xleftarrow{\$} \mathbb{Z}^*_p$, $w \xleftarrow{\$} \mathbb{G}_1$, computes $Y_2 = g^{y_2}, \widetilde{Y}_2 = \widetilde{g}^{y_2}, Y_3 = g^{y_3}, \widetilde{Y}_3 = \widetilde{g}^{y_3}, Z = w^{\frac{1}{z}}, X_{pl} = w^{\frac{1}{x_{pl}}}, X_{ar} = w^{\frac{1}{x_{ar}}}$. $\mathcal{C}$ sets $pp = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, w, \widetilde{g})$, $\mathsf{PK}_{ci} = (Y_1, Y_2, Y_3, \widetilde{X}, \widetilde{Y}_1, \widetilde{Y}_2, \widetilde{Y}_3, Z)$, $\mathsf{PK}_{pl} = X_{pl}$, $\mathsf{PK}_{ar} = X_{ar}$, stores $(y_2, y_3, z, x_{pl}, x_{ar})$ and outputs $(pp, \mathsf{PK}_{ci}, \mathsf{PK}_{pl}, \mathsf{PK}_{ar})$ to $\mathcal{A}$. Besides, $\mathcal{C}$ constructs an extractor $\mathcal{E}$ for the proof of knowledge system.

• *Query Phase:* The challenger initializes an empty set $\mathcal{D}$ and an empty table $\mathcal{T}$. The adversary adaptively issues the following queries.

- *Credential Issue Query:* $\mathcal{A}$ submits a public key PK_{dv}, an identity ID_{dv} and an initial balance val as a query. If $\mathsf{PK}_{dv} \in \mathcal{D}$, $\mathcal{C}$ aborts. Then, $\mathcal{C}$ runs the protocol Issue and obtains a tuple $(ID_{dv}, \mathsf{PK}_{dv}, val, \mathsf{Cm}_{dv,1}, \Pi^1_{dv})$ from $\mathcal{A}$. If the proof Π^1_{dv} does not pass the verification, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ runs $\mathcal{E}$ to extract the witness $(x_{dv}, dsid_{dv}, k_{dv})$ from Π^1_{dv}. $\mathcal{C}$ queries $\mathcal{O}(m)$ with x_{dv} and obtains a pair $(h, h' = h^{x+y_1 \cdot x_{dv}})$. $\mathcal{C}$ selects $dsid_{ci} \xleftarrow{\$} \mathbb{Z}^*_p$, constructs $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}) = (h, h' h^{k_{dv}+y_2(dsid_{dv}+dsid_{ci})+y_3 \cdot val}) = (h, h^{k_{dv}+x+y_1 \cdot x_{dv}+y_2(dsid_{dv}+dsid_{ci})+y_3 \cdot val})$ and sends $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}, dsid_{ci})$ to $\mathcal{A}$. Then, $\mathcal{C}$ inserts $PK_{dv} = g^{x_{dv}}$ into $\mathcal{D}$.

- *Parking Card Top-up Query:* $\mathcal{A}$ submits an anonymous credential cred_{dv} and an incremental value inc as a query. $\mathcal{C}$ runs the protocol TopUp and obtains a tuple $(\mathsf{cred}'_{dv}, inc, \Pi^2_{dv}) = ((\sigma'_{dv,1}, \sigma'_{dv,2}), inc, \Pi^2_{dv})$ from $\mathcal{A}$. If the proof Π^2_{dv} does not pass the verification, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ selects $\psi_{ci} \xleftarrow{\$} \mathbb{Z}^*_p$, computes $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}) = (\sigma'^{\psi_{ci}}_{dv,1}, (\sigma'_{dv,2} \cdot \sigma'^{y_3 \cdot inc}_{dv,1})^{\psi_{ci}})$ and sends $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ to $\mathcal{A}$.

- *Driver Parking and Pre-Payment Query:* $\mathcal{A}$ submits an anonymous credential cred_{dv} and a parking charge $chrg$ as a query. $\mathcal{C}$ runs the protocol $\mathsf{Pre\text{-}Payment}$ and obtains a tuple $(\mathsf{cred}'_{dv}, T_{dv}, E_{dv}, C_{dv}, val, chrg, \Pi^3_{dv})$ from $\mathcal{A}$. If the proof Π^3_{dv} does not pass the verification, $\mathcal{C}$ aborts. Otherwise, the challenger $\mathcal{C}$ inserts (T_{dv}, E_{dv}, C_{dv}) to $\mathcal{T}$ and returns T_{dv} to $\mathcal{A}$.

- *Parking Card Fee Deduction Query:* $\mathcal{A}$ submits an anonymous credential cred_{dv} and T_{dv} as a query. $\mathcal{C}$ searches the entry (T_{dv}, E_{dv}, C_{dv}) in $\mathcal{T}$ with the specified T_{dv}. If such entry exists, $\mathcal{C}$ runs the protocol FeeDED to obtain a tuple $(\mathsf{cred}'_{dv}, T_{dv}, \mathsf{Cm}_{dv,2}, \Pi^4_{dv})$ from $\mathcal{A}$; otherwise, $\mathcal{C}$ aborts. If the proof Π^4_{dv} dose not pass the verification, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ runs $\mathcal{E}$ to extract the witness $(x_{dv}, val, chrg, \epsilon_{dv}, \eta_{dv}, c_{dv}, d_{dv}, e_{dv}, dsid)$ from Π^4_{dv}. $\mathcal{C}$ queries $\mathcal{O}(m)$ with x_{dv} and obtains a pair $(h, h' = h^{x+y_1 \cdot x_{dv}})$. $\mathcal{C}$ constructs $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2}) = (h, h' h^{\eta_{dv}+y_2(dsid+e_{dv})+y_3(val-chrg)}) = (h, h^{eta_{dv}+x+y_1 \cdot x_{dv}+y_2(dsid+e_{dv})+y_3(val-chrg)})$ and sends $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ to $\mathcal{A}$.

• *Forge:* $\mathcal{A}$ outputs an anonymous credential $\mathsf{cred}_{DV^*} = (\sigma_{dv^*,1}, \sigma_{dv^*,2})$, a secret-public key-pair $(\mathsf{SK}_{dv^*}, \mathsf{PK}_{dv^*}) = (x_{dv^*}, g^{x_{dv^*}})$, a double-spending identifier $dsid^*$ and a balance value val^* as a query, with the restriction that PK_{dv^*} corresponds to SK_{dv^*} and $\mathsf{PK}_{dv^*} \notin \mathcal{D}$. Define $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game. If $\mathbb{E}_{\mathcal{A}}$ happens, the equation $e(\sigma_{dv^*,1}, \widetilde{X}\widetilde{Y}_1^{x_{dv^*}} \widetilde{Y}_2^{dsid^*} \widetilde{Y}_3^{val^*}) = e(\sigma_{dv^*,2}, \widetilde{g})$ holds. Then, we have

$$\begin{aligned} e(\sigma_{dv^*,2}, \widetilde{g}) &= e(\sigma_{dv^*,1}, \widetilde{X}\widetilde{Y}_1^{x_{dv^*}} \widetilde{Y}_2^{dsid^*} \widetilde{Y}_3^{val^*}) \\ &= e(\sigma_{dv^*,1}, \widetilde{g})^{x+y_1 \cdot x_{dv^*}+y_2 \cdot dsid^*+y_3 \cdot val^*} \\ &= e(\sigma_{dv^*,1}^{x+y_1 \cdot x_{dv^*}+y_2 \cdot dsid^*+y_3 \cdot val^*}, \widetilde{g}). \end{aligned}$$

Denote $\sigma_{dv^*,1}$ as h^*, $\sigma_{dv^*,2}$ as $(h^*)^{x+y_1 \cdot x_{dv^*}+y_2 \cdot dsid^*+y_3 \cdot val^*}$. $\mathcal{C}$ can then break the LRSW assumption by constructing a pair $(\hat{h^*}, \hat{h'^*}) = (\sigma_{dv^*,1}, \sigma_{dv^*,2}/(\sigma_{dv^*,1})^{y_2 \cdot dsid^*+y_3 \cdot val^*}) = (h^*, (h^*)^{x+y_1 \cdot x_{dv^*}})$. Since $\mathsf{PK}_{dv^*} = g^{x_{dv^*}}$ does not exist in $\mathcal{D}$, then x_{dv^*} has not been queried to $O(m)$ by the challenger $\mathcal{C}$. Thus, the pair $(\hat{h^*}, \hat{h'^*})$ is a well constructed solution for the LRSW assumption.

• *Probability Analysis:* Let $\mathbb{E}_{LRSW}$ be the event that $(\hat{h^*}, \hat{h'^*})$ is a well constructed solution for the LRSW assumption, $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game. $\epsilon' = Pr[\mathbb{E}_{LRSW}] = Pr[\mathbb{E}_{LRSW}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + Pr[\mathbb{E}_{LRSW}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + \epsilon = \epsilon$.

Theorem 2: The anonymous credential in AnoPay is secure under impersonation attack if the DL assumption holds in Type-III bilinear group.

Proof: Assuming there exists a PPT adversary $\mathcal{A}$ that can break the security game *(Impersonation of Driver)* with non-negligible probability ϵ, we can construct a PPT challenger $\mathcal{C}$ to break the DL assumption with non-negligible probability $\epsilon' = \epsilon$.

• *Setup and Keygen Phase:* The challenger $\mathcal{C}$ is given an instance of DL problem: $(w, A = w^a)$. $\mathcal{C}$ runs Setup and sets the system parameter $\mathsf{pp} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, w, \widetilde{g})$. $\mathcal{C}$ runs $\mathsf{CIKeygen}$, $\mathsf{PLKeygen}$, $\mathsf{ARKeygen}$ to generate $(\mathsf{PK}_{ci}, \mathsf{SK}_{ci}, \mathsf{PK}_{pl}, \mathsf{SK}_{pl}, \mathsf{PK}_{ar}, \mathsf{SK}_{ar})$, and sends $(\mathsf{pp}, \mathsf{PK}_{ci}, \mathsf{PK}_{pl}, \mathsf{PK}_{ar}, \mathsf{SK}_{ci}, \mathsf{SK}_{pl}, \mathsf{SK}_{ar})$ to $\mathcal{A}$. Besides, $\mathcal{C}$ constructs a simulator $\mathcal{S}$ for the proof of knowledge system.

• *Query Phase:* The challenger $\mathcal{C}$ initializes two empty tables $\mathcal{T}_1$ and $\mathcal{T}_2$. The adversary $\mathcal{A}$ adaptively issues the following queries.

- *Credential Issue Query:* $\mathcal{A}$ submits an identity ID_{dv} and an initial balance val to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ selects $\alpha_{dv}, k_{dv}, dsid_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathsf{Cm}_{dv,1} \xleftarrow{\$} \mathbb{G}_1$ and sets $\mathsf{PK}_{dv} = A^{\alpha_{dv}} = w^{a \cdot \alpha_{dv}}$. Then, $\mathcal{C}$ runs the simulator $\mathcal{S}$ to generate Π_{dv}^1 and runs the protocol Issue with the adversary $\mathcal{A}$ to obtain $(\mathsf{cred}_{dv}, dsid)$. $\mathcal{C}$ inserts $(ID_{dv}, \alpha_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ to $\mathcal{T}_1$.

- *Parking Card Top-up Query:* $\mathcal{A}$ submits an identity ID_{dv} and a incremental value inc to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves the entry $(ID_{dv}, \alpha_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$ and runs TopUp to obtain $(\mathsf{cred}_{dv}^*, val^*)$. Then, the challenger updates the entry with $(ID_{dv}, \alpha_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}^*, dsid, val^*)$. As $\mathcal{C}$ does not hold the secret key SK_{dv}, the proof Π_{dv}^2 in TopUp is produced by running $\mathcal{S}$.

- *Driver Parking and Pre-Payment Query:* $\mathcal{A}$ submits an identity ID_{dv} and a parking charge $chrg$ to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves the entry $(ID_{dv}, \alpha_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$. $\mathcal{C}$ selects $\xi_{dv}, \zeta_{dv}, c_{dv}, d_{dv}, e_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $T_{dv}, E_{dv}, C_{dv,1}$ as usual, and computes $C_{dv,2} = w^{d_{dv}+e_{dv}} \cdot A^{\alpha_{dv} \cdot \frac{1}{x_{ar}}} = w^{d_{dv}+e_{dv}} \cdot Y_{ar}^{a \cdot \alpha_{dv}}$, where x_{ar} is $\mathcal{AR}$'s secret key SK_{ar}. Then, $\mathcal{C}$ runs $\mathcal{S}$ to simulate Π_{dv}^3, sends the tuple $(\mathsf{cred}'_{dv}, T_{dv}, E_{dv}, C_{dv}, val, chrg, \Pi_{dv}^3)$ to $\mathcal{A}$ and inserts $(T_{dv}, E_{dv}, C_{dv}, ID_{dv}, chrg)$ into $\mathcal{T}_2$.

- *Parking Card Fee Deduction Query:* $\mathcal{A}$ submits a transaction identifier T_{dv} as a query. If the entry specified by T_{dv} does not exist in $\mathcal{T}_2$, $\mathcal{C}$ aborts. Otherwise, the challenger $\mathcal{C}$ retrieves the entry $(T_{dv}, E_{dv}, C_{dv}, ID_{dv}, chrg)$ from $\mathcal{T}_2$ and retrieves the corresponding entry $(ID_{dv}, \alpha_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$. $\mathcal{C}$ runs FeeDED to obtain $(\mathsf{cred}_{dv}^*, dsid^*, val^*)$. Similar to the **Parking Card Top-up Query**, the commitment $\mathsf{Cm}_{dv,2}$ is randomly chosen from $\mathbb{G}_1$ and the proof Π_{dv}^4 is generated by running $\mathcal{S}$. After the interaction, $\mathcal{C}$ updates the entry specified by ID_{dv} in $\mathcal{T}_1$ to $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}^*, dsid^*, val^*)$.

• *Forge:* $\mathcal{A}$ outputs an identity ID_{dv}^*, an anonymous credential cred_{dv^*}, a secret-public key-pair $(\mathsf{SK}_{dv^*}, \mathsf{PK}_{dv^*})$, a double-spending identifier $dsid$ and a balance value val^* as a forgery. Let x_{dv^*} denote SK_{dv^*}, $\mathbb{E}_{\mathcal{A}}$ denote the event that $\mathcal{A}$ wins the game. $\mathcal{C}$ retrieves $(\alpha_{dv^*}, \mathsf{PK}_{dv^*}, \mathsf{cred}_{dv^*})$ from $\mathcal{T}_1$ with ID_{dv}^* and aborts if such entry does not exist. If $\mathbb{E}_{\mathcal{A}}$ happens, the equation $\mathsf{PK}_{dv^*} = w^{x_{dv^*}} = A^{\alpha_{dv^*}} = w^{a \cdot \alpha_{dv^*}}$ holds. Then, $\mathcal{C}$ outputs $a = x_{dv^*}/\alpha_{dv^*}$ to break the DL assumption.

• *Probability Analysis:* Let $\mathbb{E}_{DL}$ be the event that $\mathcal{C}$ breaks the DL assumption, and $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game. $\epsilon' = Pr[\mathbb{E}_{DL}] = Pr[\mathbb{E}_{DL}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] + Pr[\mathbb{E}_{DL}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + Pr[\mathbb{E}_{DL}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] = 0 + \epsilon = \epsilon$.

Theorem 3: The anonymous credential in AnoPay is secure under message modification attacks.

Proof: The message modification attacks implemented on AnoPay fall into three types. (1) *Attack-1:* the adversary modifies the messages sent by $\mathcal{DV}$ in protocols Issue, TopUp, $\mathsf{Pre\text{-}Payment}$ and FeeDED. (2) *Attack-2:* the adversary modifies the messages sent by $\mathcal{CI}$ in protocols Issue, TopUp and FeeDED. (3) *Attack-3:* the adversary modifies the messages transmitted in protocols $\mathsf{Aggregate}$ and Trace. In the case of *Attack-1*. and *Attack-3*, the transmitted messages contain a zero-knowledge proof (e.g., $\Pi_{dv}^1, \Pi_{dv}^2, \Pi_{dv}^3, \Pi_{dv}^4$), which proves that the sender knows the secret key bound to his public key or anonymous credential, and the knowledge in other messages. The secret key is only held by the message sender and not transmitted through any channel. If any transmitted message is modified, the adversary cannot forge a valid zero-knowledge proof for the the modified message and the integrity is guaranteed. The integrity under *Attack-2* is ensured based on the unforgeability of PS signature [31]. In this case, the transmitted messages contain a tuple $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$, where $\hat{\sigma}_{dv,2} = \hat{\sigma}_{dv,1}^{x+y_1 \cdot x_{dv}+y_2 \cdot dsid+y_3 \cdot val+r}$ (r is a random element in $\mathbb{Z}_p^*$). The tuple $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ is essentially a PS signature signed on messages $(x_{dv}, dsid, val)$ by $\mathcal{CI}$. The adversary cannot forge a valid $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ on the modified messages. The verification of $(\hat{\sigma}_{dv,1}, \hat{\sigma}_{dv,2})$ will abort if any message sent by $\mathcal{CI}$ in protocols Issue, TopUp and FeeDED is modified.

B. Anonymity of Driver

Theorem 4: The proposed system is anonymous if the divisible decision Diffie-Hellman (DDDH) assumption holds in $\mathbb{G}_1$.

Proof: Assume there exists a PPT adversary $\mathcal{A}$ can break the anonymity of AnoPay with non-negligible advantage ϵ, we can construct a PPT challenger $\mathcal{C}$ to break the divisible decision Diffie-Hellman (DDDH) assumption with non-negligible advantage $\epsilon' = \frac{\epsilon}{2}$.

• *Setup and Keygen Phase:* The challenger $\mathcal{C}$ is given an instance of DDDH problem: $(w, A = w^a, B = w^b, \Gamma = w^c)$, where c equals to $\frac{a}{b}$ or a random value $r \in_R \mathbb{Z}_p^*$. $\mathcal{C}$ runs Setup and sets the system parameter $\mathsf{pp} = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, w, \widetilde{g})$, $\mathsf{PK}_{ar} = B$. $\mathcal{C}$ runs $\mathsf{CIKeygen}$, $\mathsf{PLKeygen}$ to generate $(\mathsf{PK}_{ci}, \mathsf{SK}_{ci}, \mathsf{PK}_{pl}, \mathsf{SK}_{pl})$, and sends $(\mathsf{pp}, \mathsf{PK}_{ci}, \mathsf{PK}_{pl}, \mathsf{PK}_{ar}, \mathsf{SK}_{ci}, \mathsf{SK}_{pl})$ to $\mathcal{A}$. Besides, $\mathcal{C}$ constructs a simulator $\mathcal{S}$ for the proof of knowledge system.

• *Query Phase:* The challenger $\mathcal{C}$ initializes two empty tables $\mathcal{T}_1$ and $\mathcal{T}_2$. The adversary $\mathcal{A}$ adaptively issues the following queries.

- *Credential Issue Query:* $\mathcal{A}$ submits an identity ID_{dv} and an initial balance val to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ selects $x_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $(\mathsf{SK}_{dv}, \mathsf{PK}_{dv}) = (x_{dv}, w^{x_{dv}})$. Then, $\mathcal{C}$ runs the protocol Issue with the adversary $\mathcal{A}$ to obtain $(\mathsf{cred}_{dv}, dsid)$. $\mathcal{C}$ inserts $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ to $\mathcal{T}_1$.

- *Parking Card Top-up Query:* $\mathcal{A}$ submits the driver's identity ID_{dv} and a incremental value inc to $\mathcal{C}$ as a query. $\mathcal{C}$ retrieves $(\mathsf{cred}_{dv}, dsid)$ from list and runs the $\mathsf{TopUp}(\mathcal{CI}(\mathsf{SK}_{ci}, \mathsf{PK}_{ci}, \mathsf{pp}) \leftrightarrow \mathcal{DV}(\mathsf{SK}_{dv}, \mathsf{cred}_{dv}, dsid, val, inc, \mathsf{pp}))$ with adversary $\mathcal{A}$. If ID_{dv} is not stored in $\mathcal{C}$'s list, $\mathcal{C}$ aborts.

$\mathcal{A}$ submits an identity ID_{dv} and a incremental value inc to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves the entry $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$ and runs TopUp to obtain $(\mathsf{cred}_{dv}^*, val^*)$. Then, the challenger updates the entry with $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}^*, dsid, val^*)$.

- *Driver Parking and Pre-Payment Query:* $\mathcal{A}$ submits an identity ID_{dv} and a parking charge $chrg$ to $\mathcal{C}$ as a query. If the entry specified by ID_{dv} does not exist in $\mathcal{T}_1$, $\mathcal{C}$ aborts. Otherwise, $\mathcal{C}$ retrieves the entry $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$. $\mathcal{C}$ selects $\xi_{dv}, \zeta_{dv}, c_{dv}, d_{dv}, e_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\mathsf{cred}'_{dv} = (\sigma_{dv,1}^{\xi_{dv}}, (\sigma_{dv,2} \cdot \sigma_{dv,1}^{\zeta_{dv}})^{\xi_{dv}})$, $T_{dv} = w^{dsid}$, $E_{dv} = (Y_{pl}^{c_{dv}}, Z^{d_{dv}}, B^{e_{dv}})$, $C_{dv} = (w^{c_{dv}+d_{dv}} \cdot g^{chrg}, w^{d_{dv}+e_{dv}} \cdot B^{x_{dv}})$, and generates a corresponding proof Π_{dv}^3. Then, $\mathcal{C}$ sends the tuple $(\mathsf{cred}'_{dv}, T_{dv}, E_{dv}, C_{dv}, val, chrg, \Pi_{dv}^3)$ to $\mathcal{A}$ and inserts $(T_{dv}, E_{dv}, C_{dv}, ID_{dv}, chrg)$ into $\mathcal{T}_2$.

- *Parking Card Fee Deduction Query:* $\mathcal{A}$ submits a transaction identifier T_{dv} as a query. If the entry specified by T_{dv} does not exist in $\mathcal{T}_2$, $\mathcal{C}$ aborts. Otherwise, the challenger $\mathcal{C}$ retrieves the entry $(T_{dv}, E_{dv}, C_{dv}, ID_{dv}, chrg)$ from $\mathcal{T}_2$ and retrieves the corresponding entry $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$ from $\mathcal{T}_1$. $\mathcal{C}$ runs FeeDED to obtain $(\mathsf{cred}^*_{dv}, dsid^*, val^*)$. After the interaction, $\mathcal{C}$ updates the entry specified by ID_{dv} in $\mathcal{T}_1$ to $(ID_{dv}, \mathsf{SK}_{dv}, \mathsf{PK}_{dv}, \mathsf{cred}^*_{dv}, dsid^*, val^*)$.

- *Trace Query:* $\mathcal{A}$ submits a transaction identifier T_{dv} as a query. If the entry specified by T_{dv} does not exist in $\mathcal{T}_2$, $\mathcal{C}$ aborts. Otherwise, the challenger $\mathcal{C}$ retrieves the entry $(T_{dv}, E_{dv}, C_{dv}, ID_{dv}, chrg)$ from $\mathcal{T}_2$. $\mathcal{C}$ returns ID_{dv} to the adversary $\mathcal{A}$.

• *Challenge:* $\mathcal{A}$ selects two identities $(ID^*_{dv_1}, ID^*_{dv_2})$ and a parking charge $chrg^*$. $\mathcal{C}$ flips a coin to choose $b \in \{0,1\}$ and searches the entry $(ID^*_{dv_b}, \mathsf{SK}^*_{dv_b}, \mathsf{PK}^*_{dv_b}, \mathsf{cred}^*_{dv_b}, dsid^*_b, val^*_b)$ from $\mathcal{T}_1$. The challenger $\mathcal{C}$ selects $\xi^*_{dv}, \zeta_{dv}{}^*, \eta^*_{dv}, c^*_{dv}, d^*_{dv} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\mathsf{cred}'^*_{dv} = (\sigma_{dv,1}^{\xi^*_{dv}}, (\sigma_{dv,2} \cdot \sigma_{dv,1}^{\zeta^*_{dv}})^{\xi^*_{dv}})$, $T^*_{dv} = w^{dsid^*_b}$, $E^*_{dv} = (Y_{pl}^{c^*_{dv}}, Z^{d^*_{dv}}, A)$, $C^*_{dv} = (w^{c^*_{dv}+d^*_{dv}} \cdot g^{chrg^*}, w^{d^*_{dv}} \cdot \Gamma \cdot B^{x^*_{dv_b}})$. $\mathcal{C}$ runs the simulator $\mathcal{S}$ to generate the proof Π^{3*}_{dv} and sends the tuple $(\mathsf{cred}'^*_{dv}, T^*_{dv}, E^*_{dv}, C^*_{dv}, val^*, chrg^*, \Pi^{3*}_{dv})$ to $\mathcal{A}$.

• *Query Phase 2:* The adversary $\mathcal{A}$ adaptively issues the queries as the query phase 1. If the submitted $ID_{dv} \notin \{ID^*_{dv_1}, ID^*_{dv_2}\}$, the challenger $\mathcal{C}$ aborts.

• *Guess:* $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the game if $b = b'$.

• *Probability Analysis:* Let $\mathbb{E}_{\mathcal{A}}$ be the event that $\mathcal{A}$ wins the game, ϵ_{DDDH} be the event that $\mathcal{C}$ distinguishes the DDDH quadruple, $\mathbb{E}_D$ be the case that $T = (w, A = w^a, B = w^b, \Gamma = w^c)$ is a DDDH quadruple, $\mathbb{E}_R$ be the case that T is a random quadruple. As we analyzed before, only the element $C^*_{dv,2} = w^{d^*_{dv}} \cdot \Gamma \cdot B^{x^*_{dv_b}}$ contains the secret key $x^*_{dv_b}$.

If Γ is an random element in $\mathbb{G}_1$, the component $C^*_{dv,2} = w^{d^*_{dv}} \cdot \Gamma \cdot B^{x^*_{dv_b}}$ is random and the probability for the adversary $\mathcal{A}$ to win this game is $\frac{1}{2}$ (i.e., $Pr[\mathbb{E}_{\mathcal{A}}|\mathbb{E}_R] = \frac{1}{2}$), otherwise $\mathbb{E}_{\mathcal{A}}$ happens with the probability $\epsilon + \frac{1}{2}$ (i.e., $Pr[\mathbb{E}_{\mathcal{A}}|\mathbb{E}_D] = \epsilon + \frac{1}{2}$). When $\mathbb{E}_{\mathcal{A}}$ happens, $\mathcal{C}$ always outputs 1 to denote that T is a DDDH quadruple. Thus, $Pr[\epsilon_{DDDH}|\mathbb{E}_{\mathcal{A}} \wedge \mathbb{E}_D] = 1$, $Pr[\epsilon_{DDDH}|\mathbb{E}_{\mathcal{A}} \wedge \mathbb{E}_R] = 0$, $Pr[\epsilon_{DDDH}|\overline{\mathbb{E}}_{\mathcal{A}} \wedge \mathbb{E}_D] = 0$, $Pr[\epsilon_{DDDH}|\overline{\mathbb{E}}_{\mathcal{A}} \wedge \mathbb{E}_R] = 1$.

$$
\begin{aligned}
\epsilon' &= |Pr[\epsilon_{DDDH}] - \frac{1}{2}| \\
&= |Pr[\epsilon_{DDDH}|\mathbb{E}_{\mathcal{A}}] \cdot Pr[\mathbb{E}_{\mathcal{A}}] + \\
&\quad Pr[\epsilon_{DDDH}|\overline{\mathbb{E}}_{\mathcal{A}}] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}] - \frac{1}{2}| \\
&= |Pr[\epsilon_{DDDH}|\mathbb{E}_{\mathcal{A}} \wedge \mathbb{E}_D] \cdot Pr[\mathbb{E}_{\mathcal{A}}|\mathbb{E}_D] \cdot Pr[\mathbb{E}_D] \\
&\quad + Pr[\epsilon_{DDDH}|\mathbb{E}_{\mathcal{A}} \wedge \mathbb{E}_R] \cdot Pr[\mathbb{E}_{\mathcal{A}}|\mathbb{E}_R] \cdot Pr[\mathbb{E}_R] \\
&\quad + Pr[\epsilon_{DDDH}|\overline{\mathbb{E}}_{\mathcal{A}} \wedge \mathbb{E}_D] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}|\mathbb{E}_D] \cdot Pr[\mathbb{E}_D] \\
&\quad + Pr[\epsilon_{DDDH}|\overline{\mathbb{E}}_{\mathcal{A}} \wedge \mathbb{E}_R] \cdot Pr[\overline{\mathbb{E}}_{\mathcal{A}}|\mathbb{E}_R] \cdot Pr[\mathbb{E}_R] - \frac{1}{2}| \\
&= |1 \cdot (\epsilon + \frac{1}{2}) \cdot \frac{1}{2} + 0 + 0 + 1 \cdot \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2}| = \frac{\epsilon}{2}.
\end{aligned}
$$

VI. Performance Analysis

A. Theoretical Analysis and Comparison

In this subsection, we compare AnoPay with pay-by-phone parking schemes [4], [6], [7], blockchain-based parking payment schemes [8], [9], [10], [11], anonymous e-cashes [30], [39], [40] and the vehicle parking schemes with anonymous authentication [5], [16], [17], [18], [19], [20] in terms of functionality, computation costs and storage overhead.

Table II compares AnoPay with the related works in terms of functionality.

• *Anonymous payment:* Is an essential property in privacy-preserving parking systems, which provides a reliable guarantee for the driver's identity privacy. Among these works, the schemes in [16], [17], [18], [19], [20] do not support anonymous payment, while the others and AnoPay possess this essential property.

• *Driver authentication:* The pay-by-phone parking schemes [4], [6], [7] and the schemes with anonymous e-cashes [30], [39], [40] fail to support identity authentication for drivers.

• *Expenses hiding:* The parking fee recorded in each parking receipt should be kept confidential. The payment schemes in [8], [9], [10] do not support parking expenses hiding. The schemes in [16], [17], [18], [19], [20] are not relevant because they do not provide payment functionality.

• *Identity disclosure:* Allows a trusted entity to reveal the identity of a driver who has generated a problematic parking receipt. This function is the basic requirement of accountability and essential for detecting double-spending and tracing malicious drivers. The schemes in [4], [6], [7], [11], [16], [20] fail to support this property. The schemes in [30], [39], [40] only support tracking of double-spending users but do not support tracking of malicious users in other cases.

• *Double-spending resistance:* Is an essential requirement for any payment system. Unfortunately, the schemes in [16], [17], [18], [19], [20] fail to ensure double-spending resistance.

• *Unlinkability:* Guarantees that different parking receipts generated by the same driver cannot be linked by any adversary; this ensures that no adversary can track any driver's moving trajectory or infer his real identity. All except schemes in [8], [9], [10] achieve unlinkability.

• *Credential update:* A parking credential is parameterized with a vector of attributes (e.g., driving license number, driving years, vehicle type, balance) and when authenticating, drivers can prove possession of a parking credential that fulfills a certain

TABLE II
FUNCTION COMPARISON

Scheme	Anonymous Payment	Driver Authentication	Expenses Hiding	Identity Disclosing	Double-spending Resistance	Unlinkability	Credential Update
[4]	√	⊥	√	⊥	√	√	⊥
[6]	√	⊥	√	⊥	√	√	⊥
[7]	√	⊥	√	⊥	√	√	⊥
[8]	√	√	⊥	√	√	⊥	⊥
[9]	√	√	⊥	√	√	⊥	⊥
[10]	√	√	⊥	√	√	⊥	⊥
[11]	√	√	√	⊥	√	√	⊥
[39]	√	⊥	√	DS only	√	√	⊥
[30]	√	⊥	√	DS only	√	√	⊥
[40]	√	⊥	√	DS only	√	√	⊥
[5]	√	√	√	√	√	√	⊥
[17]	⊥	√	⊥	√	⊥	√	⊥
[16]	⊥	√	⊥	⊥	⊥	√	⊥
[18]	⊥	√	⊥	√	⊥	√	⊥
[19]	⊥	√	⊥	√	⊥	√	⊥
[20]	⊥	√	⊥	⊥	⊥	√	⊥
AnoPay	√	√	√	√	√	√	√

√: supported function ⊥: unsupported function DS only: supported in case of double-spending

policy (e.g., "balance $\geq$ 10 dollar") without revealing anything about the attributes except that they satisfy the policy. Credential update allows users to directly update attributes instead of requesting a new credential. AnoPay supports the update of parking card balance and can be extended to update other credential attributes. During the update procedure, the credential issuer knows the update function without learning the driver's previous attribute. Therefore, the update process preserves anonymity of the driver, who can anonymously top up the parking card or make a fee deduction. The comparison in Table II indicates that this feature is exclusive for AnoPay.

The comparison in Table II indicates that AnoPay is versatile in functionality and security.

Table III compares the computation cost of the algorithms Top-up, Pay and Trace in different schemes. Let t_e be the time cost of an exponential calculation and t_p be the time cost of a pairing calculation. Let Top-up, Pay and Trace denote the computation costs in parking card top-up, parking fee payment and identity disclosing phases, respectively. In AnoPay, Top-up and Trace correspond to TopUp and Trace protocols, respectively; the cost of Pay is the sum of computation overheads in Pre-Payment and FeeDED protocols.

• Denote k_1 as the top-up amount. The computation complexities of Top-up in the schemes [4], [5], [6], [7] are $O(3k_1)t_e$, $O(3k_1)t_e$, $O(4k_1)t_e$, $O(4k_1)t_e$, respectively. The top-up overheads in schemes [4], [5], [6], [7] increase linearly with the top-up amount k_1, which causes a high delay for relatively high-value payment. The schemes in [8], [9], [11], [16], [17], [18] provide no top-up construction. The computational complexities in the schemes [30], [39], [40] and AnoPay are constant.

• Denote k_2 as the amount of parking fee. The computation complexities of Pay in the schemes [4], [5], [6], [7], [11], [30], [40] grow with the payment amount k_2, which are $O(3k_2)t_e$, $O(4k_2)t_e$, $O(5k_2)t_e$, $O(14k_2)t_e$, $O(n_g \cdot k_2)t_e$, $O(12k_2)t_e$+$O(1)t_p$, $O(6k_2)t_e$+$O(1)t_p$ respectively. The payments in schemes [8], [9] are conducted on blockchains with smart contracts, which cannot be evaluated using t_e and t_p for smart contracts. The schemes in [16], [17], [18] do not support parking payment. The computation overheads of Pay in [39] and AnoPay are constant.

• Denote n as the number of registered users. The computation complexities of tracing user's identity in [5], [8], [17], [40] are $O(n)t_p$, $O(3n)t_p$, $O(n)t_p$, $O(2n)t_p$, respectively. The tracing complexities in the schemes [18], [39] are $O(2n)t_e$+$O(n)t_p$. These schemes execute large pairing operations to test each registered user until the malicious user is found. Therefore, their tracing algorithms are inefficient and incur high computation overheads in [5], [8], [17], [18], [39], [40]. The schemes in [4],

TABLE III
COMPUTATION OVERHEAD COMPARISON

Scheme	Top-up	Pay	Trace
[4]	$O(3k_1) \cdot t_e$	$O(3k_2) \cdot t_e$	⊥
[6]	$O(4k_1) \cdot t_e$	$O(5k_2) \cdot t_e$	⊥
[7]	$O(4k_1) \cdot t_e$	$O(14k_2) \cdot t_e$	⊥
[8]	⊥	$*$	$O(3n) \cdot t_p$
[9]	⊥	$*$	$O(1) \cdot t_e$
[11]	⊥	$O(n_g \cdot k_2) \cdot t_e$	⊥
[39]	$O(1) \cdot t_e$+ $O(1) \cdot t_p$	$O(1) \cdot t_e$+ $O(1) \cdot t_p$	$O(2n)t_e$+ $O(n)t_p$
[30]	$O(1) \cdot t_e$	$O(12k_2) \cdot t_e$+ $O(1) \cdot t_p$	$O(1) \cdot t_e$
[40]	$O(1) \cdot t_e$	$O(6k_2) \cdot t_e$+ $O(1) \cdot t_p$	$O(2n) \cdot t_p$
[5]	$O(3k_1) \cdot t_e$	$O(4k_2) \cdot t_e$	$O(n) \cdot t_p$
[17]	⊥	⊥	$O(n) \cdot t_p$
[16]	⊥	⊥	⊥
[18]	⊥	⊥	$O(2n) \cdot t_e$+$O(n) \cdot t_p$
AnoPay	$O(1) \cdot t_e$+ $O(1) \cdot t_p$	$O(1) \cdot t_e$+ $O(1) \cdot t_p$	$O(1) \cdot t_e$

t_e: exponential operation time; t_p: pairing operation time;
n: number of registered users; k_1/k_2: top-up/payment amount;
⊥: function unsupported; $*$: pay by blockchain;
n_g: number of account group in [11];

TABLE IV
STORAGE OVERHEAD COMPARISON

Scheme	$\lvert\mathcal{DV}\rvert$	$\lvert\mathcal{CI}\rvert$	$\lvert\mathcal{PL}\rvert$
[4]	$O(3k_0)$	$O(m)$	$\perp$
[6]	$O(3k_0)$	$O(m)$	$\perp$
[7]	$O(3k_0)$	$O(m)$	$\perp$
[8]	$O(1)$	$O(4n)$	$O(3l)$
[9]	$O(1)$	$O(3n)$	$\perp$
[11]	$O(1)$	$\perp$	$\perp$
[39]	$O(1)$	$O(3m)$	$O(4l)$
[30]	$O(1)$	$O(m)$	$O(5m)$
[40]	$O(1)$	$O(3m)$	$O(l+4m)$
[5]	$O(2k_0)$	$O(4n+m)$	$O(2l+2m)$
[17]	$O(1)$	$O(3n)$	$O(4l)$
[16]	$O(1)$	$O(l)$	$O(l)$
[18]	$O(1)$	$O(7n+9l)$	$\perp$
AnoPay	$O(1)$	$O(2n+6l)$	$O(3l)$

n: number of registered users; m: total parking payment;
k_0: $\mathcal{DV}$'s parking card balance; l: number of parking transactions;

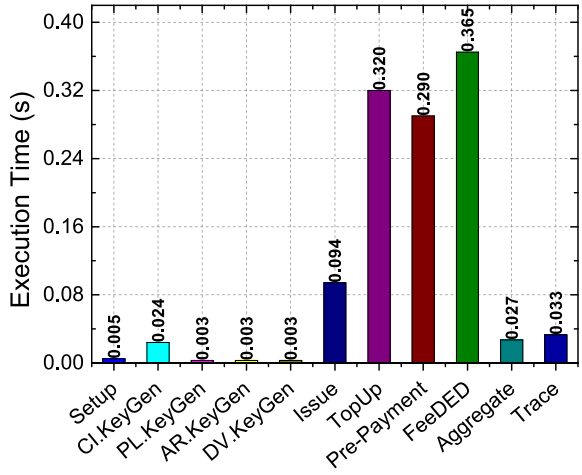


Fig. 7. Execution time of AnoPay.

[6], [7], [11], [16] do not support accountability. The computation overheads for tracing in [9], [30] and AnoPay are $O(1)t_e$.

Table III demonstrates that AnoPay achieves constant computation overheads in top-up, pay and trace phases.

Table IV compares the storage overhead of AnoPay with other schemes. The notations $|\mathcal{DV}|$, $|\mathcal{CI}|$ and $|\mathcal{PL}|$ denote the storage overhead of the entities $\mathcal{DV}$, $\mathcal{CI}$ and $\mathcal{PL}$, respectively. In AnoPay, the public parameter pp and the public keys $(\mathsf{PK}_{ci}, \mathsf{PK}_{pl}, \mathsf{PK}_{ar})$ are contained in each entity's storage by default. $\mathcal{DV}$ stores the tuple $(\mathsf{PK}_{dv}, \mathsf{SK}_{dv}, \mathsf{cred}_{dv}, dsid, val)$. $\mathcal{CI}$ stores its secret key SK_{ci}, the tuple $(\mathsf{PK}_{dv}, ID_{dv})$ for each registered $\mathcal{DV}$, and the pre-payment receipt (T_{dv}, E_{dv}, C_{dv}) for each parking transaction. $\mathcal{PL}$ stores its secret key SK_{pl}, the parking charges $\{chrg_i\}_{i\in I}$ and the pre-payment receipts $\{T_{dv_i}, E_{dv_i,1}\}_{i\in I}$. For the blockchain-based schemes [8], [9], [11], we only consider the private components that are not stored in blockchain.

• Denote k_0 as the balance in $\mathcal{DV}$'s parking card. The spatial complexity of $\mathcal{DV}$ in schemes [4], [5], [6], [7] are $O(3k_0)$, $O(2k_0)$, $O(3k_0)$, $O(3k_0)$, respectively. The storage overhead of $\mathcal{DV}$ in the other schemes including AnoPay are constant.

• Let n be the number of registered users, m be the total parking payment, l be the number of parking transactions (which equals to the number of pre-payment receipts in AnoPay). The spatial complexities of $|\mathcal{CI}|$ in the schemes [4], [5], [6], [7], [30], [39], [40] grow with the total payment m, while in the schemes [5], [8], [9], [17] and AnoPay they grow with the number of registered users n. On the other hand, the costs of $|\mathcal{CI}|$ in the schemes [16], [18] and AnoPay grow with l. Note that m is much greater than l in general.

• The spatial complexities of $\mathcal{PL}$ in the schemes [5], [8], [16], [17], [30], [39], [40] and AnoPay are $O(2l+2\,m)$, $O(3\,l)$, $O(l)$, $O(4\,l)$, $O(5\,m)$, $O(4\,l)$, $O(l+4\,m)$, $O(3\,l)$, respectively. The storage sizes in the schemes [5], [30], [40] linearly increase with the payment amount m, which have large storage overheads. The storage overhead of $|\mathcal{PL}|$ in AnoPay is close to that in the schemes [8], [16], [17], [39].

The comparison in Table IV shows that AnoPay has advantage in storage overheads for $\mathcal{DV}$, $\mathcal{CI}$ and $\mathcal{PL}$.

Table V presents the computation costs and communication costs of the algorithms Setup, CI.KeyGen, PL.KeyGen, AR.KeyGen, DV.KeyGen, Issue, TopUp, Pre-Payment, FeeDED, Aggregate, Trace and zero-knowledge proofs Π^1_{dv}, Π^2_{dv}, Π^3_{dv}, Π^4_{dv}, Π^1_{pl}, Π^1_{ci}, Π^2_{ci}, Π^1_{ar}, respectively. Denote $t_{e_1}, t_{e_2}, t_{e_T}$ as the time cost of an exponential calculation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$; t_p the time cost of a pairing calculation; $|\mathbb{Z}_p|, |\mathbb{G}_1|, |\mathbb{G}_2|, |\mathbb{G}_T|$ the size of an element in $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively. Let l be the number of pre-payment receipts stored by $\mathcal{PL}$. The computation costs of Issue, TopUp, Pre-Payment, FeeDED for $\mathcal{DV}$ are $4t_{e_1}+3t_{e_2}+2t_p$, $4t_{e_1}+3t_{e_2}+2t_p$, $10t_{e_1}$, $8t_{e_1}+3t_{e_2}+2t_p$, respectively. The computation costs of Issue, TopUp, FeeDED for $\mathcal{CI}$ are $4t_{e_1}$, $3t_{e_1}$ and $4t_{e_1}$, respectively. The zero knowledge proofs Π^3_{dv} and Π^4_{dv} in pre-payment and fee deduction phases consume a bit more calculations: Π^3_{dv} requires $9t_{e_1}+4t_{e_2}+3t_p+t_h$, $16t_{e_1}+4t_{e_2}+t_{e_T}+3t_p+t_h$ for $\mathcal{DV}$, $\mathcal{CI}$, respectively, and Π^4_{dv} costs $4t_{e_1}+4t_{e_2}+3t_p+t_h$, $13t_{e_1}+4t_{e_2}+t_{e_T}+3t_p+t_h$ for proof generation and verification, respectively.

In general, Table V demonstrates that the computation and communication overheads of most algorithms in AnoPay achieve constant complexity.

B. Experimental Analysis

We implemented AnoPay on a laptop with an Intel(R) Core(TM) i7-6700HQ CPU, 16 GB RAM, 512 GB SSD running Windows 10 operating system. The experiments were conducted using MIRACL library and Type-III pairing on BN curve $E: y^2 = x^3 + 2$ over finite field F_q. The Type-III pairing was instantiated using Tate pairing with embedding degree 12, and SHA-256 was selected as the collision-resistant hash function. To simulate the network situation and propagation delay, we use the network simulation library OMNeT++ 5.8 and SUMO 1.8.0. The simulation parameters are listed in Table VI.

(1) Fig. 7 shows the computation costs of diverse algorithms in AnoPay. Fig. 8 gives out the communication overheads of

TABLE V
COMPUTATION AND COMMUNICATION OVERHEAD OF ANOPAY

Algorithms	Setup	CI.KeyGen	PL.KeyGen	AR.KeyGen	DV.KeyGen
	$\mathcal{CI}$	$\mathcal{CI}$	$\mathcal{PL}$	$\mathcal{AR}$	$\mathcal{DV}$
Computation Cost	$\perp$	$5t_{e_1} + 4t_{e_2}$	t_{e_1}	t_{e_1}	t_{e_1}
Communication Cost	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\|$	$5\|\mathbb{Z}_p\| + 5\|\mathbb{G}_1\| + 4\|\mathbb{G}_2\|$	$\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + \|\mathbb{G}_1\|$

Algorithms	Issue		TopUp		Pre-Payment	
	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{DV}$	$\mathcal{PL}$
Computation Cost	$4t_{e_1} + 3t_{e_2} + 2t_p$	$4t_{e_1}$	$4t_{e_1} + 3t_{e_2} + 2t_p$	$3t_{e_1}$	$10t_{e_1}$	$\perp$
Communication Cost	$\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$2\|\mathbb{Z}_p\| + 8\|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + 6\|\mathbb{G}_1\|$

Algorithms	FeeDED		Aggregate		Trace	
	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{PL}$	$\mathcal{CI}$	$\mathcal{CI}$	$\mathcal{AR}$
Computation Cost	$8t_{e_1} + 3t_{e_2} + 2t_p$	$4t_{e_1}$	t_{e_1}	$2t_{e_1}$	t_{e_1}	$2t_{e_1}$
Communication Cost	$4\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_p\| + (l+1)\|\mathbb{G}_1\|$	$\perp$	$3\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$

Algorithms	$\mathbf{\Pi^1_{dv}}$		$\mathbf{\Pi^2_{dv}}$		$\mathbf{\Pi^3_{dv}}$	
	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{DV}$	$\mathcal{PL}$
Computation Cost	$4t_{e_1} + t_h$	$6t_{e_1} + t_h$	$4t_{e_2} + 3t_p + t_h$	$4t_{e_2} + t_{e_T} + 3t_p + t_h$	$9t_{e_1} + 4t_{e_2} + 3t_p + t_h$	$16t_{e_1} + 4t_{e_2} + t_{e_T} + 3t_p + t_h$
Communication Cost	$3\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$\perp$	$5\|\mathbb{Z}_p\| + 2\|\mathbb{G}_T\|$	$\perp$	$8\|\mathbb{Z}_p\| + 7\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\|$	$\perp$

Algorithms	$\mathbf{\Pi^4_{dv}}$		$\mathbf{\Pi^1_{pl}}$		$\mathbf{\Pi^1_{ci}}$	
	$\mathcal{DV}$	$\mathcal{CI}$	$\mathcal{PL}$	$\mathcal{CI}$	$\mathcal{CI}$	$\mathcal{PL}$
Computation Cost	$4t_{e_1} + 4t_{e_2} + 3t_p + t_h$	$13t_{e_1} + 4t_{e_2} + t_{e_T} + 3t_p + t_h$	$2t_{e_1} + t_h$	$4t_{e_1} + t_h$	$2t_{e_1} + t_h$	$4t_{e_1} + t_h$
Output size	$10\|\mathbb{Z}_p\| + 4\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\|$	$\perp$	$3\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$\perp$	$2\|\mathbb{Z}_p\| + 3\|\mathbb{G}_1\|$	$\perp$

Algorithms	$\mathbf{\Pi^2_{ci}}$		$\mathbf{\Pi^1_{ar}}$			
	$\mathcal{CI}$	$\mathcal{AR}$	$\mathcal{AR}$	$\mathcal{CI}$		
Computation Cost	$2t_{e_1} + t_h$	$4t_{e_1} + t_h$	$3t_{e_1} + t_h$	$6t_{e_1} + t_h$		
Output size	$2\|\mathbb{Z}_p\| + 2\|\mathbb{G}_1\|$	$\perp$	$2\|\mathbb{Z}_p\| + 4\|\mathbb{G}_1\|$	$\perp$		

$t_{e_1}, t_{e_2}, t_{e_T}$: time cost of an exponential calculation in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$
$|\mathbb{Z}_p|, |\mathbb{G}_1|, |\mathbb{G}_2|, |\mathbb{G}_T|$: size of an element in $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$
t_p: time cost of a pairing calculation
l: number of pre-payment receipts

TABLE VI
SIMULATION PARAMETERS

Parameter	Value
Simulation area	2500 m * 2500 m
Simulation time	200 s
Maximal interface distance	2600 m
PHY layer standard	IEEE 802.11p
MAC layer standard	IEEE 1609.4
Data rate	6 Mbps
Transmission power	20 mW
Noise floor	-98 dBm
Sensitivity	-110 dBm
Maximum vehicle velocity	14 m/s

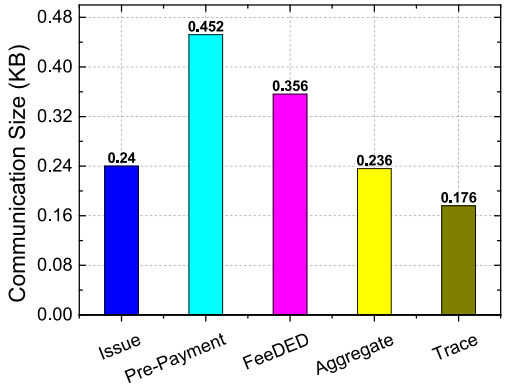


Fig. 8. Communication overhead.

TopUp, Pre-Payment, FeeDED, Aggregate and Trace algorithms. Fig. 9 shows the storage sizes of the entities $\mathcal{DV}$, $\mathcal{CI}$ and $\mathcal{PL}$, respectively.

• Fig. 7 shows that Issue, TopUp, Pre-Payment and FeeDED require relatively more calculations in AnoPay, and their execution times are 0.094 s, 0.320 s, 0.290 s and 0.365 s, respectively. The generation of $\mathcal{CI}$'s secret-public key pair requires 0.024 s. The execution to aggregate 10 pre-payment receipts consumes 0.027 s, and tracing a pre-payment receipt consumes 0.033 s. The other algorithms are lightweight and consume less than 0.05 s.

• As shown in Fig. 8, the communication overheads of TopUp, Pre-Payment and FeeDED are 0.24 KB, 0.452 KB and 0.356 KB, respectively. It takes 0.236 KB to execute Aggregate with 10 pre-payment receipts, which also slightly increases with

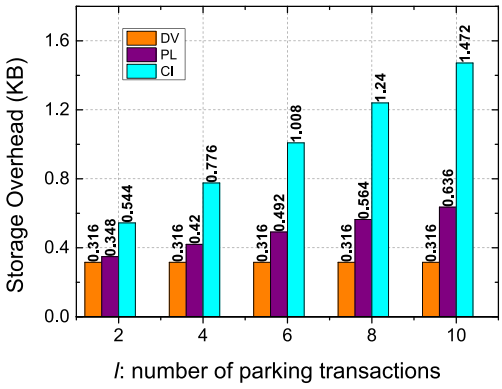


Fig. 9. Storage overhead.

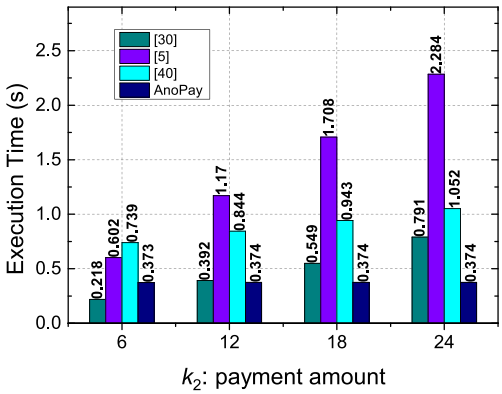


Fig. 10. Payment time comparison.

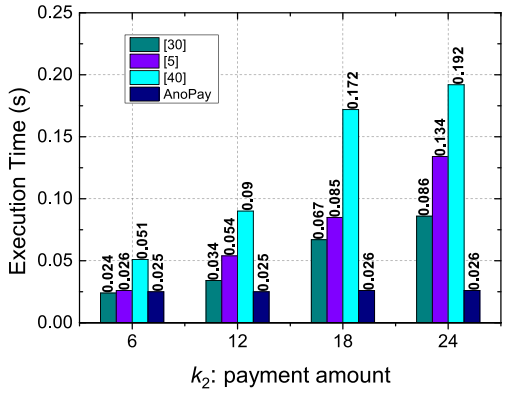


Fig. 11. Aggregate time comparison.

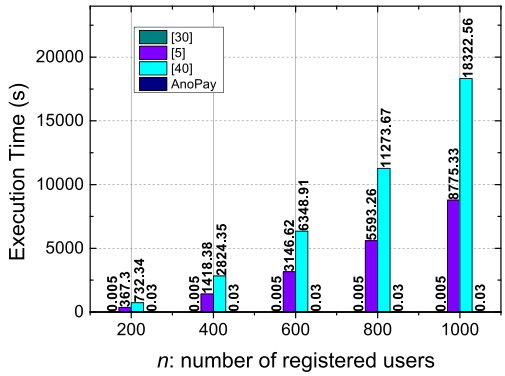


Fig. 12. Trace time comparison.

the number of transactions. The communication overhead for executing the algorithm Trace is 0.176 KB.

• In Fig. 9, the storage overhead of $\mathcal{DV}$ is constant, and those of $\mathcal{PL}$ and $\mathcal{CI}$ increase linearly with the number of parking transactions l. It costs $\mathcal{DV}$ 0.318 KB to store the anonymous credential and secret-public key pair. When $l = 1$, the storage overheads of $\mathcal{CI}$ and $\mathcal{PL}$ are 0.428 KB and 0.312 KB, respectively. When $l = 5$, we have $|\mathcal{CI}| = 0.892$ KB and $|\mathcal{PL}| = 0.456$ KB.

(2) In the following, we compare the efficiency of AnoPay with the schemes in [5], [30], [40] since they achieve better performance among the related works and have similar functionalities as AnoPay. Fig. 10, Figs. 11 and 12 present various computation overheads of FeeDED, Aggregate and Trace in [5], [30], [40] and AnoPay.

• As shown in Fig. 10, the payment overheads in [5], [30], [40] and AnoPay are 1.17 s, 0.392 s, 0.844 s and 0.374 s respectively when $k_2 = 12$, and AnoPay has the best payment performance when $k_2 \geq 12$. AnoPay achieves efficient parking payment by leveraging updatable credential and ZKP technology.

• Fig. 11 shows that the time costs of schemes [5], [30], [40] vary with k_2. They cost at least 0.024 s for $k_2 = 6$, and 0.086 s for $k_2 = 24$, while AnoPay costs 0.025 s for $k_2 = 6$, and 0.026 s for $k_2 = 24$. AnoPay costs 15 t_e to run Aggregate, which is constant and does not vary with the payment amount k_2. When $k_2 > 6$, AnoPay has the best efficiency among the compared systems.

• Fig. 12 indicates that the scheme of [30] and AnoPay achieve better tracing efficiency, while the time costs for the trace algorithms in [5], [40] are much higher. When the number of registered users increases to $n = 200$, it costs 367.3 s, 0.005 s, 732.34 s to run the trace algorithms in [5], [30], [40], respectively, while AnoPay takes 0.003 s to run its trace algorithm. The Trace algorithm of AnoPay avoids heavy pairing calculation, which remains efficient even though the number of users increases.

(3) The storage overheads of $\mathcal{CI}$, $\mathcal{PL}$ are compared in Figs. 13 and 14, respectively. Let k_2/l denote the average parking payment in each transaction. We perform the experiments in Fig. 13, Fig. 14 when setting the number of transactions $l = 100$ and $k_2/l = 6, 12, 18, 24$.

• Fig. 13 shows the storage overheads of $\mathcal{CI}$ in our comparison. In AnoPay, $\mathcal{CI}$ stores (T_{dv}, E_{dv}, C_{dv}) for each parking transaction, which is independent to the payment amount in the transaction. As shown in Fig. 13, $\mathcal{CI}$ costs 11.912 KB to store 100 transaction receipts, which is constant when the average parking payment k_2/l varies. When $k_2/l \geq 12$, the storage overhead of AnoPay is the lowest among the compared schemes.

• Fig. 14 presents the storage overheads of $\mathcal{PL}$ in our comparison. The storage overheads of $\mathcal{PL}$ in [5], [30], [40] are at least 22.4 KB for $k_2/l = 6$, and 80 KB for $k_2/l = 24$. In AnoPay,

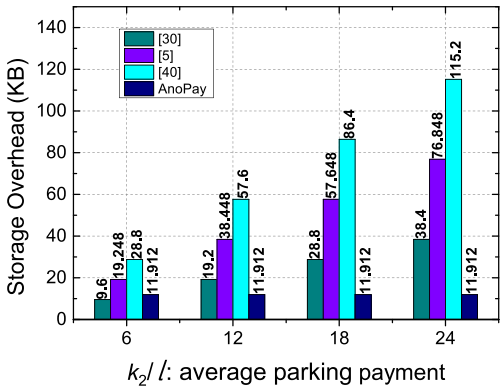


Fig. 13. $\mathcal{CI}$ storage comparison.

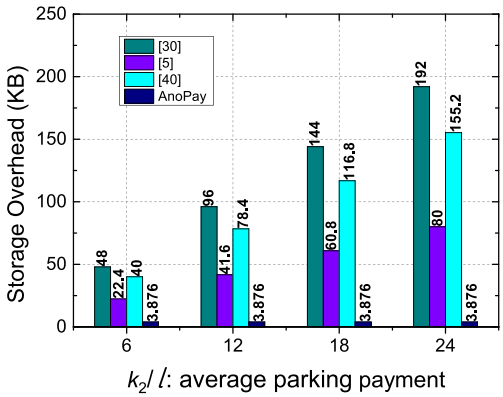


Fig. 14. $\mathcal{PL}$ storage comparison.

$\mathcal{PL}$ is required to store $(chrg, T_{dv}, E_{dv,1})$ only for each parking transaction, which is used for aggregating the parking fee. The storage overheads are both 3.876 KB when setting $k_2/l = 6, 24$, which is much lower than those in schemes [5], [30], [40].

VII. Conclusion and Future Work

In this paper, we investigated the anonymous payment problem for vehicle parking and presented an efficient anonymous parking payment system (AnoPay). AnoPay achieved constant payment overhead and efficient transaction aggregation, where updatable attribute-based anonymous credential, zero-knowledge proof and linear homomorphic encryption techniques were introduced to ensure the anonymity, unforgeability and accountability. The security models of AnoPay were formally defined, and the security properties of AnoPay were formally proved. Extensive experiments were conducted for the proposed system and related works, showing that AnoPay was significantly more efficient compared with existing solutions. In our future work, we will design new anonymous payment scheme based on post-quantum cryptography to resist quantum attacks. Quantum-resistant systems are able to resist the quantum computing related attacks, but also face the issues of large key size and high computation overheads. We will deal with theses challenges and design new schemes for vehicle parking to balance the efficiency and security.

References

[1] T. Lin, H. Rivano, and F. Le Mouël, "A survey of smart parking solutions," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3229–3253, Dec. 2017.

[2] "Smart parking market report 2019–2023," 2018. [Online]. Available: https://iot-analytics.com/product/smart-parking-market-report-2019--2023/

[3] "Vermonter sues a leading parking app after data breach," 2021. [Online]. Available: https://vtdigger.org/2021/06/10/vermonter-sues-a-leading-parking-app-after-data-breach/

[4] R. Garra, S. Martínez, and F. Sebé, "A privacy-preserving pay-by-phone parking system," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 5697–5706, Jul. 2017.

[5] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "ASAP: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 4, pp. 703–715, Jul./Aug. 2018.

[6] R. Borges and F. Sebé, "Parking tickets for privacy-preserving pay-by-phone parking," in *Proc. 18th ACM Workshop Privacy Electron. Soc.*, 2019, pp. 130–134.

[7] R. Borges and F. Sebé, "An efficient privacy-preserving pay-by-phone system for regulated parking areas," *Int. J. Inf. Secur.*, vol. 20, pp. 715–727, 2020.

[8] M. M. Badr, W. Al Amiri, M. M. Fouda, M. M. Mahmoud, A. J. Aljohani, and W. Alasmary, "Smart parking system with privacy preservation and reputation management using blockchain," *IEEE Access*, vol. 8, pp. 150823–150843, 2020.

[9] C. Zhang et al., "BSFP: Blockchain-enabled smart parking with fairness, reliability and privacy protection," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6578–6591, Jun. 2020.

[10] P. Dzurenda, C. A. Tafalla, S. Ricci, and L. Malina, "Privacy-preserving online parking based on smart contracts," in *Proc. 16th Int. Conf. Availability Rel. Secur.*, 2021, pp. 1–10.

[11] L. Wang, X. Lin, E. Zima, and C. Ma, "Towards airbnb-like privacy-enhanced private parking spot sharing based on blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 2411–2423, Mar. 2020.

[12] S. Noether et al., "Ring confidential transactions," *Ledger*, vol. 1, pp. 1–18, 2016.

[13] J. Blömer, J. Bobolz, D. Diemert, and F. Eidens, "Updatable anonymous credentials and applications to incentive systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1671–1685.

[14] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. Annu. Int. cryptol. Conf.*, 2004, pp. 41–55.

[15] H. Liu, H. Ning, Y. Zhang, and L. T. Yang, "Aggregated-proofs based privacy-preserving authentication for V2G networks in the smart grid," *IEEE Trans. Smart Grid*, vol. 3, no. 4, pp. 1722–1733, Dec. 2012.

[16] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11169–11180, Nov. 2018.

[17] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-preserving smart parking navigation supporting efficient driving guidance retrieval," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6504–6517, Jul. 2018.

[18] J. Ni, X. Lin, and X. Shen, "Toward privacy-preserving valet parking in autonomous driving era," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2893–2905, Mar. 2019.

[19] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2467–2476, Sep. 2017.

[20] A. Maria, V. Pandi, J. D. Lazarus, M. Karuppiah, and M. S. Christo, "BBAAS: Blockchain-based anonymous authentication scheme for providing secure communication in VANETs," *Secur. Commun. Netw.*, vol. 2021, pp. 1–11, 2021.

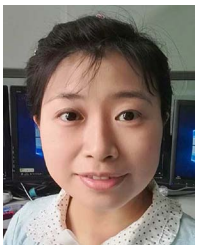
[21] R. Lu, X. Lin, H. Zhu, and X. Shen, "An intelligent secure and privacy-preserving parking scheme through vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2772–2785, Jul. 2010.

[22] T. W. Chim, S.-M. Yiu, L. C. Hui, and V. O. Li, "VSPN: VANET-based secure and privacy-preserving navigation," *IEEE Trans. Comput.*, vol. 63, no. 2, pp. 510–524, Feb. 2014.

[23] J. Timpner, D. Schürmann, and L. Wolf, "Trustworthy parking communities: Helping your neighbor to find a space," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 1, pp. 120–132, Jan./Feb. 2016.

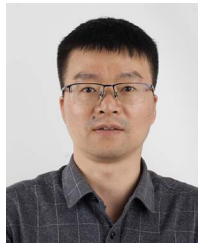
[24] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology*. Berlin, Germany: Springer, 1983, pp. 199–203.

[25] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Proc. Conf. Theory Appl. Cryptogr.*, 1988, pp. 319–327.
[26] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-Cash," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, 2005, pp. 302–321.
[27] B. D. Vo, "A fair payment system with online anonymous transfer," Ph.D. dissertation, Massachusetts Institute of Technology, 2006.
[28] K. Wei, A. J. Smith, Y.-F. Chen, and B. Vo, "WhoPay: A scalable and anonymous payment system for peer-to-peer environments," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 13–13.
[29] S. Canard, D. Pointcheval, O. Sanders, and J. Traoré, "Divisible e-cash made practical," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2015, pp. 77–100.
[30] B. Lian, G. Chen, J. Cui, and M. Ma, "Compact E-Cash with efficient coin-tracing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 220–234, Jan./Feb. 2021.
[31] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proc. Cryptographers' Track RSA Conf.*, 2016, pp. 111–126.
[32] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2003, pp. 301–312.
[33] N. P. Smart, "The discrete logarithm problem on elliptic curves of trace one," *J. cryptol.*, vol. 12, no. 3, pp. 193–196, 1999.
[34] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *J. Cryptol.*, vol. 7, no. 1, pp. 1–32, 1994.
[35] I. Damgård, "On σ-protocols," *Lecture Notes*, University of Aarhus, Department for Computer Science, 2002.
[36] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proc. Int. Conf. Secur. Commun. Netw.*, 2002, pp. 268–289.
[37] J. Camenisch and A. Lysyanskaya, "Signature schemes and anonymous credentials from bilinear maps," in *Proc. Annu. Int. cryptol. Conf.*, 2004, pp. 56–72.
[38] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[39] D. Pointcheval, O. Sanders, and J. Traoré, "Cut down the tree to achieve constant complexity in divisible e-cash," in *Proc. IACR Int. Workshop Public Key Cryptogr.*, 2017, pp. 61–90.
[40] F. Bourse, D. Pointcheval, and O. Sanders, "Divisible e-cash from constrained pseudo-random functions," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2019, pp. 679–708.
[41] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.



Yang Yang (Member, IEEE) received the BSc degree from Xidian University, Xi'an, China, in 2006 and the PhD. degree from Xidian University, China, in 2011. She is a full professor with College of Computer Science and Big Data, Fuzhou University. She is also a research scientist with School of Computing and Information System, Singapore Management University. Her research interests are in the area of information security and privacy protection. She has published more than 60 papers in TIFS, TDSC, TSC, TCC, TII, etc.

Wenyi Xue received the BSc degree from Fuzhou University, Fuzhou, China, in 2020. He is currently working toward the PhD degree under supervision of Prof. Yang Yang in College of Computer Science and Big Data, Fuzhou University, Fujian, China. His research interests are in the area of privacy protection and zero-knowledge proof.



Yonghua Zhan received the master degree from Fuzhou University, Fuzhou, China, in 2017. He is currently working toward the PhD degree under supervision of Prof. Yang Yang in College of Computer Science and Big Data, Fuzhou University, Fujian, China. His research interests are in the area of privacy protection and IoT.

Minming Huang received the BSc and MSc degrees from Fuzhou University, Fuzhou, China, in 2007 and 2010, respectively. Currently, he is a senior research enginner with School of Computing and Information System, Singapore Management University. His research interests are in the area of privacy protection and financial technology (FinTech).

Yingjiu Li received the PhD degree from George Mason University, in 2003. He had been a faculty member at Singapore Management University from 2003 to 2019. Now, he is a ripple professor with Department of Computer and Information Science, University of Oregon. He has published more than 130 papers in Cybersecurity, and co-authored two academic books. His research interests include IoT security and privacy, mobile security, and data security and privacy.

Robert H. Deng (Fellow, IEEE) is an AXA chair professor of Cybersecurity in the School of Computing and Information Systems, Singapore Management University. His research interests include data security, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, etc.